Preface

The need for a comprehensive survey-type exposition on formal languages and related mainstream areas of computer science has been evident for some years. In the early 1970s, when the book *Formal Languages* by the second-mentioned editor appeared, it was still quite feasible to write a comprehensive book with that title and include also topics of current research interest. This would not be possible anymore. A standard-sized book on formal languages would either have to stay on a fairly low level or else be specialized and restricted to some narrow sector of the field.

The setup becomes drastically different in a collection of contributions, where the best authorities in the world join forces, each of them concentrating on their own areas of specialization. The present three-volume Handbook constitutes such a unique collection. In these three volumes we present the current state of the art in formal language theory. We were most satisfied with the enthusiastic response given to our request for contributions by specialists representing various subfields. The need for a Handbook of Formal Languages was in many answers expressed in different ways: as an easily accessible historical reference, a general source of information, an overall course-aid, and a compact collection of material for self-study. We are convinced that the final result will satisfy such various needs.

The theory of formal languages constitutes the stem or backbone of the field of science now generally known as theoretical computer science. In a very true sense its role has been the same as that of philosophy with respect to science in general: it has nourished and often initiated a number of more specialized fields. In this sense formal language theory has been the origin of many other fields. However, the historical development can be viewed also from a different angle. The origins of formal language theory, as we know it today, come from different parts of human knowledge. This also explains the wide and diverse applicability of the theory. Let us have a brief look at some of these origins. The topic is discussed in more detail in the introductory Chapter 1 of Volume 1.

The main source of the theory of formal languages, most clearly visible in Volume 1 of this Handbook, is *mathematics*. Particular areas of mathematics important in this respect are combinatorics and the algebra of semigroups and monoids. An outstanding pioneer in this line of research was

Axel Thue. Already in 1906 he published a paper about avoidable and unavoidable patterns in long and infinite words. Thue and Emil Post were the two originators of the formal notion of a rewriting system or a grammar. That their work remained largely unknown for decades was due to the difficult accessibility of their writings and, perhaps much more importantly, to the fact that the time was not yet ripe for mathematical ideas, where noncommutativity played an essential role in an otherwise very simple setup.

Mathematical origins of formal language theory come also from mathematical logic and, according to the present terminology, computability theory. Here the work of Alan Turing in the mid-1930s is of crucial importance. The general idea is to find models of computing. The power of a specific model can be described by the complexity of the language it generates or accepts. Trends and aspects of mathematical language theory are the subject matter of each chapter in Volume 1 of the Handbook. Such trends and aspects are present also in many chapters in Volumes 2 and 3.

Returning to the origins of formal language theory, we observe next that much of formal language theory has originated from *linguistics*. In particular, this concerns the study of grammars and the grammatical structure of a language, initiated by Noam Chomsky in the 1950s. While the basic hierarchy of grammars is thoroughly covered in Volume 1, many aspects pertinent to linguistics are discussed later, notably in Volume 2.

The *modeling* of certain objects or phenomena has initiated large and significant parts of formal language theory. A model can be expressed by or identified with a language. Specific tasks of modeling have given rise to specific kinds of languages. A very typical example of this are the L systems introduced by Aristid Lindenmayer in the late 1960s, intended as models in developmental biology. This and other types of modeling situations, ranging from molecular genetics and semiotics to artificial intelligence and artificial life, are presented in this Handbook. Words are one-dimensional, therefore linearity is a feature present in most of formal language theory. However, sometimes a linear model is not sufficient. This means that the language used does not consist of words (strings) but rather of trees, graphs, or some other nonlinear objects. In this way the possibilities for modeling will be greatly increased. Such extensions of formal language theory are considered in Volume 3: languages are built from nonlinear objects rather than strings.

We have now already described the contents of the different volumes of this Handbook in brief terms. Volume 1 is devoted to the mathematical aspects of the theory, whereas applications are more directly present in the other two volumes, of which Volume 3 also goes into nonlinearity. The division of topics is also reflected in the titles of the volumes. However, the borderlines between the volumes are by no means strict. From many points of view, for instance, the first chapters of Volumes 2 and 3 could have been included in Volume 1.

We now come to a very important editorial decision we have made. Each of the 33 individual chapters constitutes its own entity, where the subject matter is developed from the beginning. References to other chapters are only occasional and comparable with references to other existing literature. This style of writing was suggested to the authors of the individual chapters by us from the very beginning. Such an editorial policy has both advantages and disadvantages as regards the final result. A person who reads through the whole Handbook has to get used to the fact that notation and terminology are by no means uniform in different chapters; the same term may have different meanings, and several terms may mean the same thing. Moreover, the prerequisites, especially in regard to mathematical maturity, vary from chapter to chapter. On the positive side, for a person interested in studying only a specific area, the material is all presented in a compact form in one place. Moreover, it might be counterproductive to try to change, even for the purposes of a handbook, the terminology and notation already well-established within the research community of a specific subarea. In this connection we also want to emphasize the diversity of many of the subareas of the field. An interested reader will find several chapters in this Handbook having almost totally disjoint reference lists, although each of them contains more than 100 references.

We noticed that guaranteed timeliness of the production of the Handbook gave additional impetus and motivation to the authors. As an illustration of the timeliness, we only mention that detailed accounts about DNA computing appear here in a handbook form, less than two years after the first ideas about DNA computing were published.

Having discussed the reasons behind our most important editorial decision, let us still go back to formal languages in general. Obviously there cannot be any doubt about the mathematical strength of the theory – many chapters in Volume 1 alone suffice to show the strength. The theory still abounds with challenging problems for an interested student or researcher. Mathematical strength is also a necessary condition for applicability, which in the case of formal language theory has proved to be both broad and diverse. Some details of this were already mentioned above. As the whole Handbook abounds with illustrations of various applications, it would serve no purpose to try to classify them here according to their importance or frequency. The reader is invited to study from the Handbook older applications of context-free and contextual grammars to linguistics, of parsing techniques to compiler construction, of combinatorics of words to information theory, or of morphisms to developmental biology. Among the newer application areas the reader may be interested in computer graphics (application of L systems, picture languages, weighted automata), construction and verification of concurrent and distributed systems (traces, omega-languages, grammar systems), molecular biology (splicing systems, theory of deletion), pattern matching, or cryptology, just to mention a few of the topics discussed in the Handbook.

About Volume 1

Some brief guidelines about the contents of the present Volume 1 follow. Chapter 1 is intended as an introduction, where also historical aspects are taken into account. Chapters 2, 3, 5, 6, 8, 9 each give a comprehensive survey of one important subarea of the basic theory of formal languages. The innovative nature of these surveys will become apparent to a knowledgeable reader. Indeed, at least some of these surveys can be classified as the best or first-of-its-kind survey of the area. While the three first-mentioned chapters (2, 3, and 5) are basic for the grammatical or computational aspects of the theory, the remaining three chapters (6, 8, and 9) are basic for the algebraic aspects. Grammatical (resp. algebraic) issues are discussed further in Chapters 4 and 12 (resp. 7, 10, and 11).

Acknowledgements

We would like to express our deep gratitude to all the authors of the Handbook. Contrary to what is usual in case of collective works of this kind, we hoped to be able to keep the time schedule – and succeeded because of the marvellous cooperation of the authors. Still more importantly, thanks are due because the authors were really devoted to their task and were willing to sacrifice much time and energy in order to achieve the remarkable end result, often exceeding our already high expectations.

The Advisory Board consisting of J. Berstel, C. Calude, K. Culik II, J. Engelfriet, H. Jürgensen, J. Karhumäki, W. Kuich, M. Nivat, G. Păun, A. Restivo, W. Thomas, and D. Wood was of great help to us at the initial stages. Not only was their advice invaluable for the planning of the Handbook but we also appreciate their encouragement and support.

We would also like to extend our thanks from the actual authors of the Handbook to all members of the scientific community who have supported, advised, and helped us in many ways during the various stages of work. It would be a hopeless and maybe also unrewarding task to try to single out any list of names, since so many people have been involved in one way or another.

We are grateful also to Springer-Verlag, in particular Dr. Hans Wössner, Ingeborg Mayer, J. Andrew Ross, and Gabriele Fischer, for their cooperation, excellent in every respect. Last but not least, thanks are due to Marloes Boon-van der Nat for her assistance in all editorial stages, and in particular, for keeping up contacts to the authors.

September 1996 Grzegorz Rozenberg, Arto Salomaa

Formal Languages: an Introduction and a Synopsis

Alexandru Mateescu and Arto Salomaa

1. Languages, formal and natural

What is a language? By consulting a dictionary one finds, among others, the following explanations:

1. The body of words and systems for their use common to people who are of the same community or nation, the same geographical area, or the same cultural tradition.
2. Any set or system of signs or symbols used in a more or less uniform fashion by a number of people who are thus enabled to communicate intelligibly with one other.
3. Any system of formalized symbols, signs, gestures, or the like, used or conceived as a means of communicating thought, emotion, etc.

The definitions 1–3 reflect a notion “language” general and neutral enough for our purposes.

Further explanations are more closely associated with the spoken language and auditory aspects or are otherwise too far from the ideas of this Handbook. When speaking of formal languages, we want to construct formal grammars for defining languages rather than to consider a language as a body of words somehow given to us or common to a group of people. Indeed, we will view a *language* as a set of finite strings of symbols from a finite alphabet. Formal *grammars* will be devices for defining specific languages. Depending on the context, the finite strings constituting a language can also be referred to as *words*, *sentences*, *programs*, etc. Such a formal idea of a language is compatible with the definitions 1–3, although it neglects all semantic issues and is restricted to *written* languages.

The idea of a *formal language* being a set of finite strings of symbols from a finite alphabet constitutes the core of this Handbook. Certainly all written languages, be they natural, programming or any other kind, are contained in this idea. On the other hand, formal languages understood in this general fashion have very little form, if any. More structure has to be imposed on them, and at the same time one can go beyond the linear picture of strings. Both approaches will be tried in the present Handbook.

How does one specify a formal language? If we are dealing with a finite set of strings we can, at least in principle, specify the set simply by listing its elements. With infinite languages we have a different situation: we have to invent a finitary device to produce infinite languages. Such finitary devices can be called *grammars*, *rewriting systems*, *automata*, etc. Many stories about them will be told in this Handbook. In fact, a major part of formal language theory can be viewed as the study of finitary devices for generating infinite languages.

What is today known as the Theory of Formal Languages has emerged from various origins. One of the sources is *mathematics*, in particular, certain problems in combinatorics and in the algebra of semigroups and monoids. A pioneer in this line of research was Axel Thue at the beginning of the 20th century; in [11, 12] he investigated avoidable and unavoidable patterns in long and infinite words. Together with Emil Post [8], Thue also introduced the formal notion of a rewriting system or a grammar.

Another shade in the mathematical origins of formal language theory comes from logic and, according to the current terminology, the theory of computing. Here the work of Alan Turing [13] is of crucial importance. The general idea is to find models of computing. The power of a specific model can be described by the complexity of the languages it generates or accepts.

Trends and aspects of mathematical language theory will be the subject matter of this Volume I of the Handbook. Same scattered glimpses of it will be presented already in Section 2 of this chapter.

It is quite obvious and natural that much of formal language theory has originated from *linguistics*. Indeed, one can trace this development very far in the past, as will be pointed out later on in this section. Specifically, the study of grammars initiated by Noam Chomsky [1] has opened new vistas in this development.

Many parts of formal language theory have originated from *modeling* certain objects or phenomena. A model can be expressed by or identified with a language. Specific tasks of modeling have given rise to specific types of languages. Very typical examples are L systems of Aristid Lindenmayer [5] intended as models in developmental biology. This and other types of modeling situations, dealing for instance with molecular genetics, semiotics, artificial life and artificial intelligence, will be presented in this Handbook, in particular in Volume 2. Sometimes a linear model is not sufficient. This means that the language used does not consist of strings (words) but rather of trees, graphs or some other *many-dimensional* objects. Such extensions of formal language theory will be considered in Volume 3 of this Handbook: languages are built from many-dimensional objects rather than strings and, thus, the possibilities for modeling will be greatly increased.

The remainder of this Section 1 is devoted to the study of natural languages. We give, very briefly, some glimpses of linguistics from three different points of view: historical studies, genetics and neuroscience. We only hope to give some idea about what is going on. Each of these three aspects of linguistics would require a handbook on its own!

1.1 Historical linguistics

Linguists have tried to investigate, as well as to classify, natural languages existing in the world today, some 6000 in number. The study is becoming increasingly important: it is estimated that about half of the now existing 6000 languages will die out during the next century.

How did a specific language become prevalent in the area in which it is now spoken? It could have been taken to its current territory by farmers, traders, conquerors etc. Often multidisciplinary methods are used to clarify the development. The scientific study of language, *linguistics*, can often penetrate deeper in the past than the oldest written records. Related languages are compared to construct their immediate progenitors. From progenitors one goes to their predecessors and, finally, to their ultimate ancestor or *protolanguage.*

We spoke above of "related" languages. What does this mean? Already for more than 200 years, linguists have recognized that some languages have similarities in vocabulary, grammar, formation of new constructs or the use of sounds strong enough to be grouped in the same family, stemming from a common ancestor. A *language family*, that is a family of natural languages, results by such ancestral alliances that can be traced back in history. This idea is quite different from the notion of a language family in formal language theory met very frequently in this Handbook.

Linguists are far from unanimous about the existing language families. The whole trend of trying to find similarities between different languages has been sometimes condemned as unfruitful. On the contrary, some linguists tend to emphasize the differences that make languages seem unrelated, as well as to use only small independent units in the classification. Such linguists also tend to rule out spurious relationships. The importance of reconstructing protolanguages has been also questioned.

However, certain specific language families have won wide acceptance. Among such families are the *Indo-European* family, which will be discussed in more detail below, the *Hamito-Semitic* (or *Afro-Asiatic*) family, which consists of the Semitic languages and many languages of North Africa, as well as the *Altaic* family whose major representatives are Finnish and Hungarian. We will mention still in Section 1.2 some other language families whose legitimacy has perhaps not won such a wide acceptance.

What does it mean from a practical point of view that two languages, say Finnish and Hungarian, belong to the same family and, moreover, that linguists are unanimous about this state of affairs? Does it imply that a native user of one language is able to use or understand the other language? Certainly not. Some few basic nouns, such as the words for "fish", "water" and "blood", bear easily observable resemblance in Finnish and Hungarian. There is even one word, the word "tuli", having in certain contexts the same meaning in both languages: in "TULIpunainen" and "TULIpiros" the prefix means "flaming", the whole word meaning "flaming red". However, this is merely accidental. In Hungarian "tuli" stems from the Indo-European "tulip", whereas in Finnish it is of old Uralic origin. An often quoted example is the sentence "the train is coming". It is claimed that the sentence is the same in both languages but the Finnish word for "the train" means "is coming" in Hungarian, and vice versa. The claim, however, is quite inaccurate – and so we omit further details. In general one can say that the vocabularies of Finnish and Hungarian lie as far apart as those of English and Persian (which also belong to the same well-recognized family, Indo-European languages). Among easily observable similarities between Finnish and Hungarian are certain grammatical features (no distinction between "he" and "she", no articles for nouns) and a phenomenon called vowel harmony in speech.

Let us go back to the idea of language families. In linguistics the idea dates back for more than 200 years. It is customary to date the field of linguistics to its first significant achievement, the argument propounded in 1786 by Sir William Jones, a British judge at the High Court in Calcutta, who observed relationships between Sanskrit, Greek, Latin, Gothic and Persian. (The year 1786 is such a starting point for linguistics as 1906 is for formal language theory; the first paper of Axel Thue about combinatorics on words appeared in 1906.) Common vocabulary and structural features suggested to Jones that the languages had "sprung from some common source". Nowadays this source language is known as Indo-European. More explicitly, Jones wrote about Sanskrit, Greek and Latin that "no philologer could examine them all three without believing them to have sprung from some common source which, perhaps, no longer exists".

The construction of a protolanguage applies various methods, most of which can be classified as inductive inference. Early linguists in their attempts to reconstruct the Indo-European protolanguage relied heavily on Grimm's law of "sound shift": sets of consonants displace each other in a predictable and regular fashion. This law was suggested in 1822 by Jacob Grimm, who is universally famed for the fairy tales he wrote with his brother Wilhelm. (For instance, "Snow White" and "Cinderella" are universally known as Walt Disney movies.) According to the law, softer "voiced" consonants b, d, g yield to harder "voiceless" consonants p, t, k. A typical development is

dhar	*draw*	*tragen*
(*Sanskrit*)	(*English*)	(*German*)

Another development, based on different phenomena, can be depicted as follows, see Figure 1:

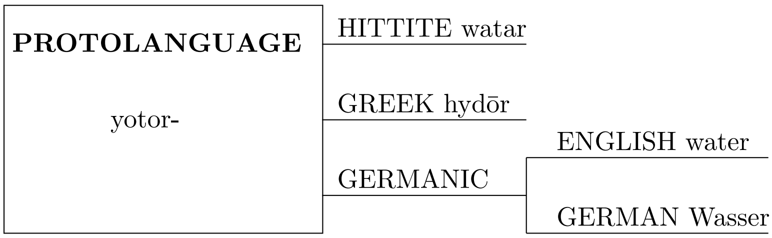
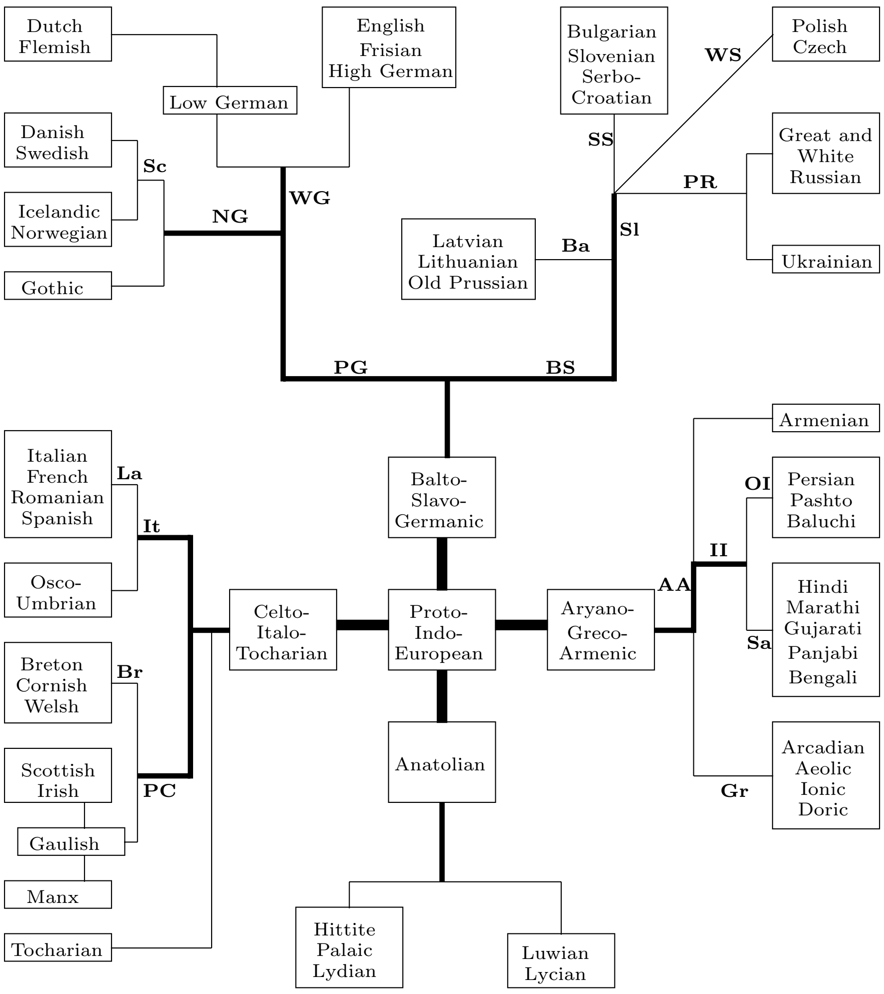


Fig. 1

Linguists have traced back diverging pathways of linguistic transformation, as well as human migration. There are many views about the Indo-European protolanguage and its homeland. Rules such as Grimm's law were used to construct an Indo-European vocabulary. On this basis, conclusions were made about how its speakers lived. The landscape and climate described by the vocabulary was originally placed in Europe between Alps and the Baltic and North seas. More recent evidence [2] places the probable origin of the Indo-European language in western Asia. The language described by the reconstructed Indo-European protolanguage is mountainous – there are many words for high mountains lakes and rapid rivers. According to [2], the vocabulary fits the landscape of eastern Transcaucasia, where the protolanguage flourished some 6, 000 years ago. The protolanguage split into dialects which evolved into distinct languages. The latter split into further dialects, and so forth. The family tree presented in Figure 2 is based on information from [2].

The time depth of the Indo-European protolanguage is only about 6, 000 years. Attempts have been made to go deeper into the origins. If the aim is a single protolanguage, one has to go back probably well beyond 20, 000 years. A rather well-known macrofamily, called *Nostratic*, comprises the Indo-European, Hamito-Semitic and Altaic families and is estimated to date back some 15, 000 years.

The following table, see Figure 3, compares some Indo-European languages, as well as two languages outside the family.



Sc-Scandinavian			**WS**-Western Slavic
La-Latin			**PR**-Proto Russian
Br-Brittanic		**SS**-Southern Slavic	**OI**-Old Iranian
NG-Northern Germanic		**Sl**-Slavic	**II**-Indo Iranian
It-Italic	**WG**-Western Germanic	**Ba**-Baltic	**AA**-Armeno Aryan
PC-Proto Celtic	**PG**-Proto Germanic	**BS**-Balto Slavic	**Sa**-Sanskrit
			Gr-Greek

Fig. 2

English	**Old German**	**Latin**	**Romanian**
One	Ains	Unus	Unu
Two	Twai	Duo	Doi
Three	Thrija	Tres	Trei
Four	Fidwor	Quattuor	Patru
Five	Fimf	Quinque	Cinci
Six	Saihs	Sex	Şase
Seven	Sibum	Septem	Şapte
Eight	Ahtau	Octo	Opt
Nine	Niun	Novem	Nouă
Ten	Taihum	Decem	Zece

English	**Greek**	**Sanskrit**	**Japanese**	**Finnish**
One	Heis	Ekas	Hiitotsu	Yksi
Two	Duo	Dva	Futatsu	Kaksi
Three	Treis	Tryas	Mittsu	Kolme
Four	Tettares	Catvaras	Yottsu	Neljä
Five	Pente	Panca	Itsutsu	Viisi
Six	Heks	Sat	Muttsu	Kuusi
Seven	Hepta	Sapta	Nanatsu	Seitsemän
Eight	Okto	Asta	Yattsu	Kahdeksan
Nine	Ennea	Nava	Kokonotsu	Yhdeksän
Ten	Deka	Dasa	To	Kymmenen

Fig. 3

1.2 Language and evolution

The idea of linking the evolution of languages to biology goes back to Charles Darwin. He wrote in the *Descent of Man* (1871), "The formation of different languages and of distinct species, and the proofs that both have been developed through a gradual process, are curiously parallel." In Chapter 14 of *On the Origin of Species* he also stated that if the tree of genetic evolution were known, it would enable scholars to predict that of linguistic evolution.

If one wants to go deeper in the past, as indicated at the end of Section 1.1, one has to combine genetic evidence with archeological and linguistic evidence. Indeed, molecular genetics can test some elements of proposed theories of the evolution of languages. One compares gene frequencies in various populations and converts the data into a structure, where genetic distance can be represented. One is then able to see to what extent genetic relationships confirm predictions arising from supposed evolution of languages and language families. Genetic and linguistic histories at least roughly correspond because both diverge when populations split apart. The following chart from

[9], see Figure 4, is based on work by Luigi Luca Cavalli-Sforza and Merritt Ruhlen. The genetic closeness of populations is shown by their distance from a common branching point (left). Their linguistic closeness (in terms of language families and superfamilies) is depicted similarly on the right.

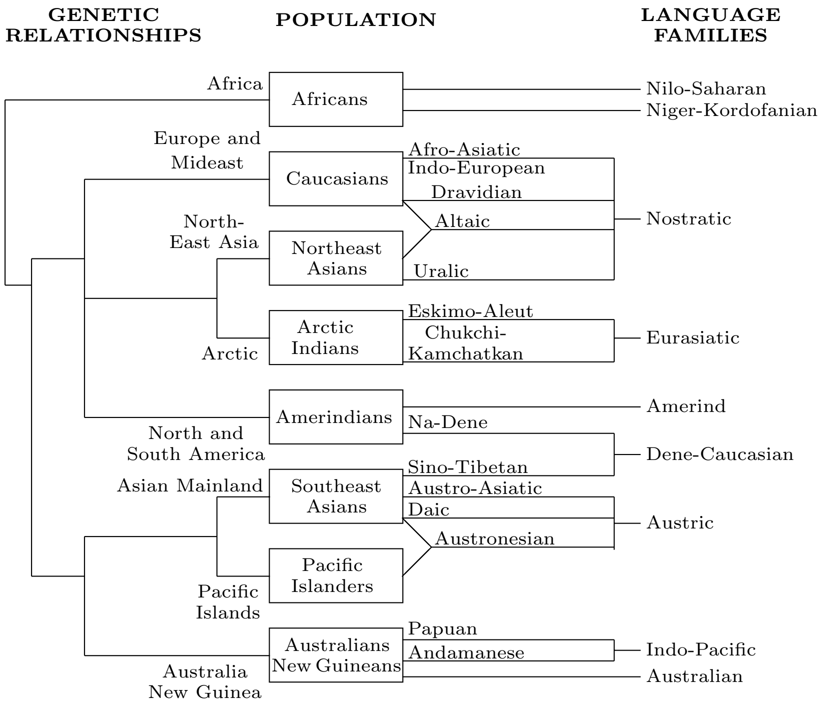


Fig. 4

An inevitable conclusion is that the distribution of genes correlates very well with that of languages. Apparently genes do not control language in a deterministic way, so the correlation is explained by history: the circumstances of birth determine the language to which one is exposed. Human populations are during the evolution fragmented into parts which settle in different locations. Then the fragments evolve both linguistic and genetic patterns ear-marked by the branching points. This explains the correlation.

It is not claimed that evolution transfers simple languages into more complex ones. It is quite widely agreed among linguists that no language, living or dead, is "better" than any other. Language alters, but it neither improves nor degenerates. For scientific discussions, Modern English maybe is better than Old English but the potential resources of both languages are the same. However, as pointed out by [9], earlier generations of linguists had no such reservations about ranking languages. August Schleicher in the 19th century classified languages according to their structures. In Schleicher's view, Chinese is an "isolating" language using simple elements and, consequently, is more "primitive" than Turkish, which builds its words from distinct forms. He put "inflecting" languages like German higher. Sanskrit was ranked highest because its inflections were so elaborate. On these terms, the languages of many hunter-gatherers are elevated above those of the linguists themselves!

1.3 Language and neural structures

How much preprogramming is there in the brain when a child starts learning a language? Any child can learn any language without making as many grammatical and other mistakes as one would expect from a structure without programming, a *tabula rasa*. It can be claimed that language is as innate in the infant as flight is in the eaglet. This means that children do not so much learn language as somehow just develop it in response to a stimulus.

Taking into account the enormous complexity of language phenomena, many researchers have wondered whether the neural machinery involved will ever be understood to any reasonable extent. Indeed, the question about the origin of language in a neuroscientific sense has often been considered hopeless. Linguistic Society of Paris banned its discussion in 1866 because it had generated so much mere talk and so little real knowledge, if any. In spite of the ban, the discussion has continued. Many theories have been presented about the origin of language, starting from onomatopoetic words such as "cuckoo", from emotional interjections, from oral gestures etc. Especially recently the progress in understanding the brain structures responsible for language has accelerated significantly. Tools such as magnetic resonance imaging (MRI) have made it possible to locate brain lesions accurately in patients suffering from aphasia. In such a way specific language deficits have been correlated with damage to particular regions of the brain. Positron emission tomography (PET) makes it possible to study brain activities of healthy persons engaged in linguistic tasks.

Still all important questions remain to be answered about how the brain stores and processes language. However, the motivation to clarify at least some of the issues is great. Language is a superb means of communication, increasing in importance as the concepts become more abstract. Try to communicate without words the background for the events in Eastern Europe in 1989, or the intricacies of Hamlet!

2. Glimpses of mathematical language theory

We now return to the actual topic of our Handbook, the Theory of Formal Languages. It was already indicated in Section 1 above that certain branches of mathematics constitute a major source of origin for formal language theory. The term "mathematical language theory" describes mathematical (algebraic) aspects of formal language theory – the main emphasis is on the mathematical theory rather than any applications. The purpose of this section is to give the reader some scattered glimpses of mathematical language theory. The idea is not to present any coherent theory but rather to give some views about the very basics, about words and languages. We want to give an uninitiated reader some feeling about what is going on. We have tried to select nontrivial problems whose solutions are presentable without too much technical apparatus and yet readable without previous knowledge or consulting other sources. Section 3 will be quite different in nature. It is a telegraphic review of (at least some) basic concepts and results in formal language theory. The purpose has been to collect basic formal language theory in one place for quick reference. Our original idea was to add another section describing briefly the contents of the Handbook. However, such a description already appears in the Prefaces. Due to the marvellous timeliness of the individual authors, it was not possible to write anything more comprehensive without endangering the publication schedule.

2.1 Words and languages

An *alphabet* is a finite nonempty set. The elements of an alphabet Σ are called *letters* or *symbols.* A *word* or *string* over an alphabet Σ is a finite sequence consisting of zero or more letters of Σ, whereby the same letter may occur several times. The sequence of zero letters is called the *empty word*, written λ. Thus, λ, 0, 1, 110, 00100 are words over the "binary" alphabet $\Sigma = \{0, 1\}$. The set of all words (resp. of all nonempty words) over an alphabet Σ is denoted by Σ^* (resp. Σ^+). Observe that Σ^* and Σ^+ are always infinite. If x and y are words over Σ, then so is their *catenation* (or *concatenation*) xy, obtained by juxtaposition, that is, writing x and y after one another. Catenation is an associative operation and the empty word λ acts as an identity: $w\lambda = \lambda w = w$ holds for all words w. Because of the associativity, we may use the notation w^i in the usual way. By definition, $w^0 = \lambda$. In algebraic terms, Σ^* and Σ^+ are the free *monoid* and *semigroup* generated by Σ with respect to the operation of catenation and with the unit element λ.

The notions introduced above will be very basic throughout this Handbook. The notation may vary, for instance, the empty word is often denoted by ε. We will denote alphabets by Σ or V, the latter coming from "vocabulary". The elements are thought as indivisible units; it depends on the viewpoint whether they are called "letters" or, coming from a vocabulary, "words".

We continue with some central terminology concerning words. The *length* of a word w, in symbols $\mid w \mid$, is the number of letters in w when each letter is counted as many times as it occurs. Again by definition, $\mid \lambda \mid = 0$. A word v is a *subword* of a word w if there are words u_1 and u_2 (possibly empty) such that $w = u_1 v u_2$. If $u_1 = \lambda$ (resp. $u_2 = \lambda$) then v is also called a *prefix* of w (resp. a *suffix* of w). Observe that w itself and λ are subwords, prefixes and suffixes of w. Other subwords, prefixes and suffixes are called *nontrivial.*

Let us write a word w in the form $w = u_1v_1u_2v_2 \dots u_nv_n$, for some integer n and words u_i, v_i, some of them possible empty. Then the word $v = v_1v_2 \dots v_n$ is a *scattered subword* of w. (The notion will be the same if we say that the word $u = u_1u_2 \dots u_n$ is a scattered subword of w. This follows because we may choose some of the u's and v's to be empty.) Thus, a scattered subword is not a coherent segment but may consist of parts picked up from here and there, without changing the order of letters. The French school emphasizes the coherence by using the term "factor" for our "subwords" and saving the term "subword" for our scattered subwords. A nonempty word w is *primitive* if it is not a proper power, that is, the equation $w = u^i$ does not hold for any word u and integer $i \geq 2$. Every word w possesses a unique *primitive root* u: u is the shortest word such that $w = u^i$, for some $i \geq 1$. Obviously, a nonempty word is primitive iff its primitive root equals the word itself.

Words $v = xy$ and $w = yx$ are termed *conjugates.* Thus, a conjugate of a word is obtained if a prefix is transferred to become a suffix. The prefix may be empty; a word is always its own conjugate. Clearly, if u and v are conjugates of w, then also u and v are conjugates among themselves. (Thus, the relation is an equivalence.) If w is primitive, so are its conjugates. Conjugates of a word are often called also *circular variants.*

Thue's work [11, 12] at the beginning of this century dealt with avoidable and unavoidable patterns occurring in long words. Let us consider the pattern $xx = x^2$. A word w is *square-free* if it contains no subword xx, where x is a nonempty word. Thus, a word being square-free means that it avoids the pattern xx. Now the size of the alphabet becomes quite significant. Assume that we are dealing with the binary alphabet $\{0, 1\}$. Can we construct long square-free words? No. The pattern xx is unavoidable as soon as the length of the word is at least 4. The words 010 and 101 of length 3 are indeed square-free – they are the only square-free words of length 3. But we cannot continue them with a further letter without losing square-freeness. Take the word 010. Both continuations 0100 and 0101 contain a square as subword.

Things are different if the alphabet contains at least 3 letters. Thue showed how to construct in this case infinite square-free sequences of letters. He also showed how to construct infinite *cube-free* sequences (that is, avoiding the pattern $xxx = x^3$) over a binary alphabet. The reader is referred to [10] for compact proofs of these nontrivial results.

We now proceed from words to languages. Subsets, finite or infinite, of Σ^* are referred to as (*formal*) *languages* over Σ. Thus,

$$L_1 = \{\lambda, 0, 111, 1001\} \text{ and } L_2 = \{0^p \mid p \text{ prime }\}$$

are languages over the binary alphabet. A finite language can always, at least in principle, be defined as L_1 above: by listing all of its words. Such a procedure is not possible for infinite languages. Some finitary specification other than simple listing is needed to define an infinite language. Much of formal language theory deals with such finitary specifications of infinite languages: grammars, automata etc.

Having read Section 1 of this chapter, the reader might find our terminology somewhat unusual: a language should consist of sentences rather than words, as is the case in our terminology. However, this is irrelevant because we have to choose some terminology. Ours reflects the mathematical origin. But the basic set, its elements and strings of elements could equally well be called “vocabulary”, “words” and “sentences”.

Various *operations* will be defined for languages: how to get new languages from given ones. Regarding languages as sets, we may immediately define the *Boolean* operations of *union*, *intersection* and *complementation* in the usual fashion. The operation of *catenation* is extended to concern languages in the natural way:

$$L_1L_2 = \{w_1w_2 \mid w_1 \in L_1 \text{ and } w_2 \in L_2\}.$$

The notation L^i is extended to concern languages, now $L^0 = \{\lambda\}$. The *catenation closure* or *Kleene star* (resp. *Kleene plus*) of a language L, in symbols L^* (resp. L^+) is defined to be the union of all nonnegative powers of L (resp. of all positive powers of L). Observe that this definition is in accordance with our earlier notations Σ^* and Σ^+ if we understand Σ as the finite language whose words are the singleton letters.

The operations of union, catenation, and Kleene star are referred as *regular* operations. A language L over Σ is termed *regular* if L can be obtained from the “atomic” languages $\emptyset$ (the empty language) and $\{a\}$, where a is a letter of Σ, by applying regular operations finitely many times. By applying union and catenation, we obviously get all finite languages. The star operation is needed to produce infinite languages. When we speak of the *family* of regular languages, we mean all languages that are regular over some Σ, that is, the alphabet may vary from language to language. This family is very central in formal language theory. It corresponds to strictly finite computing devices, finite automata. Regular languages will be the subject matter of the next chapter of this Handbook. The family has many desirable formal properties. It is *closed* under most of the usual operations for languages, in particular, under all Boolean operations.

A language L over Σ is *star-free* if L can be obtained from the atomic languages $\{a\}$, where $a \in \Sigma$, by applying Boolean operations (complement is taken with respect to Σ^*) and catenation finitely many times. Since regular languages are closed under Boolean operations, it follows that every star-free language is regular. We will return to star-free languages in Section 2.3.

2.2 About commuting

A feature common for all languages, formal or natural, is the noncommutativity of letters and sounds. The English words “no” and “on” are very different. The monoid Σ^* is noncommutative except when Σ consists of one letter a only: the word a^{i+j} results if we catenate a^i and a^j, independently of which one comes first. Because of the noncommutativity, the mathematical problems about formal languages are rather tricky, and results from classical algebra or analysis are seldom applicable.

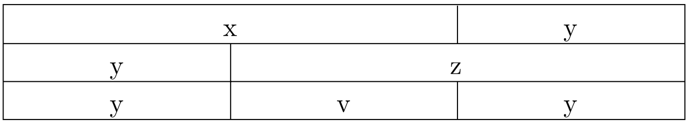
In this Section 2.2 we will present the basic results about commuting. What can be said about words satisfying certain specific commutativity conditions? The results belong to *combinatorics on words*, an area to which a chapter is devoted in this Handbook. Such results are very useful in situations, where we meet the same word in different positions and want to make conclusions about the word.

The first result is very fundamental: what can be said about a word, where the same word appears both as a prefix and as a suffix?

Theorem 2.1. *Assume that $xy = yz$, for some words x, y, z, where $x \neq \lambda$. Then there are words u, v and a nonnegative integer k such that*

$$x = uv,\ y = (uv)^k u = u(vu)^k,\ z = vu.$$

Proof. Assume first that $| x | \geq | y |$. Then from the equation $xy = yz$ we see, by reading prefixes of length $| x |$, that $x = yv$, for some word v. (If $| x | = | y |$ then $v = \lambda$.) The situation can be depicted as follows:



We may obviously choose $k = 0$ and $u = y$. Then $x = yv = uv$, $y = (uv)^0 u$ and $z = vy = vu$.

If $| x | < | y |$, we use induction on the length of y. The basis is clear. If $| y | = 0$, then $y = \lambda$, and we may choose $u = \lambda$, $v = x = z$, $k = 0$. (Recall that $v^0 = \lambda$.)

Suppose, inductively, that the assertion holds for all $| y | \leq n$ and consider the case $| y | = n + 1$. Because $| x | < | y |$, we obtain $y = xw$ by reading prefixes of length $| y |$ from the equation $xy = yz$. By substituting, we infer that $xxw = xwz$, hence $xw = wz$. We may now apply the inductive hypothesis: because $x \neq \lambda$, we have $| w | < | y |$ and, hence, $| w | \leq n$. (Observe that we cannot find u, v as required if $x = z = \lambda$ and $y \neq \lambda$. That's why we have the assumption $x \neq \lambda$ in Theorem 2.1.)

Consequently, for some u, v and k,

$$x = uv,\ y = (uv)^k u = u(vu)^k,\ z = vu.$$

Because $y = xw = uv(uv)^k u = (uv)^{k+1} u$, we have completed the inductive step. □

Our next result answers the basic question about commuting: when is it possible that $xy = yx$ holds between words x and y? The result is that x and y must be powers of the same word, so we are essentially back in the case of a one-letter alphabet. Theorems 2.1 and 2.2 are often referred to as "Lyndon's Theorem", due to [6].

Theorem 2.2. *If $xy = yx$ holds between nonempty words x and y, then there is a word w and nonnegative integers i, j such that $x = w^i$ and $y = w^j$.*

Proof. The proof is by induction on the length of xy. The conclusion is immediate for the case $\mid xy \mid = 2$. Now suppose the Theorem is true for all xy with $\mid xy \mid \leq n$. Let $\mid xy \mid = n + 1$. Then by Theorem 2.1, $x = uv$, $y = (uv)^k u$ for some words u, v and nonnegative integer k. Observe that $uv(uv)^k u = (uv)^k uuv$; hence $uvu = uuv$ and $uv = vu$. Since $\mid uv \mid \leq n$, by the induction hypothesis u and v are powers of a common word w, i.e., $u = w^p$ and $v = w^q$, for some nonnegative integers p and q. It follows that $x = uv = w^{p+q}$ and $y = (uv)^k u = w^{k(p+q)+p}$. Hence, $x = w^i$ and $y = w^j$, where $i = p + q$ and $j = k(p + q) + p$. □

Theorem 2.2 can also be expressed by saying that the equation $xy = yx$ is *periodicity forcing*: the equation admits only "periodic" solutions, where the unknowns are repetitions of the same period. Some other such periodicity forcing equations are known, for instance,

$$xyz = y^i,\ i \geq 2.$$

The equation can hold between x, y, z only if they are powers of the same word.

As an application of our commutativity considerations, we present a result concerning conjugates and primitive words.

Theorem 2.3. *A word w has $\mid w \mid$ different conjugates iff w is primitive.*

Proof. Clearly, w can never have more than $\mid w \mid$ conjugates. The "only if"-part is clear. If $w = u^i$, $i \geq 2$, then w equals the conjugate obtained by transferring one u from the prefix position to the suffix position.

To prove the "if"-part, suppose that w is primitive. Proceed indirectly, assuming that two of w's conjugates are equal: $w_1 = w_2$. Since also w_1 is a conjugate of w_2, we obtain

$$w_1 = xy = yx = w_2,\ x \neq \lambda,\ y \neq \lambda.$$

By Theorem 2.2, x and y are powers of the same word u. Since both x and y are nonempty, $w_1 = w_2 = u^i$, $i \geq 2$, implying that w_1 and w_2 are imprimitive. As a conjugate of w_1, the word w is also not primitive, a contradiction. Hence, all conjugates of w are different. There are $\mid w \mid$ of them, as seen by transferring the letters of w, one after the other, from the beginning to the end. □

2.3 About stars

The operation of Kleene star is a powerful one. It produces infinite languages from finite ones. Together with the other two regular operations, union and catenation, it can lead to very complicated compositions. In fact, for any k, there are regular languages that cannot be represented with fewer than k nested star operations.

We present in this section two properties of the star operation. The first one tells us that, as regards languages over one letter, the star of the language is always very simple, no matter how complicated the original language is. The basic reason for this simplicity is again commutativity. Because words over one letter commute, the intricacies of the star language vanish.

Theorem 2.4. *For every language L over the alphabet $\{a\}$, there is a finite language $L_F \subseteq L$, such that $L^* = L_F^*$.*

Proof. The theorem holds for finite languages L, we just choose $L_F = L$. Assume that L is infinite and a^p is the shortest nonempty word in L, $p \geq 1$. Clearly, $\{a^p\}^* \subseteq L^*$. (This follows by the definition of the star operation. Because a^p is in L, all powers of a^p are in L^*.) If this inclusion is actually an equality (and it must be so if $p = 1$), there is nothing more to prove, we choose $L_F = \{a^p\}$. Otherwise, let a^{q_1} be the shortest word from the difference $L^* - \{a^p\}^*$. (The difference consists of words in L^* but not in $\{a^p\}^*$.) It follows that q_1 can be written in the form

$$q_1 = t_1 p + r_1,\ 0 < r_1 < p,\ t_1 \geq 1.$$

(We cannot have q_1 divisible by p because all exponents of a divisible by p are produced by $\{a^p\}^*$.) Again we see that $\{a^p, a^{q_1}\}^* \subseteq L^*$. If this is not an equality, we let a^{q_2} be the shortest word in the difference $L^* - \{a^p, a^{q_1}\}^*$. It follows that q_2 is of the form

$$q_2 = t_2 p + r_2,\ 0 < r_2 < p,\ r_2 \neq r_1,\ t_2 \geq 1.$$

(All words leaving the remainder r_1 when divided by p are obtained using a^p and a^{q_1}.)

The procedure is continued. At the kth step we obtain the word a^{q_k}, with

$$q_k = t_k p + r_k,\ 0 < r_k < p,\ r_k \neq r_1,\ r_2,\ \ldots,\ r_{k-1},\ t_k \geq 1.$$

Since there are at most p possible remainders r_i, including the remainder 0 stemming from p, we actually will have the equality

$$\{a^p,\ a^{q_1},\ \ldots,\ a^{q_s}\}^* = L^*, \text{ for some } s < p. \qquad \square$$

We will now consider languages $\{w\}^*$. Thus, the language consists of all powers of w, we write it simply w^*. The question is: when is w^* star-free? Of course, the star occurs in the definition of w^* but there may be a way of getting around it. The question is not easy. For instance, considering languages

$$(*) \qquad (1010)^*,\ (10101)^*,\ (101010)^*$$

one observes, after considerable reflection, that the middle one is star-free, whereas the others are not. Where does this state of affairs depend on? We will settle this question completely. First we will prove some auxiliary results. We assume that w is over the alphabet Σ and, moreover, Σ is the *minimal* alphabet of w: all letters of Σ actually occur in w.

Lemma 2.1. *The languages $\emptyset$, $\{\lambda\}$, Σ^* and Σ^+ are star-free over Σ.*

Proof. The claim follows by the equations

$$\emptyset = \{a\} \cap \{a^2\}, \text{ where } a \in \Sigma,$$

$$\Sigma^* = \sim \emptyset,\ \Sigma^+ = \Sigma\Sigma^*,\ \{\lambda\} = \sim \Sigma^+. \qquad \square$$

We say that a language L is *noncounting* iff there is an integer n such that, for all words x, y, z,

$$xy^n z \in L \text{ iff } xy^{n+1}z \in L.$$

Lemma 2.2. *Every star-free language is noncounting.*

Proof. Let L be a star-free language, $L \subseteq \Sigma^*$.

The proof is by induction on the structure of L, i.e. by induction on the number of Boolean operations and catenations used to define L.

If $L = \emptyset$ or $L = \{a\}$, $a \in \Sigma$, then L is noncounting for $n = 2$.

Now, assume that $L = L_1 \cup L_2$ and let n_i be the constant for which L_i is noncounting, $i = 1, 2$. Define $n = \max\{n_1, n_2\}$. Assume that $\alpha\beta^n\gamma \in L$. It follows that $\alpha\beta^n\gamma \in L_1$ or $\alpha\beta^n\gamma \in L_2$. If $\alpha\beta^n\gamma \in L_1$, then using the noncounting property $(n - n_1)$ times for L_1, we obtain that $\alpha\beta^{n_1}\gamma \in L_1$. Again, using the same property $(n - n_1 + 1)$ times for L_1, we obtain that $\alpha\beta^{n+1}\gamma \in L_1$. It follows that $\alpha\beta^{n+1}\gamma \in L_1 \cup L_2 = L$. The converse is similar, i.e., if $\alpha\beta^{n+1}\gamma \in L$, we use the same argument to prove that $\alpha\beta^n\gamma \in L$.

If $L = \sim L_1$ then obviously, L satisfies the noncounting property for $n = n_1$.

If $L = L_1 \cap L_2$, then observe that $L = \sim (\sim L_1 \cap \sim L_2)$.

Finally, assume that $L = L_1L_2$ and define $n = n_1 + n_2 + 1$. Assume that $\alpha\beta^n\gamma \in L = L_1L_2$. There are $u_i \in L_i$, $i = 1, 2$, such that $\alpha\beta^n\gamma = u_1u_2$. Observe that at least one of the words u_i, contains a subword β^m, with $m \geq n_i$, $i = 1, 2$. Without loss of generality, assume that $u_1 = \alpha\beta^m u'$ with $m \geq n_1$. Using the noncounting property $(m - n_1)$ times for L_1 we conclude that $\alpha\beta^{n_1}u' \in L_1$. Again, using the same property $(m - n_1 + 1)$ times for L_1, we deduce that $\alpha\beta^{m+1}u' \in L_1$. Hence, $\alpha\beta^{m+1}u'u_2 = \alpha\beta^{n+1}\gamma \in L_1L_2 = L$. The converse is analogous.

Hence, if L is a star-free language, then L is a noncounting language. □

The converse of Lemma 2.2 holds in the form: every regular noncounting language is star-free. We will need Lemma 2.2 only in the form indicated.

We are now in the position to establish the characterization result. The result also immediately clarifies why only the middle one of the languages (∗) is star-free.

Theorem 2.5. *The language w^*, $w \neq \lambda$, is star-free iff w is primitive.*

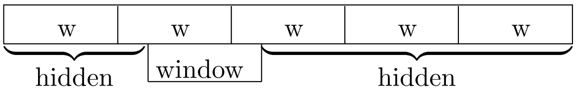
Proof. Consider first the "only if"-part, supposing that $w = u^i$, $i \geq 2$. We want to show that w^* is not star-free. By Lemma 2.2, it suffices to prove that w^* is not noncounting. Assume the contrary, and let n be the corresponding constant. Choose now, in the definition of "noncounting", $x = u^j$, $y = u$, $z = \lambda$, where j is such that $j + n$ is divisible by i. Then $xy^nz = u^{j+n} \in w^*$, whereas $xy^{n+1}z = u^{j+n+1} \notin w^*$. Consequently, w^* is not noncounting.

For the "if"-part, assume that w is primitive. To prove that w^* is star-free, let us first have a look at what kind of properties are expressible in a star-free form. The set of all words having the word x as a subword, prefix or suffix can be expressed in the form

$$\Sigma^* x \Sigma^*,\ x\Sigma^*,\ \Sigma^* x,$$

respectively. By Lemma 2.1, these sets are star-free. Similarly, $\sim (\Sigma^* x \Sigma^*)$ stands for the set of all words *not* having the word x as a subword.

Consider a word in w^*, say w^5. Let us keep most of the word hidden, and move a scanning *window* of length $|\, w \,|$ across w^5:



What words do we see through the window? At the beginning and at the end we see w. At all other times, in fact at all times, we see a word in $C(w)$, the set of *conjugates* of w. (Recall that w is its own conjugate.)

Let us now try to express in star-free terms the condition that a word x is in w^*. We assume that $x \neq \lambda$ because we can add $\{\lambda\}$ as a separate term of the union. The word x must *begin* and *end correctly*, that is, it has to belong to the intersection

$$w\Sigma^* \cap \Sigma^* w.$$

Furthermore, all *subwords* of x of length $|\, w \,|$ must be conjugates of w, that is, belong to $C(w)$. We express this by saying that no subwords of length $|\, w \,|$ and outside $C(w)$ are allowed to appear in x, that is, x belongs to the intersection ("subwords OK")

$$SUBWOK = \bigcap_{y \notin C(w), |y|=|w|} \sim (\Sigma^* y \Sigma^*)$$

(We can compute the number of words y, that is, the number of sets to be intersected. Assume that Σ has k letters. Then there are altogether $k^{|w|}$ words of length $\mid w \mid$. By Theorem 2.3, $k^{|w|} - \mid w \mid$ are outside $C(w)$.) We are now ready to write a star-free expression for w^*.

$$(**) \qquad w^* = \{\lambda\} \cup (w\Sigma^* \cap \Sigma^* w \cap SUBWOK).$$

By Lemma 2.1, the right side of the equation $(**)$ is star-free. We still have to show that our equation is correct.

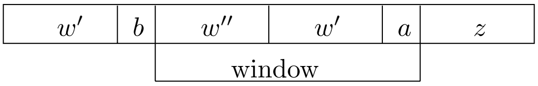
It is obvious that the left side of the equation is included in the right side: if a nonempty word is in w^*, it begins and ends correctly and has no incorrect subword and, consequently, belongs to all sets in the intersection. To prove the reverse inclusion, we assume the contrary. Let x be the shortest word on the right side of our equation $(**)$ that is not in w^*. Then $x \neq \lambda$ and we may write $x = wx_1$ (because $x \in w\Sigma^*$). We must have $x_1 \neq \lambda$ because $w\lambda = w \in w^*$.

If x_1 has the prefix w, we argue as follows. Since x has w as suffix, so does x_1. The word x_1 has only correct subwords (no subwords of length $\mid w \mid$ outside $C(w)$) because x has only correct subwords. Consequently, x_1 belongs to the right side of the equation $(**)$. This implies that $x_1 \in w^*$ because $\mid x_1 \mid < \mid x \mid$ and x was the shortest word belonging to the right side and not to w^*. But now also $x = wx_1 \in w^*$, a contradiction.

Therefore, w is not a prefix of x_1. Assume first that there is an i such that the ith letter a of x_1 differs from the ith letter b of w - let i be the smallest such index. Thus

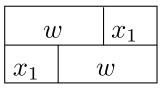
$$x_1 = w'az,\ w = w'bw'',\ a \in \Sigma,\ b \in \Sigma,\ a \neq b,$$

where the words w', w'', z may be empty. Consider now the subword $w''w'a$ of $x = wx_1$:



This subword is of length $\mid w \mid$, so it must belong to $C(w)$. On the other hand, we have clearly $w''w'b \in C(w)$. But both words $w''w'a$ and $w''w'b$ cannot be in $C(w)$ because every letter occurs the same number of times in each word of $C(w)$. Thus our assumption has led to a contradiction and, consequently, no such i exists.

The only remaining possibility is that x_1 is a proper prefix of w, $w = x_1w_1$, for some $w_1 \neq \lambda$. Since x belongs to the right side of equation $(**)$, the word w is also a suffix of x, leading to the following situation:

$x =$ 

Here w_1 is the overlapping part of the two w's. By Theorem 2.3, w is not primitive, which is a contradiction. (The same conclusion could be reached by Theorem 2.2 as well.) □

Our proof shows clearly, where and why the primitivity of w is needed. For instance, if $w = u^3$ and $x_1 = u$ then the word wx_1 belongs to the right side of the equation $(**)$ but not to w^*.

2.4 Avoiding scattered subwords

Sometimes you deal with collections of words, none of which is a subword of any other word in the collection. This happens, for instance, when a subword fulfils the purpose you have in mind equally well as the whole word. Then, from whatever language L you originally dealt with, you take only a *basis*: a subset K of L such that (*i*) no word in K is a subword of another word in K, and (*ii*) every word in L possesses some word of K as a subword. A desirable situation would be that a basis is finite. Unfortunately, this does not always happen. For the language

$$L = \{ba^ib \mid i \geq 1\},$$

the only basis is L itself because none of the words in L is subword of another word in L.

The situation becomes entirely different if, instead of subwords, scattered subwords are considered. In this case a rather surprising result can be obtained, a result that certainly is not intuitively obvious: If no word in a language K is a scattered subword of another word in K, then K is necessarily finite.

Let us use in this context the notation $v \leq w$ to mean that v is a scattered subword of w. The relation is a *partial order*. This means that the following three conditions hold for all words u, v, w.

(i) $u \leq u$,
(ii) If $u \leq v$ and $v \leq u$ then $u = v$,
(iii) If $u \leq v$ and $v \leq w$ then $u \leq w$.

"Partial" means here that u and v can also be *incomparable*:
neither $u \leq v$ nor $v \leq u$. This is the case, for instance, if $u = ab$ and $v = ba$. The notation $u \not\leq v$ means that either u and v are incomparable, or else both $v \leq u$ and $v \neq u$. Observe that the similarly defined relation in terms of subwords, rather than scattered subwords, is also a partial order.

We now prove the result already referred to above. The result is due to [3]. An earlier version, based on algebraic considerations, appeared in [4].

Theorem 2.6. *Assume that a language L contains no two distinct words v and w such that $v \leq w$. Then L is finite.*

Proof. We show that a contradiction arises if we assume that there is an infinite language with no two comparable words. This is shown by induction on the size of the alphabet. The statement clearly holds for languages over one-letter alphabet because in this case any two words are comparable. Assume inductively that the statement holds for alphabets up to a certain size but it no longer holds if we add one more letter, to get an alphabet Σ. Thus, there are languages over Σ not satisfying Theorem 2.6, whereas Theorem 2.6 holds true for all languages over smaller alphabets.

For each infinite $L \subseteq \Sigma^*$ consisting of pairwise incomparable words, there is a shortest $x_L \in \Sigma^*$ such that $x_L \not\leq y$, for all $y \in L$. (The existence of x_L can be seen as follows. We first take any $x \in L$ and $a \in \Sigma$. Then $ax \not\leq y$, for all $y \in L$. Otherwise, if there is a word $y' \in L$ with $ax \leq y'$, we infer $x \leq ax \leq y'$, which is a contradiction. We now find out whether there are words shorter than ax with the same property concerning $\leq$ and choose the shortest among them for x_L. The word x_L is not necessarily unique, only its length is unique.)

We now fix the language L in such a way that the corresponding word x_L is shortest possible. Thus, there is no infinite language $K \subseteq \Sigma^*$ consisting of pairwise incomparable words such that $x_K \in \Sigma^*$ is a shortest word with the property $x_K \not\leq y$, for all $y \in K$, and $\mid x_K \mid < \mid x_L \mid$.

Clearly, $x_L \neq \lambda$ because $\lambda \leq y$, for all y. We express x_L as a catenation of letters, some of them possibly equal:

$$x_L = a_1 a_2 \ldots a_k,\ a_i \in \Sigma.$$

If $k = 1$ and $x_L = a_1$ then the condition $x_L \not\leq y$, for all $y \in L$, implies that L is over the alphabet $\Sigma - \{a_1\}$, which contradicts the inductive hypothesis. Hence, $k \geq 2$ and $a_1 a_2 \ldots a_{k-1} \neq \lambda$.

Clearly, $\mid a_1 a_2 \ldots a_{k-1} \mid < \mid x_L \mid$.

If $a_1 a_2 \ldots a_{k-1} \not\leq y$ holds for infinitely many words $y \in L$, the subset of L consisting of those infinitely many words would contradict our choice of L and x_L. Thus, by removing a finite subset of L, we get an infinite language $L' = \{y_i \mid i = 1, 2, \ldots\} \subseteq L$ such that

$$a_1 a_2 \ldots a_{k-1} \leq y_i, \text{ for all } y_i \in L' .$$

Thus, $a_1 a_2 \ldots a_{k-1}$ appears as a scattered subword in every word of L'. For each $y_i \in L'$, we now find the occurrence of $a_1 a_2 \ldots a_{k-1}$ leftmost in the following sense. First we find the leftmost a_1 in y_i. After this occurrence of a_1, we find the leftmost a_2. After this occurrence of a_2, we find the leftmost a_3, and so forth. Thus, we write each y_i in the form

$$y_i = z_1^i a_1 z_2^i a_2 \ldots z_{k-1}^i a_{k-1} z_k^i,$$

for unique words $z^i_j \in (\Sigma - \{a_j\})^*$, $1 \leq j \leq k-1$. An important observation now is that, for all i, also $z^i_k \in (\Sigma - \{a_k\})^*$. This follows because, otherwise, $x_L = a_1 a_2 \ldots a_k \leq y_i$, a contradiction. Thus, for each fixed j, all the words z^i_j, $i \geq 1$, are over an alphabet smaller than Σ.

We now construct a decreasing sequence

$$N_1 \supseteq N_2 \supseteq N_3 \supseteq \ldots \supseteq N_k$$

of infinite sets N_j of positive integers such that,

$$(*) \qquad \text{whenever } m, n \in N_j, m < n, 1 \leq j \leq k, \text{ then } z^m_j \leq z^n_j.$$

The definition of the sets N_j proceeds inductively. For notational convenience, we let $N_0 = \{i \mid i \geq 1\}$. Assume that the infinite set N_j has already been defined. Here either $j = 0$, or else $1 \leq j \leq k-1$ and $(*)$ is satisfied. We define the set N_{j+1} as follows. Consider the set

$$Z_{j+1} = \{z^i_{j+1} \mid i \in N_j\}.$$

If the set Z_{j+1} is finite we consider, for each word $w \in Z_{j+1}$, the set

$$M(w) = \{i \mid i \in N_j \text{ and } z^i_{j+1} = w\}.$$

Since, N_j is infinite, at least one of the sets $M(w)$ is infinite. We choose N_{j+1} to be any such set; then $(*)$ is automatically satisfied.

If Z_{j+1} is infinite, we may use the inductive hypothesis because $Z_{j+1} \subseteq (\Sigma - \{a_{j+1}\})^*$. Consequently, there are only finitely many pairwise incomparable elements in Z_{j+1}. From this fact we may conclude that there is an infinite chain of elements of Z_{j+1} satisfying:

$$(**) \qquad z^{n_1}_{j+1} \leq z^{n_2}_{j+1} \leq z^{n_3}_{j+1} \leq \ldots, \text{ where } n_1 < n_2 < n_3 < \ldots$$

Let us now argue how this conclusion can be made.

If we have an infinite chain of elements of Z_{j+1}

$$(**)' \qquad z^{m_1}_{j+1} \leq z^{m_2}_{j+1} \leq z^{m_3}_{j+1} \leq \cdots,$$

we may obviously conclude the existence of $(**)$ by defining $n_1 = m_1$ and

$$n_{r+1} = min\{s \mid m_s > n_r\},\ r \geq 1.$$

On the other hand, there must be an infinite chain $(**)'$. For if all maximal chains (that is, chains that cannot be continued further) are finite, then, since the different greatest elements in the maximal chains are pairwise incomparable, infinitely many distinct chains have the same greatest element, say z^{max}_{j+1}. This means that infinitely many words w satisfy $w \leq z^{max}_{j+1}$, which is clearly impossible.

We now define the set N_{j+1} (in the case that Z_{j+1} is infinite) by $N_{j+1} = \{n_r \mid r \geq 1\}$. The definition and $(**)$ guarantee that $(*)$ is satisfied.

Having defined the set N_k, we choose two distinct numbers $m, n \in N_k$, $m < n$. Consequently, $m, n \in N_j$, for all j, $1 \leq j \leq k$. By $(*)$,

$$z_j^m \leq z_j^n,\ 1 \leq j \leq k.$$

We infer further that

$$y_m = z_1^m a_1 z_2^m a_2 \ldots z_{k-1}^m a_{k-1} z_k^m \leq z_1^n a_1 z_2^n a_2 \ldots z_{k-1}^n a_{k-1} z_k^n = y_n.$$

Thus, y_m and y_n are not after all incomparable, which is a contradiction. □

Two observations should be made regarding the proof given. We did not only exhibit two comparable words y_m and y_n but rather an infinite chain of words, each of which is a scattered subword of the next one. Indeed, the numbers of N_k in increasing order produce such a chain.

The second observation is that the use of N_0, for which $(*)$ is not required, makes it possible to combine the basis of induction and the inductive step. It is also instructive to consider the case, where Σ has only two letters. Then each of the sets Z_{j+1} is over one letter (not the same one for all j !), and the ideas of the proof become more straightforward.

Theorem 2.6 has many interesting consequences. By an *infinite word* or *ω-word* over an alphabet Σ we mean an infinite sequence of letters of Σ. For instance, an infinite decimal expansion of a real number is an infinite word over the alphabet $\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$. The following result is an immediate corollary of Theorem 2.6.

Corollary 2.1. *No matter how an infinite word is divided into blocks of finite length, one of the blocks is a scattered subword of another block.* □

Coming back to the discussion at the beginning of the Section 2.4, we say that a subset L_0 of a language L is a *basis* of L if, for every $y \in L$, there is a word $x \in L_0$ such that $x \leq y$, that is, x is a scattered subword of y. Clearly, the set of minimal elements with respect to the relation $\leq$ constitutes a basis. Thus, the following result is immediate.

Corollary 2.2. *Every language possesses a finite basis.* □

Corollary 2.2 is just another formulation of Theorem 2.6. The first documentation of the result appears in [4] in the following form.

Theorem 2.7. *(Higman) If X is any set of words formed from a finite alphabet, it is possible to find a finite subset X_0 of X such that, given a word w in X, it is possible to find w_0 in X_0 such that the letters of w_0 occur in w in their right order, though not necessarily consecutively.* □

An important remark is here in order. We have not in our discussions paid any attention to effectiveness, let alone efficiency. What is for Higman "possible to find" often leads to noncomputable tasks and undecidable problems. Decidability and complexity issues will be frequently discussed later on in this Handbook.

Some further corollaries of Theorem 2.6 could be mentioned. Starting with an arbitrary language L, we define the following two languages:

$$SSW(L, \leq) = \{z \in \Sigma^* \mid y \leq z, \text{ for some } y \in L\},$$

$$SSW(L, \geq) = \{z \in \Sigma^* \mid y \geq z, \text{ for some } y \in L\}.$$

Thus, the latter language consists of all scattered subwords of the words in L, whereas the former consists of all words "generated" by L in regard to the relation $\leq$: every word in $SSW(L, \leq)$ contains some word of L as a scattered subword. The notation SSW comes from "scattered subword".

Corollary 2.3. *For every language L, both of the languages $SSW(L, \leq)$ and $SSW(L, \geq)$ are star-free regular languages.*

Proof. By Corollary 2.2, L possesses a finite basis L_0. For a word $w = a_1 a_2 \dots a_k$, $a_i \in \Sigma$, $1 \leq i \leq k$, the language

$$SSW(\{w\}, \leq) = \Sigma^* a_1 \Sigma^* a_2 \dots \Sigma^* a_k \Sigma^*$$

is star-free (see Lemma 2.1 in Section 2.3). We now claim that

$$SSW(L, \leq) = \bigcup_{w \in L_0} SSW(\{w\}, \leq),$$

which shows that the left side is star-free. The claim follows because every word containing some word of L as a scattered subword also contains some word of L_0 as a scattered subword, and the right side of the equation consists of all words having some word of L_0 as a scattered subword.

Denote by K the complement of the language $SSW(L, \geq)$. It suffices to prove that K is star-free. By the first part of the proof we know that $SSW(L, \leq)$ is star-free. But

$$K = SSW(K, \leq).$$

Indeed, the inclusion of the left side in the right side is obvious. To prove the reverse inclusion, assume that there is a word x in the difference $SSW(K, \leq) - K$. This means that

$$x \in SSW(K, \leq) \cap SSW(L, \geq).$$

Thus, $y \leq x$, for some $y \in K$. On the other hand, x is a scattered subword of some word $z \in L$. This implies that also y is a scattered subword of z. But K consists of words that are not scattered subwords of words in L and, hence, $y \notin K$. This contradiction completes the proof. □

Corollary 2.3 is quite remarkable. Any language generates (in two different ways) a structurally very simple language. However, our previous remark applies also here: in general, we cannot find the languages $SSW(L, \leq)$ and $SSW(L, \geq)$ effectively.

Our final corollary is usually referred to as "König's Lemma". It deals with k-dimensional vectors of nonnegative integers, that is, ordered k-tuples $(m_1, m_2, \ldots, m_k)$, where each m_i is a nonnegative integer. A partial order between such vectors is defined by

$$(m_1, m_2, \ldots, m_k) \leq (n_1, n_2, \ldots, n_k) \text{ iff } m_i \leq n_i, \text{ for all } i,\ 1 \leq i \leq k.$$

Corollary 2.4. *Every set of pairwise incomparable vectors is finite.*

Proof. Associate with the vector $(m_1, m_2, \ldots, m_k)$ the word $a_1^{m_1} a_2^{m_2} \ldots a_k^{m_k}$ over the alphabet $\{a_1, a_2, \ldots, a_k\}$. Then the relation $\leq$ holds between two vectors iff it holds between the associated words, and Corollary 2.4 now immediately follows by Theorem 2.6. □

2.5 About scattered residuals

Our last illustration requires some maturity in algebra, although no specific knowledge is needed.

Let A be a set. $\mathcal{P}(A)$ denotes the set of all subsets of the set A. If $a \in \Sigma$, then $|x|_a$ is the number of occurrences of the letter a in x. If $x \in \Sigma^*$, then the *commutative closure* of x is

$$com(x) = \{w \in \Sigma^* \mid \textit{for all } a \in \Sigma, |w|_a = |x|_a\}.$$

If L is a language, then the *commutative closure* of L is

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

A language L is called *commutative* if and only if $L = com(L)$.

The *shuffle* operation between words, denoted Ш, is defined recursively by

$$(au \text{ Ш } bv) = a(u \text{ Ш } bv) \cup b(au \text{ Ш } v),$$

and

$$(u \text{ Ш } \lambda) = (\lambda \text{ Ш } u) = \{u\},$$

where $u, v \in \Sigma^*$ and $a, b \in \Sigma$.

The shuffle operation is extended in the natural way to languages:

$$L_1 \text{ Ш } L_2 = \bigcup\nolimits_{u \in L_1, v \in L_2} (u \text{ Ш } v).$$

Let L_1 and L_2 be languages over Σ. The *scattered residual* of L_1 by L_2 is defined as:

$$L_1 \to_s L_2 = \bigcup_{u \in L_1, v \in L_2} (u \to_s v)$$

where

$$\begin{aligned} u \to_s v = \; & \{u_1 u_2 \ldots u_{k+1} \in \Sigma^* | \; k \geq 1, u = u_1 v_1 u_2 v_2 \ldots u_k v_k u_{k+1}, \\ & v = v_1 v_2 \ldots v_k, \; u_i \in \Sigma^*, 1 \leq i \leq k+1, v_i \in \Sigma^*, 1 \leq i \leq k\}. \end{aligned}$$

Thus, the scattered residual $u \to_s v$ indicates the result of deleting v from u in a scattered way. The following lemma provides an equivalent definition of the scattered residual. The straightforward proof is omitted.

Lemma 2.3. *For any languages L_1 and L_2*

$$L_1 \to_s L_2 = \{w \in \Sigma^* \mid \exists v \in L_2, (w \text{ Ш } v) \cap L_1 \neq \emptyset\}.$$

Moreover, for any words u, $v \in \Sigma^*$,

$$u \to_s v = \{w \in \Sigma^* \mid u \in w \text{ Ш } v\}. \qquad \square$$

A *deterministic finite automaton*, *DFA*, is a construct $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$, where Q is a finite set of *states*, Σ is an alphabet called the *input alphabet*, $q_0 \in Q$ is the *initial state*, $F \subseteq Q$ is the set of *final states* and

$$\delta : Q \times \Sigma \longrightarrow Q$$

is the *transition function.*

The transition function δ is extended to a function $\overline{\delta}$,

$$\overline{\delta} : Q \times \Sigma^* \longrightarrow Q,$$

by defining:

(i) $\overline{\delta}(q, \lambda) = q$, for all $q \in Q$,
(ii) $\overline{\delta}(q, wa) = \delta(\overline{\delta}(q, w), a)$, for all $w \in \Sigma^*$, $a \in \Sigma$, $q \in Q$.

In the sequel, $\overline{\delta}$ will be denoted simply by δ.

The *language accepted* by the DFA $\mathcal{A}$ is

$$L(\mathcal{A}) = \{w \in \Sigma^* \mid \delta(q_0, w) \in F\}.$$

Theorem 2.8. *Let L be a commutative language. The following conditions are equivalent:*

a)L is accepted by some DFA.
b)L has finitely many scattered residuals.

Proof. Let L be a commutative language over the alphabet Σ. Define the relation $\sim_L$ between languages over Σ as follows:

$$L_1 \sim_L L_2 \text{ if and only if } L \rightarrow_s L_1 = L \rightarrow_s L_2.$$

It follows immediately that the relation $\sim_L$ is an equivalence relation.

Denote by $\mathcal{P}(\Sigma^*)/_{\sim_L}$ the set of all equivalence classes with respect to the relation $\sim_L$. The equivalence class of a language K will be denoted by $[K]$.

a) $\Longrightarrow$ *b).* Let $\mathcal{A}= (Q, \Sigma, \delta, q_0, F)$ be a finite deterministic automaton such that $L(\mathcal{A}) = L$. We will assume that all states from Q are *accessible*, i.e. for each $p \in Q$ there is a word $u \in \Sigma^*$ such that $\delta(q_0, u) = p$. Moreover, without loss of generality, we may assume that the automaton $\mathcal{A}$ satisfies the following condition of minimality:

$$(*) \qquad \text{if } \{u \in \Sigma^* | \delta(p, u) \in F\} = \{v \in \Sigma^* | \delta(q, v) \in F\}, \text{ then } p = q.$$

Let X be the set $\mathcal{P}(Q)$ and consider the set of all functions defined on X and with values in X, denoted X^X.

Let φ be the function mapping the set $\mathcal{P}(\Sigma^*)/_{\sim_L}$ into the set X^X, defined by

$$\varphi([K])(Y) = \delta(Y, K).$$

Claim 1. The function φ is well defined.

Assume that K_1 and K_2 are languages over Σ such that $K_1 \sim_L K_2$. We prove that in this situation $\delta(Y, K_1) = \delta(Y, K_2)$ for all $Y \subseteq Q$. Let p be a state in $\delta(q, K_1)$ for some $q \in Y$. It follows that there is a word $u_1 \in K_1$, such that $\delta(q, u_1) = p$. Let α, β be words such that $\delta(q_0, \alpha) = q$ and $\delta(p, \beta) \in F$. Therefore, $\alpha u_1 \beta \in L$ and consequently, $\alpha\beta \in L \rightarrow_s K_1$. But, $K_1 \sim_L K_2$ and hence, $\alpha\beta \in L \rightarrow_s K_2$. Therefore, there exists $u_2 \in K_2$ such that $(\alpha\beta \text{ Ш } u_2) \cap L \neq \emptyset$. It follows that $\alpha u_2 \beta \in com(L) = L$. Moreover, note that for all $\alpha, \beta \in \Sigma^*$,

$$\alpha u_1 \beta \in L \text{ if and only if } \alpha u_2 \beta \in L.$$

Thus, from condition $(*)$, it follows that $\delta(q, u_2) = p \in \delta(Y, K_2)$. The converse inclusion is established analogously. Therefore Claim 1 is true.

Claim 2. φ is an injective function.

We will prove that if for all $Y \subseteq Q$, $\delta(Y, K_1) = \delta(Y, K_2)$, then $K_1 \sim_L K_2$.

Let u be in $L \rightarrow_s K_1$. There exists $v_1 \in K_1$ such that $(u \text{ Ш } v_1) \cap L \neq \emptyset$. Hence, $uv_1 \in com(L) = L$. Assume that $\delta(q_0, u) = p$ and observe that $\delta(p, v_1) \in F$ and $v_1 \in K_1$. From the equality $\delta(Y, K_1) = \delta(Y, K_2)$, for all $Y \subseteq Q$, it follows that there exists $v_2 \in K_2$ such that $\delta(p, v_2) \in F$. Hence, $uv_2 \in L$ and thus u is in $L \rightarrow_s K_2$. Therefore, $L \rightarrow_s K_1 \subseteq L \rightarrow_s K_2$ and note that the converse inclusion is analogous. Therefore φ is an injective function.

Now, observe that the set X^X is a finite set and hence the set $P(\Sigma^*)/_{\sim_L}$ is finite, too.

b) $\Longrightarrow$ a). Define the finite deterministic automaton $\mathcal{A}= (Q, \Sigma, \delta, q_0, F)$, where
$Q = \mathcal{P}(\Sigma^*)/_{\sim_L}$. Note that Q is a finite set. Define $q_0 = [\{\lambda\}]$ and $F = \{[K] | K \cap L \neq \emptyset\}$. Observe that from $[K] = [K']$ and $K \cap L \neq \emptyset$ it follows that $K' \cap L \neq \emptyset$. The transition function, $\delta : Q \times \Sigma \to Q$ is defined by

$$\delta([B], \sigma) = [B \text{ Ш } \sigma].$$

Observe that δ is well defined, i.e. if $B \sim_L B'$, then $B \text{ Ш } \sigma \sim_L B' \text{ Ш } \sigma$. In order to prove that $L(\mathcal{A}) = L$ assume first that $w \in L(\mathcal{A})$. Therefore, $\delta([\lambda], w) \in F$. If $w = w_1 w_2 \ldots w_n$, where $w_i \in \Sigma, i = 1, \ldots, n$, then it is easy to see that $\delta([\lambda], w) = [w_1 \text{ Ш } w_2 \text{ Ш } \ldots \text{ Ш } w_n]$. Hence, there is a language K, such that $K \cap L \neq \emptyset$ and $K \sim_L (w_1 \text{ Ш } \ldots \text{ Ш } w_n)$. Consequently, $(w_1 \text{ Ш } w_2 \text{ Ш } \ldots \text{ Ш } w_n) \cap L \neq \emptyset$. Therefore, $w = w_1 w_2 \ldots w_n \in com(L) = L$. Thus $w \in L$. Conversely, assume that $w \in L$. Now, if $w = w_1 w_2 \ldots w_n, w_i \in \Sigma, i = 1, \ldots, n$, then $\delta([\lambda], w) = [w_1 \text{ Ш } w_2 \text{ Ш } \ldots \text{ Ш } w_n]$ and thus $(w_1 \text{ Ш } w_2 \text{ Ш } \ldots \text{ Ш } w_n) \cap L \neq \emptyset$. Hence, $[w_1 \text{ Ш } w_2 \text{ Ш } \ldots \text{ Ш } w_n] \in F$ and therefore $w \in L(\mathcal{A})$. Consequently, $L = L(\mathcal{A})$. □

It will be seen in many connections in this Handbook that L is regular iff L is acceptable by a DFA. Thus, Theorem 2.7 yields,

Theorem 2.9. *Let L be a commutative language. The following conditions are equivalent:*

a) L is a regular language.
b) L has finitely many scattered residuals. □

3. Formal languages: a telegraphic survey

This section will be very different from the preceding ones. Following [7], we will present a "telegraphic survey" of some of the basics of formal language theory. This means that we just quickly run through very many concepts and results, without going into any details. It is recommended that the survey is consulted only when need arises.

3.1 Language and grammar. Chomsky hierarchy

Basic terminology and notation

Alphabet: a finite nonempty set of abstract *symbols* or *letters.*
V^*: the free monoid generated by the alphabet V under the operation of *catenation* or *concatenation* (the catenation of x and y is denoted by xy; the catenation of n copies of $x \in V^*$ is also denoted by x^n).

Word/string: an element of V^*.
λ: the empty string (the identity of the monoid V^*; $V^* - \{\lambda\}$ is denoted by V^+).
$|x|$: the *length* of the string $x \in V^*$.
$|x|_a$: the number of occurrences of $a \in V$ in $x \in V^*$.
Language: a subset of V^*.
λ-*free language:* a subset of V^+.
Parikh mapping: for $V = \{a_1, a_2, \ldots, a_k\}$, the mapping $\Psi_V : V^* \longrightarrow N^k$ defined by $\Psi_V(x) = (|x|_{a_1}, |x|_{a_2}, \ldots, |x|_{a_k})$.
Linear set: a set $M \subseteq N^k$ such that $M = \{v_0 + \sum_{i=1}^{m} v_i x_i \mid x_i \in N\}$, for some $v_0, v_1, \ldots, v_m$ in N^k.
Semilinear set: a finite union of linear sets.
Semilinear language: a language $L \subseteq V^*$ with semilinear $\Psi_V(L)$.
Letter-equivalent languages: two languages $L_1, L_2 \subseteq V^*$ such that $\Psi_V(L_1) = \Psi_V(L_2)$.
Length set (of a language L): *length*$(L) = \{|x| \mid x \in L\}$.

Operations with languages

Union, intersection, complement: usual set operations.
Concatenation: $L_1 L_2 = \{xy \mid x \in L_1, y \in L_2\}$.
Kleene star (*): $L^* = \cup_{i \geq 0} L^i$, where $L_0 = \{\lambda\}$, $L^{i+1} = L^i L, i \geq 0$.
Kleene +: $L^+ = \cup_{i \geq 1} L^i$.
Substitution: a mapping $s : V \longrightarrow 2^{U^*}$, V, U alphabets, extended to V^* by $s(\lambda) = \{\lambda\}$, $s(ax) = s(a)s(x)$, $a \in V, x \in V^*$. For $L \subseteq V^*, s(L) = \cup_{x \in L} s(x)$.
Finite substitution: substitution s with $s(a)$ finite for all symbols a.
Morphism: substitution s with $s(a)$ singleton set for all symbols a (usual monoid morphism).
λ-*free substitution/morphism:* substitution/morphism such that $\lambda \in s(a)$ for no symbol a.
Inverse morphism: a mapping $h^{-1} : U^* \longrightarrow 2^{V^*}$ defined by $h^{-1}(x) = \{y \in V^* \mid h(y) = x\}, x \in U^*$, for a morphism $h : V^* \longrightarrow U^*$.
Shuffle: $x ш y = \{x_1 y_1 \ldots x_n y_n \mid x = x_1 \ldots x_n, y = y_1 \ldots y_n, x_i, y_i \in V^*, 1 \leq i \leq n, n \geq 1\}$. For $L_1, L_2 \subseteq V^*$, $L_1 ш L_2 = \{w \in V^* \mid w \in x ш y$, for some $x \in L_1, y \in L_2\}$.
Mirror image: if $x = a_1 a_2 \ldots a_n, a_i \in V, 1 \leq i \leq n$, then $mi(x) = a_n \ldots a_2 a_1$. For $L \subseteq V^*$, $mi(L) = \{mi(x) \mid x \in L\}$.
Left quotient (of L_1 with respect to L_2, $L_1, L_2 \subseteq V^*$): $L_2 \backslash L_1 = \{w \mid$ there is $x \in L_2$ such that $xw \in L_1\}$.
Left derivative (of L with respect to $x, L \subseteq V^*, x \in V^*$): $\partial_x(L) = \{x\} \backslash L$.
Right quotient/derivative: symmetrically.
Given an n-ary operation with languages, α, we say that a family F of languages *is closed* under operation α if $\alpha(L_1, \ldots, L_n) \in F$ for every $L_1, \ldots, L_n \in F$.
A language $L \subseteq V^*$ is called *regular* if it can be obtained from elements of V

and $\emptyset$ using finitely many times the operations of union, concatenation and Kleene star.
A family of languages is *nontrivial* if it contains at least one language different from $\emptyset$ and $\{\lambda\}$. A nontrivial family of languages is called *trio* or a *cone* if it is closed under λ-free morphisms, inverse morphisms, and intersection with regular languages. A trio closed under union is called *semi-AFL* (AFL = abstract family of languages). A semi-AFL closed under concatenation and Kleene + is called *AFL*. A trio/semi-AFL/AFL is said to be *full* if it is closed under arbitrary morphisms (and Kleene $*$ in the case of AFL's). A family of languages closed under none of the six AFL operations is called *anti-AFL*.

1. The family of regular languages is the smallest full trio.
2. Each (full) semi-AFL closed under Kleene + is a (full) AFL.
3. If F is a family of λ-free languages which is closed under concatenation, λ-free morphisms, and inverse morphisms, then F is closed under intersection with regular languages and union, hence F is a semi-AFL. (If F is also closed under Kleene +, then it is an AFL.)
4. If F is a family of languages closed under intersection with regular languages, union with regular languages, and substitution with regular languages, then F is closed under inverse morphisms.
5. Every semi-AFL is closed under substitution with λ-free regular languages. Every full semi-AFL is closed under substitution with arbitrary regular languages and under left/right quotient with regular languages.

Chomsky grammars

A *phrase-structure grammar* is a quadruple $G = (N, T, S, P)$, where N, T are disjoint alphabets, $S \in N$, and $P \subseteq V_G^* N V_G^* \times V_G^*$, for $V_G = N \cup T$. The elements of N are called *nonterminal* symbols, those of T are called *terminal* symbols, S is the *start symbol* or the *axiom*, and P the set of *production rules*; $(u, v) \in P$ is written in the form $u \rightarrow v$.
For $x, y \in V_G^*$ we write $x \Longrightarrow_G y$ iff $x = x_1 u x_2, y = x_1 v x_2$, for some $x_1, x_2 \in V_G^*$ and $u \rightarrow v \in P$. If G is understood, we write $x \Longrightarrow y$. One says that x directly derives y (with respect to G). The reflexive and transitive closure of the relation $\Longrightarrow$ is denoted by $\Longrightarrow^*$. The *language generated* by G is $L(G) = \{x \in T^* \mid S \Longrightarrow^* x\}$.
Two grammars G_1, G_2 are called *equivalent* if $L(G_1) = L(G_2)$.
If in $x \Longrightarrow y$ above we have $x_1 \in T^*$, then the derivation is *leftmost* and we write $x \Longrightarrow_{left} y$. The leftmost language generated by G is denoted by $L_{left}(G)$.

A phrase-structure grammar $G = (N, T, S, P)$ is called:

– *monotonous/length-increasing*, if for all $u \rightarrow v \in P$ we have $|u| \leq |v|$.
– *context-sensitive*, if each $u \rightarrow v \in P$ has $u = u_1 A u_2, v = u_1 x u_2$, for $u_1, u_2 \in V_G^*, A \in N, x \in V_G^+$. (In monotonous and context-sensitive grammars a

production $S \rightarrow \lambda$ is allowed, providing S does not appear in the right-hand members of rules in P.)
- *context-free*, if each production $u \rightarrow v \in P$ has $u \in N$.
- *linear*, if each rule $u \rightarrow v \in P$ has $u \in N$ and $v \in T^* \cup T^*NT^*$.
- *right-linear*, if each rule $u \rightarrow v \in P$ has $u \in N$ and $v \in T^* \cup T^*N$.
- *left-linear*, if each rule $u \rightarrow v \in P$ has $u \in N$ and $v \in T^* \cup NT^*$.
- *regular*, if each rule $u \rightarrow v \in P$ has $u \in N$ and $v \in T \cup TN \cup \{\lambda\}$.

The phrase-structure, context-sensitive, context-free, and regular grammars are also called type 0, type 1, type 2, and type 3 grammars, respectively.

1. The family of languages generated by monotonous grammars is equal with the family of languages generated by context-sensitive grammars.
2. The family of languages generated by right- or by left-linear grammars are equal and equal with the family of languages generated by regular grammars, as defined in Section 3.2.

We denote by *RE, CS, CF, LIN, REG* the families of languages generated by arbitrary, context-sensitive, context-free, linear, and regular grammars, respectively (RE stands for *recursively enumerable*).

3. *(Chomsky hierarchy.)* The following strict inclusions hold: $REG \subset LIN \subset CF \subset CS \subset RE$.
4. $L_{left}(G) \in CF$ for each type-0 grammar G; if G is context-free, then $L_{left}(G) = L(G)$.

Closure properties of families in Chomsky hierarchy (Y stands for *yes* and N for *no*).

	RE	*CS*	*CF*	*LIN*	*REG*
Union	Y	Y	Y	Y	Y
Intersection	Y	Y	N	N	Y
Complement	N	Y	N	N	Y
Concatenation	Y	Y	Y	N	Y
Kleene $*$	Y	Y	Y	N	Y
Intersection with regular languages	Y	Y	Y	Y	Y
Substitution	Y	N	Y	N	Y
λ-free substitution	Y	Y	Y	N	Y
Morphisms	Y	N	Y	Y	Y
λ-free morphisms	Y	Y	Y	Y	Y
Inverse morphisms	Y	Y	Y	Y	Y
Left/right quotient	Y	N	N	N	Y
Left/right quotient with regular languages	Y	N	Y	Y	Y
Left/right derivative	Y	Y	Y	Y	Y
Shuffle	Y	Y	N	N	Y
Mirror image	Y	Y	Y	Y	Y

Therefore, *RE, CF, REG* are full AFL's, *CS* is an AFL (not full), and *LIN* is a full semi-AFL.

Decidability problems

Given a class G of grammars, the following basic questions about arbitrary elements G_1, G_2 of G can be formulated:

– *equivalence:* are G_1, G_2 equivalent?
– *inclusion:* is $L(G_1)$ included in $L(G_2)$?
– *membership:* is an arbitrarily given string x an element of $L(G_1)$?
– *emptiness:* is $L(G_1)$ empty?
– *finiteness:* is $L(G_1)$ finite?
– *regularity:* is $L(G_1)$ a regular language?

The languages with decidable membership question are called *recursive* and their family lies strictly intermediate between *CS* and *RE.*

Decidability properties of classes of phrase-structure, context-sensitive, context-free, linear, and regular grammars (on top of columns we have put the corresponding families of languages; N stands for *non-decidable* and D for *decidable.*

	RE	*CS*	*CF*	*LIN*	*REG*
Equivalence	N	N	N	N	D
Inclusion	N	N	N	N	D
Membership	N	D	D	D	D
Emptiness	N	N	D	D	D
Finiteness	N	N	D	D	D
Regularity	N	N	N	N	trivial

Rice theorem: Let P be a nontrivial property of type-0 languages (there are languages having property P and also languages not having property P). Then property P is undecidable for type-0 languages.

3.2 Regular and context-free languages

Characterizations of regular languages

1. For $L \subseteq V^*$, we define the equivalence relation $\sim_L$ over V^* by $x \sim_L y$ iff $(uxv \in L \Leftrightarrow uyv \in L)$ for all $u, v \in V^*$. Then $V^*/\sim_L$ is called the *syntactic monoid* of L.
2. The set *rex* of *regular expressions* (over an alphabet V) is the smallest set of strings containing $\emptyset$ and a, for every $a \in V$, and having the following property: if e_1, e_2 are in *rex*, then $(e_1e_2), (e_1 \cup e_2)$ and e_1^* are in *rex*, too. To a regular expression e we associate a language $L(e)$ according to the following rules:

(i) $L(\emptyset) = \emptyset, L(a) = \{a\}, a \in V$,
(ii) $L((e_1 e_2)) = L(e_1)L(e_2), L((e_1 \cup e_2)) = L(e_1) \cup L(e_2), L(e_1^*) = (L(e_1))^*$.

The following facts are basic for regular languages.
1. *Kleene theorem:* A language L is in the family REG iff there is a regular expression e such that $L = L(e)$ (hence it is regular in the sense of the definition in the previous section).
2. *Myhill-Nerode theorem:* A language $L \subseteq V^*$ is regular iff $V^*/\sim_L$ is finite.

Pumping lemmas and other necessary conditions

1. *Bar-Hillel (uvwxy, pumping) lemma for context-free languages:* If $L \in CF, L \subseteq V^*$, then there are $p, q \in N$ such that every $z \in L$ with $|z| > p$ can be written in the form $z = uvwxy$, with $u, v, w, x, y \in V^*$, $|vwx| \leq q, vx \neq \lambda$, and $uv^iwx^iy \in L$ for all $i \geq 0$.
2. *Pumping lemma for linear languages:* If $L \in LIN, L \subseteq V^*$, then there are $p, q \in N$ such that every $z \in L$ with $|z| > p$ can be written in the form $z = uvwxy$, with $u, v, w, x, y \in V^*$, $|uvxy| \leq q, vx \neq \lambda$, and $uv^iwx^iy \in L$ for all $i \geq 0$.
3. *Pumping lemma for regular languages:* If $L \in REG, L \subseteq V^*$, then there are $p, q \in N$ such that every $z \in L$ with $|z| > p$ can be written in the form $z = uvw$, with $u, v, w \in V^*$, $|uv| \leq q, v \neq \lambda$, and $uv^iw \in L$ for all $i \geq 0$.
4. *Ogden pumping lemma (pumping with marked positions):* If $L \in CF, L \subseteq V^*$, then there is $p \in N$ such that for every $z \in L$ and for every at least p marked occurrences of symbols in z, we can write $z = uvwxy$, where
   (i) either each of u, v, w or each of w, x, y contains at least a marked symbol,
   (ii) vwx contains at most p marked symbols,
   (iii) $uv^iwx^iy \in L$ for all $i \geq 0$.
5. *Parikh theorem:* Every context-free language is semilinear.
6. Every context-free language is letter-equivalent with a regular language.
7. *Consequences:* (i) Every context-free language over a one-letter alphabet is regular. (ii) The length set of an infinite context-free language contains an infinite arithmetical progression.
8. All the previous conditions are only necessary, not also sufficient.

Representation theorems

The language generated by the context-free grammar $G = (\{S\}, \{a_1, b_1, \ldots, a_n, b_n\}, S, \{S \to SS, S \to \lambda\} \cup \{S \to a_iSb_i \mid 1 \leq i \leq n\})$ is called *the Dyck language* (over n pairs of symbols that can be viewed as parentheses) and it is denoted by D_n.

1. Every regular language L can be written in the form

$$L = h_4(h_3^{-1}(h_2(h_1^{-1}(a^*b)))),$$

where h_1, h_2, h_3, h_4 are morphisms.

2. *Chomsky-Schützenberger theorem:* Every context-free language L can be written in the form $L = h(D_n \cap R)$, where h is a morphism, D_n is a Dyck language and R is a regular language.
3. Every language $L \in RE$ can be written in the form $L = h(L_1 \cap L_2)$, as well as in the form $L = L_3 \backslash L_4$, where h is a morphism and L_1, L_2, L_3, L_4 are linear languages.

Normal forms

1. *Chomsky normal form:* For every context-free grammar G, a grammar $G' = (N, T, S, P)$ can be effectively constructed, with the rules in P of the forms $A \rightarrow a$ and $A \rightarrow BC$, for $A, B, C \in N, a \in T$, such that $L(G') = L(G) - \{\lambda\}$.
2. *Greibach normal form:* For every context-free grammar G, a grammar $G' = (N, T, S, P)$ can be effectively constructed, with the rules in P of the form $A \rightarrow a\alpha$, for $A \in N, a \in T, \alpha \in (N \cup T)^*$, such that $L(G') = L(G) - \{\lambda\}$.
3. *The super normal form:* For every triple (k, l, m) of nonnegative integers and for every context-free grammar G, an equivalent grammar $G' = (N, T, S, P)$ can be effectively constructed containing rules of the following forms:
   (i) $A \rightarrow xByCz$, with $A, B, C \in N, x, y, z \in T^*$, and $|x| = k, |y| = l, |z| = m$,
   (ii) $A \rightarrow x$, with $A \in N, x \in T^*, |x| \in length(L(G))$.

Variants of Chomsky and Greibach normal forms can be obtained by particularizing the parameters k, l, m above.

Ambiguity

A context-free grammar G is *ambiguous* if there is $x \in L(G)$ having two different leftmost derivations in G. A context-free language L is *inherently ambiguous* if each context-free grammar of L is ambiguous; otherwise, L is called *unambiguous.*
1. There are inherently ambiguous linear languages; each regular language is unambiguous.
2. The ambiguity problem for context-free grammars is undecidable.

3.3 L Systems

1. An *interactionless Lindenmayer system* (0L system) is a triple $G = (V, P, w)$ where V is an alphabet, P is a finite set of rules $a \rightarrow u, a \in V, u \in V^*$, such that there is at least one rule for each $a \in V$, and $w \in V^*$. For $x, y \in V^*$ we write $x \Longrightarrow_G y$ iff $x = a_1 a_1 \dots a_k$, $y = u_1 u_2 \dots u_k$ and $a_i \rightarrow u_i \in P$, $1 \leq i \leq k$ (*parallel* derivation with each letter developing independently of its neighbours). The generated language is $L(G) = \{z \in V^* \mid w \Longrightarrow_G^* z\}$.

Examples of 0L systems.

System	Axiom	Rules	Language
G_1	a	$a \to a^2$	$\{a^{2^n} \mid n \geq 0\}$
G_2	a	$a \to a, a \to a^2$	a^+
G_3	a	$a \to b, b \to ab$	$\{u_i \mid u_0 = a, u_1 = b, u_{i+2} = u_i u_{i+1}\}$
G_4	aba	$a \to aba, b \to \lambda$	$\{(aba)^{2^n} \mid n \geq 0\}$
G_5	aba	$a \to a, b \to ba^2b$	$= L(G_4)$

2. *IL Systems* ("interactions") are defined similarly except that the rules are context-dependent: the same letter may be rewritten differently in different contexts.
3. Most capital letters have a standard meaning in connection with L systems. Thus, L refers to parallel rewriting. The character O (or 0) means that information between individual letters ("cells") is zero-sided. The letter P ("propagating") means that λ is never on the right side of a rule, and D ("deterministic") that, for each configuration (a letter in a context), there is only one rule. (The systems G_1, $G_3 - G_5$ are DOL systems and, apart from G_4, also $PDOL$ systems. They are also (degenerate cases of) IL systems. The systems G_4 and G_5 are equivalent.) The letter F ("finite") attached to the name of a system means that, instead of a single axiom, there can be finitely many axioms. The letter T ("tables") means that the rules are divided into subsets. In each derivation step, rules from the same subset have to be used; D in this context means that each subset is deterministic. Finally, E, H, C indicate that the language L originally generated is modified in a certain way. E ("extensions"): L is intersected with V_T^*, where V_T ("terminals") is a subset of the original alphabet V. H ("homomorphism"): a morphic image of L under a predefined morphism is taken. C ("coding"): a letter-to-letter morphic image of L is taken. Thus, we may speak of $EDT0L$ and $HPDF0L$ systems. The families of languages generated by such systems are denoted simply by *EDT0L* and *HPDF0L.*

Convention: two languages are considered equal if they differ only by the empty word.
1. *EIL = ETIL = RE; CF⊂ E0L⊂ EPIL = EPTIL = CS.*
2. *E0L=H0L=C0L=EP0L=HP0L=EF0L=HF0L=CF0L=EPF0L=*
=HPF0L; ET0L=HTOL=CT0L=EPT0L=HPT0L=ETF0L=HTF0L=
CTF0L=EPTF0L=HPTF0L⊃ CPT0L=CPTF0L.
3. *HD0L=HPD0L=HDF0L=HPDF0L=CDF0L=CPDF0L⊃ CD0L⊃*
⊃ED0L⊃D0L⊃PD0L; EDT0L=HDT0L=CDT0L.
4. Equivalence is *undecidable* for 0L systems but *decidable* for HD0L systems. Emptiness and infinity are *decidable* for ET0L languages.
5. In general, L families have *week closure properties*, apart from extended families, notably *E0L* and *ET0L*. For instance, *D0L* and *0L* are anti-AFL's but *E0L* is closed under union, catenation, morphisms, intersection with regular sets, Kleene + (but not under inverse morphisms), whereas *ET0L* is a full AFL.

Sequences

A D0L system generates its language in a *sequence*, the word lengths in this sequence constitute a *D0L length sequence*. The length sequences of G_1 and G_3 above are the nonnegative powers of 2 and the Fibonacci numbers, respectively. Quite much is known about $D0L$ length sequences, their theory being related to the theory of *finite difference equations* with constant coefficients. The length of the nth term can be expressed as an *exponential polynomial*, a finite sum of terms of the form $\alpha n \beta^n$, where α and β are complex numbers. Thus, neither logarithmic nor subexponential (not bounded by a polynomial) growth is possible. (Both types are possible for DIL systems.) *Decidable characterizations* can be given for $HD0L$ length sequences that are not $D0L$, and of strictly increasing $D0L$ length sequences that are not $PD0L$. It is *decidable* whether or not two given $HD0L$ systems generate the same length sequence, the same question being undecidable for DIL systems.

Comment: Parallelism of derivation in L systems reflects the original motivation: L systems were introduced to model the development of (simple filamentous) organisms. Although the L systems discussed above are 1-dimensional (generate only strings), *branching structures* can be described using various conventions in computer graphics.

3.4 More powerful grammars and grammar systems

Regulated rewriting

A *programmed grammar* is a system $G = (N, T, S, P)$, where N, T are disjoint alphabets, $S \in N$, and P is a finite set of quadruples $(b : A \to z, E, F)$, where $A \to z$ is a context-free rule over $N \cup T$, b is a label from a set Lab, and E, F are subsets of Lab. For $(x, b), (y, b')$ in $V_G^* \times Lab$ we write $(x, b) \Longrightarrow (y, b')$ iff one of the next two cases holds:

(i) $x = x_1 A x_2, y = x_1 z x_2$, for $(b : A \to z, E, F) \in P$ and $b' \in E$,
(ii) $|x|_A = 0, x = y$, and $b' \in F$.

The language generated by G is defined by $L(G) = \{w \in T^* \mid (S, b) \Longrightarrow^* (w, b')$, for some $b, b' \in Lab\}$.
We denote by PR_{ac}^{λ} the family of languages generated by programmed grammars; the superscript λ is removed when one uses only λ-free grammars; the subscript ac is removed when grammars with only empty sets F in rules $(b : A \to z, E, F)$ are used.

1. $CF \subset PR \subset PR_{ac} \subset CS$, $PR \subset PR^{\lambda} \subset PR_{ac}^{\lambda} = RE$.
2. $ET0L \subset PR_{ac}$.
3. PR_{ac} is an AFL (not full).
4. The programmed grammars (of various types) are equivalent with other classes of grammars with regulated rewriting: matrix, controlled, time-varying, random context, state, etc.

Grammar systems

A *cooperating distributed* (CD) *grammar system* of degree $n, n \geq 1$, is an $(n+3)$-tuple $G = (N, T, S, P_1, \ldots, P_n)$, where N, T are disjoint alphabets, $S \in N$, and $P_1, \ldots, P_n$ are finite sets of context-free rules over $N \cup T$. The derivation relation $x \Longrightarrow_{P_i} y$ is defined in the usual way. Moreover, $x \Longrightarrow^{t}_{P_i} y$ iff $x \Longrightarrow^{*}_{P_i} y$ and there is no $z \in (N \cup T)^*$ such that $y \Longrightarrow_{P_i} z$ (maximal derivation). Then we define $L_t(G) = \{w \in T^* \mid S \Longrightarrow^{t}_{P_{i_1}} w_1 \Longrightarrow^{t}_{P_{i_2}} \cdots \Longrightarrow^{t}_{P_{i_m}} w_m = w, m \geq 1, 1 \leq i_j \leq n, 1 \leq j \leq m\}$.
We denote by $CD_n(t)$ the family of languages generated (in the t-mode) by CD grammar systems of degree at most $n, n \geq 1$.

$$CF = CD_1(t) = CD_2(t) \subset CD_3(t) = CD_n(t) = ET0L, n \geq 3.$$

Contextual grammars

An *internal contextual grammar* with F choice (where F is a given family of languages) is a system $G = (V, B, (D_1, C_1), \ldots, (D_n, C_n)), n \geq 1$, where V is an alphabet, B is a finite language over V, D_i are languages in F, and C_i are finite subsets of $V^* \times V^*, 1 \leq i \leq n$. For $x, y \in V^*$ we write $x \Longrightarrow y$ iff $x = x_1x_2x_3, y = x_1ux_2vx_3$ for $x_2 \in D_i, (u, v) \in C_i$, for some $1 \leq i \leq n$. Then the language generated by G is $L(G) = \{w \in V^* \mid z \Longrightarrow^* w \text{ for some } z \in B\}$. We denote by *IC(F)* the family of languages generated by such grammars. We consider here $F \in \{FIN, REG\}$, where *FIN* is the family of finite languages.

1. $REG \subset IC(FIN) \subset IC(REG) \subset CS$.
2. *IC(FIN)* and *IC(REG)* are both incomparable with *LIN* and *CF*.
3. Every $L \in RE$ can be written in the form $L = R \backslash L'$, where $L' \in IC(FIN)$ and R is a regular language.

3.5 Books on formal languages

A. V. Aho, J. D. Ullman, *The Theory of Parsing, Translation, and Compiling*, Prentice Hall, Engl. Cliffs, N.J., vol. I 1971, vol. II 1973.

A. V. Aho, J. D. Ullman, *Principles of Compiler Design*, Addison-Wesley, Reading, Mass., 1977.

J. Berstel, *Transductions and Context-Free Languages*, Teubner, Stuttgart, 1979.

W. Brauer, *Automatentheorie*, B. G. Teubner, Stuttgart, 1984.

R. V. Book (ed.), *Formal Language Theory. Perspectives and Open Problems*, Academic Press, New York, 1980.

E. Csuhaj-Varju, J. Dassow, J. Kelemen, Gh. Păun, *Grammar Systems. A Grammatical Approach to Distribution and Cooperation*, Gordon and Breach, London, 1994.

J. Dassow, Gh. Păun, *Regulated Rewriting in Formal Language Theory*, Springer-Verlag, Berlin, 1989.

M. D. Davis, E. J. Weyuker, *Computability, Complexity, and Languages*, Academic Press, New York, 1983.

P. J. Denning, J. B. Dennis, J. E. Qualitz, *Machines, Languages, and Computation*, Prentice-Hall, Engl. Cliffs, N.J., 1978.

S. Eilenberg, *Automata, Languages, and Machines*, Academic Press, New York, vol. A, 1974, vol. B, 1976.

E. Engeler, *Formal Languages*, Markham, Chicago, 1968.

R. W. Floyd, R. Beigel, *The Language of Machines An Introduction to Computability and Formal Languages*, Computer Science Press, 1994.

F. Gécseg, I. Péak, *Algebraic Theory of Automata*, Akadémiai Kiadó, 1972.

S. Ginsburg, *The Mathematical Theory of Context-Free Languages*, McGraw Hill, New York, 1966.

S. Ginsburg, *Algebraic and Automata-Theoretic Properties of Formal Languages*, North-Holland, Amsterdam, 1975.

A. V. Gladkij, *Leçons de linguistique mathématique*, Centre de linguistique quantitative de la faculté des sciences de l'Université de Paris, Dunod, 1970.

M. Gross, A. Lentin, *Notions sur les grammaires formelles*, Gauthier-Villars, Paris, 1967.

M. Harrison, *Introduction to Formal Language Theory*, Addison-Wesley, Reading, Mass., 1978.

G. T. Herman, G. Rozenberg, *Developmental Systems and Languages*, North-Holland, Amsterdam, 1975.

J. E. Hopcroft, J. D. Ullman, *Formal Languages and Their Relations to Automata*, Addison-Wesley, Reading, Mass., 1969.

J. E. Hopcroft, J. D. Ullman, *Introduction to Automata Theory, Languages, and Computation*, Addison-Wesley, Reading, Mass., 1979.

M. Ito (ed.), *Words, Languages, and Combinatorics*, World Scientific, Singapore, 1992.

W. Kuich, A. Salomaa, *Semirings, Automata, Languages*, Springer-Verlag, Berlin, 1986.

H. R. Lewis, C. H. Papadimitriou, *Elements of the theory of computation*, Prentice-Hall, Engl. Cliffs, NJ, 1981.

P. Linz, *An Introduction to Formal Languages and Automata*, D. C. Heath and Co., Lexington, Mass., 1990. M. Lothaire, *Combinatorics on Words*, Addison-Wesley, Reading, Mass., 1983.

S. Marcus, *Grammars and Finite Automata*, The Publ. House of the Romanian Academy, Bucharest, Romania, 1964.

A. Mateescu, D. Vaida, *Discrete Mathematical Structures. Applications*, The Publ. House of the Romanian Academy, Bucharest, Romania, 1989.

H. Maurer, *Theoretische Grundlagen der Programmiersprachen*, Hochschultaschenbücher 404, Bibliographisches Inst., 1969.

Gh. Păun, *Contextual Grammars*, The Publ. House of the Romanian Academy, Bucharest, 1982.

Gh. Păun, *Recent Results and Problems in Formal Language Theory*, The Scientific and Encyclopaedic Publ. House, Bucharest, 1984.

Gh. Păun (ed.) *Mathematical Aspects of Natural and Formal Languages*, World Scientific, Singapore, 1994.

Gh. Păun (ed.) *Artificial Life: Grammatical Models*, Black Sea University Press, Bucharest, Romania, 1995.

Gh. Păun (ed.) *Mathematical linguistics and related topics*, The Publ. House of the Romanian Academy, Bucharest, Romania, 1995.

J. E. Pin, *Varieties of Formal Languages*, Plenum Press, Oxford, 1986.

G. E. Révész, *Introduction to Formal Languages*, McGraw-Hill, New York, 1980.

G. Rozenberg, A. Salomaa, *The Mathematical Theory of L Systems*, Academic Press, New York, 1980.

G. Rozenberg, A. Salomaa, *Cornerstones of Undecidability*, Prentice Hall, New York, 1994.

G. Rozenberg, A. Salomaa (eds.), *Developments in Language Theory*, World Scientific, Singapore, 1994.

A. Salomaa, *Theory of Automata*, Pergamon, Oxford, 1969.

A. Salomaa, *Formal Languages*, Academic Press, New York, London, 1973.

A. Salomaa, *Jewels of Formal Language Theory*, Computer Science Press, Rockville, 1981.

A. Salomaa, *Computation and Automata*, Cambridge Univ. Press, Cambridge, 1985.

A. Salomaa, M. Soittola, *Automata-Theoretic Aspects of Formal Power Series*, Springer-Verlag, Berlin, 1978.

S. Sippu, E. Soisalon-Soininen, *Parsing Theory. Vol. I: Languages and Parsing*, Springer-Verlag, Berlin, 1988.

H. J. Shyr, *Free Monoids and Languages*, Hon Min Book Comp., Taichung, 1991.

D. Wood, *Grammar and L Forms. An Introduction*, *Lecture Notes in Computer Science*, 91, Springer-Verlag, Berlin, 1980.

D. Wood, *Theory of computation*, Harper& Row, New York, 1987.

References

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

[2] T. V. Gamkrelidze, V. V. Ivanov, "The Early History of Indo-European Languages", *Scientific American*, March, (1990), 82–89.

[3] L. H. Haines, "On free monoids partially ordered by embedding", *J. Combinatorial Theory*, 6 (1969) 94–98.

[4] G. H. Higman, “Ordering by divisibility in abstract algebras”, *Proc. London Math. Soc.*, 3, (1952) 326–336.
[5] A. Lindenmayer, “Mathematical models for cellular interactions in development”, I, II, *J. Theoret. Biol.*, 18, (1968) 280–315.
[6] R. C. Lyndon, M. P. Schützenberger, “The equation $a^m = b^n c^p$ in a free group”, *Michigan Math. J.*, 9, (1962) 289–298.
[7] G. Păun, A. Salomaa, “Formal Languages”, in *CRC Handbook of Discrete and Combinatorial Mathematics*, K. H. Rosen (ed.), to appear.
[8] E. Post, “Finite combinatory processes-formulation”, I, *J. Symbolic Logic*, 1, (1936) 103–105.
[9] P. E. Ross, “Hard Words”, *Scientific American*, April (1991) 70–79.
[10] A. Salomaa, *Jewels of Formal Language Theory*, Computer Science Press, Rockville, 1981.
[11] A. Thue, “Über unendliche Zeichenreihen”, *Norske Vid. Selsk. Skr., I Mat. Nat. Kl.*, Kristiania, 7, (1906) 1–22.
[12] A. Thue, “Über die gegenseitige Lage gleicher Teile gewisser Zeichenreihen”, *Norske Vid. Selsk. Skr., I Mat. Nat. Kl.*, Kristiania, 1, (1912) 1–67.
[13] A. M. Turing, “On computable numbers with an application to the Entscheidungsproblem”, *Proc. London Math. Soc.*, 2, 42, (1936) 230–265. A correction, *ibid.*, 43, 544–546.

Preface

The need for a comprehensive survey-type exposition on formal languages and related mainstream areas of computer science has been evident for some years. In the early 1970s, when the book *Formal Languages* by the second-mentioned editor appeared, it was still quite feasible to write a comprehensive book with that title and include also topics of current research interest. This would not be possible anymore. A standard-sized book on formal languages would either have to stay on a fairly low level or else be specialized and restricted to some narrow sector of the field.

The setup becomes drastically different in a collection of contributions, where the best authorities in the world join forces, each of them concentrating on their own areas of specialization. The present three-volume Handbook constitutes such a unique collection. In these three volumes we present the current state of the art in formal language theory. We were most satisfied with the enthusiastic response given to our request for contributions by specialists representing various subfields. The need for a Handbook of Formal Languages was in many answers expressed in different ways: as an easily accessible historical reference, a general source of information, an overall course-aid, and a compact collection of material for self-study. We are convinced that the final result will satisfy such various needs.

The theory of formal languages constitutes the stem or backbone of the field of science now generally known as theoretical computer science. In a very true sense its role has been the same as that of philosophy with respect to science in general: it has nourished and often initiated a number of more specialized fields. In this sense formal language theory has been the origin of many other fields. However, the historical development can be viewed also from a different angle. The origins of formal language theory, as we know it today, come from different parts of human knowledge. This also explains the wide and diverse applicability of the theory. Let us have a brief look at some of these origins. The topic is discussed in more detail in the introductory Chapter 1 of Volume 1.

The main source of the theory of formal languages, most clearly visible in Volume 1 of this Handbook, is *mathematics*. Particular areas of mathematics important in this respect are combinatorics and the algebra of semigroups and monoids. An outstanding pioneer in this line of research was

Axel Thue. Already in 1906 he published a paper about avoidable and unavoidable patterns in long and infinite words. Thue and Emil Post were the two originators of the formal notion of a rewriting system or a grammar. That their work remained largely unknown for decades was due to the difficult accessibility of their writings and, perhaps much more importantly, to the fact that the time was not yet ripe for mathematical ideas, where noncommutativity played an essential role in an otherwise very simple setup.

Mathematical origins of formal language theory come also from mathematical logic and, according to the present terminology, computability theory. Here the work of Alan Turing in the mid-1930s is of crucial importance. The general idea is to find models of computing. The power of a specific model can be described by the complexity of the language it generates or accepts. Trends and aspects of mathematical language theory are the subject matter of each chapter in Volume 1 of the Handbook. Such trends and aspects are present also in many chapters in Volumes 2 and 3.

Returning to the origins of formal language theory, we observe next that much of formal language theory has originated from *linguistics*. In particular, this concerns the study of grammars and the grammatical structure of a language, initiated by Noam Chomsky in the 1950s. While the basic hierarchy of grammars is thoroughly covered in Volume 1, many aspects pertinent to linguistics are discussed later, notably in Volume 2.

The *modeling* of certain objects or phenomena has initiated large and significant parts of formal language theory. A model can be expressed by or identified with a language. Specific tasks of modeling have given rise to specific kinds of languages. A very typical example of this are the L systems introduced by Aristid Lindenmayer in the late 1960s, intended as models in developmental biology. This and other types of modeling situations, ranging from molecular genetics and semiotics to artificial intelligence and artificial life, are presented in this Handbook. Words are one-dimensional, therefore linearity is a feature present in most of formal language theory. However, sometimes a linear model is not sufficient. This means that the language used does not consist of words (strings) but rather of trees, graphs, or some other nonlinear objects. In this way the possibilities for modeling will be greatly increased. Such extensions of formal language theory are considered in Volume 3: languages are built from nonlinear objects rather than strings.

We have now already described the contents of the different volumes of this Handbook in brief terms. Volume 1 is devoted to the mathematical aspects of the theory, whereas applications are more directly present in the other two volumes, of which Volume 3 also goes into nonlinearity. The division of topics is also reflected in the titles of the volumes. However, the borderlines between the volumes are by no means strict. From many points of view, for instance, the first chapters of Volumes 2 and 3 could have been included in Volume 1.

We now come to a very important editorial decision we have made. Each of the 33 individual chapters constitutes its own entity, where the subject matter is developed from the beginning. References to other chapters are only occasional and comparable with references to other existing literature. This style of writing was suggested to the authors of the individual chapters by us from the very beginning. Such an editorial policy has both advantages and disadvantages as regards the final result. A person who reads through the whole Handbook has to get used to the fact that notation and terminology are by no means uniform in different chapters; the same term may have different meanings, and several terms may mean the same thing. Moreover, the prerequisites, especially in regard to mathematical maturity, vary from chapter to chapter. On the positive side, for a person interested in studying only a specific area, the material is all presented in a compact form in one place. Moreover, it might be counterproductive to try to change, even for the purposes of a handbook, the terminology and notation already well-established within the research community of a specific subarea. In this connection we also want to emphasize the diversity of many of the subareas of the field. An interested reader will find several chapters in this Handbook having almost totally disjoint reference lists, although each of them contains more than 100 references.

We noticed that guaranteed timeliness of the production of the Handbook gave additional impetus and motivation to the authors. As an illustration of the timeliness, we only mention that detailed accounts about DNA computing appear here in a handbook form, less than two years after the first ideas about DNA computing were published.

Having discussed the reasons behind our most important editorial decision, let us still go back to formal languages in general. Obviously there cannot be any doubt about the mathematical strength of the theory – many chapters in Volume 1 alone suffice to show the strength. The theory still abounds with challenging problems for an interested student or researcher. Mathematical strength is also a necessary condition for applicability, which in the case of formal language theory has proved to be both broad and diverse. Some details of this were already mentioned above. As the whole Handbook abounds with illustrations of various applications, it would serve no purpose to try to classify them here according to their importance or frequency. The reader is invited to study from the Handbook older applications of context-free and contextual grammars to linguistics, of parsing techniques to compiler construction, of combinatorics of words to information theory, or of morphisms to developmental biology. Among the newer application areas the reader may be interested in computer graphics (application of L systems, picture languages, weighted automata), construction and verification of concurrent and distributed systems (traces, omega-languages, grammar systems), molecular biology (splicing systems, theory of deletion), pattern matching, or cryptology, just to mention a few of the topics discussed in the Handbook.

About Volume 2

Some brief guidelines about the contents of the present Volume 2 follow. Problems about complexity occur everywhere in language theory; Chapter 1 gives an overall account. Parsing techniques are essential in applications, both for natural and programming languages. They are dealt with in Chapter 2, while Chapters 3–6 study extensions and variations of classical language theory. While Chapter 3 continues the general theory of context-free languages, Chapters 5 and 6 are motivated by linguistics, and Chapter 4, motivated by artificial intelligence, is also applicable to distributed systems. DNA computing has been an important recent breakthrough – some language-theoretic aspects are presented in Chapter 7. Chapter 8 considers the string editing problem which in various settings models a variety of problems arising from DNA and protein sequences. Chapter 9 considers several methods of word matching that are based on the use of automata. Chapter 10 discusses the relationship between automata theory and symbolic dynamics (the latter area has originated in topology). By its very nature the whole of cryptology can be viewed as a part of language theory. Chapter 11 gives an account of language-theoretic techniques that have turned out to be especially useful in cryptology.

Acknowledgements

We would like to express our deep gratitude to all the authors of the Handbook. Contrary to what is usual in case of collective works of this kind, we hoped to be able to keep the time schedule – and succeeded because of the marvellous cooperation of the authors. Still more importantly, thanks are due because the authors were really devoted to their task and were willing to sacrifice much time and energy in order to achieve the remarkable end result, often exceeding our already high expectations.

The Advisory Board consisting of J. Berstel, C. Calude, K. Culik II, J. Engelfriet, H. Jürgensen, J. Karhumäki, W. Kuich, M. Nivat, G. Păun, A. Restivo, W. Thomas, and D. Wood was of great help to us at the initial stages. Not only was their advice invaluable for the planning of the Handbook but we also appreciate their encouragement and support.

We would also like to extend our thanks from the actual authors of the Handbook to all members of the scientific community who have supported, advised, and helped us in many ways during the various stages of work. It would be a hopeless and maybe also unrewarding task to try to single out any list of names, since so many people have been involved in one way or another.

We are grateful also to Springer-Verlag, in particular Dr. Hans Wössner, Ingeborg Mayer, J. Andrew Ross, and Gabriele Fischer, for their cooperation, excellent in every respect. Last but not least, thanks are due to Marloes Boon-van der Nat for her assistance in all editorial stages, and in particular, for keeping up contacts to the authors.

September 1996 Grzegorz Rozenberg, Arto Salomaa

Preface

The need for a comprehensive survey-type exposition on formal languages and related mainstream areas of computer science has been evident for some years. In the early 1970s, when the book *Formal Languages* by the second-mentioned editor appeared, it was still quite feasible to write a comprehensive book with that title and include also topics of current research interest. This would not be possible anymore. A standard-sized book on formal languages would either have to stay on a fairly low level or else be specialized and restricted to some narrow sector of the field.

The setup becomes drastically different in a collection of contributions, where the best authorities in the world join forces, each of them concentrating on their own areas of specialization. The present three-volume Handbook constitutes such a unique collection. In these three volumes we present the current state of the art in formal language theory. We were most satisfied with the enthusiastic response given to our request for contributions by specialists representing various subfields. The need for a Handbook of Formal Languages was in many answers expressed in different ways: as an easily accessible historical reference, a general source of information, an overall course-aid, and a compact collection of material for self-study. We are convinced that the final result will satisfy such various needs.

The theory of formal languages constitutes the stem or backbone of the field of science now generally known as theoretical computer science. In a very true sense its role has been the same as that of philosophy with respect to science in general: it has nourished and often initiated a number of more specialized fields. In this sense formal language theory has been the origin of many other fields. However, the historical development can be viewed also from a different angle. The origins of formal language theory, as we know it today, come from different parts of human knowledge. This also explains the wide and diverse applicability of the theory. Let us have a brief look at some of these origins. The topic is discussed in more detail in the introductory Chapter 1 of Volume 1.

The main source of the theory of formal languages, most clearly visible in Volume 1 of this Handbook, is *mathematics*. Particular areas of mathematics important in this respect are combinatorics and the algebra of semigroups and monoids. An outstanding pioneer in this line of research was Axel Thue. Already in 1906 he published a paper about avoidable and unavoidable patterns in long and infinite words. Thue and Emil Post were the two originators of the formal notion of a rewriting system or a grammar. That their work remained largely unknown for decades was due to the difficult accessibility of their writings and, perhaps much more importantly, to the fact that the time was not yet ripe for mathematical ideas, where noncommutativity played an essential role in an otherwise very simple setup.

Mathematical origins of formal language theory come also from mathematical logic and, according to the present terminology, computability theory. Here the work of Alan Turing in the mid-1930s is of crucial importance. The general idea is to find models of computing. The power of a specific model can be described by the complexity of the language it generates or accepts. Trends and aspects of mathematical language theory are the subject matter of each chapter in Volume 1 of the Handbook. Such trends and aspects are present also in many chapters in Volumes 2 and 3.

Returning to the origins of formal language theory, we observe next that much of formal language theory has originated from *linguistics*. In particular, this concerns the study of grammars and the grammatical structure of a language, initiated by Noam Chomsky in the 1950s. While the basic hierarchy of grammars is thoroughly covered in Volume 1, many aspects pertinent to linguistics are discussed later, notably in Volume 2.

The *modeling* of certain objects or phenomena has initiated large and significant parts of formal language theory. A model can be expressed by or identified with a language. Specific tasks of modeling have given rise to specific kinds of languages. A very typical example of this are the L systems introduced by Aristid Lindenmayer in the late 1960s, intended as models in developmental biology. This and other types of modeling situations, ranging from molecular genetics and semiotics to artificial intelligence and artificial life, are presented in this Handbook. Words are one-dimensional, therefore linearity is a feature present in most of formal language theory. However, sometimes a linear model is not sufficient. This means that the language used does not consist of words (strings) but rather of trees, graphs, or some other nonlinear objects. In this way the possibilities for modeling will be greatly increased. Such extensions of formal language theory are considered in Volume 3: languages are built from nonlinear objects rather than strings.

We have now already described the contents of the different volumes of this Handbook in brief terms. Volume 1 is devoted to the mathematical aspects of the theory, whereas applications are more directly present in the other two volumes, of which Volume 3 also goes into nonlinearity. The division of topics is also reflected in the titles of the volumes. However, the borderlines between the volumes are by no means strict. From many points of view, for instance, the first chapters of Volumes 2 and 3 could have been included in Volume 1.

We now come to a very important editorial decision we have made. Each of the 33 individual chapters constitutes its own entity, where the subject matter is developed from the beginning. References to other chapters are only occasional and comparable with references to other existing literature. This style of writing was suggested to the authors of the individual chapters by us from the very beginning. Such an editorial policy has both advantages and disadvantages as regards the final result. A person who reads through the whole Handbook has to get used to the fact that notation and terminology are by no means uniform in different chapters; the same term may have different meanings, and several terms may mean the same thing. Moreover, the prerequisites, especially in regard to mathematical maturity, vary from chapter to chapter. On the positive side, for a person interested in studying only a specific area, the material is all presented in a compact form in one place. Moreover, it might be counterproductive to try to change, even for the purposes of a handbook, the terminology and notation already well-established within the research community of a specific subarea. In this connection we also want to emphasize the diversity of many of the subareas of the field. An interested reader will find several chapters in this Handbook having almost totally disjoint reference lists, although each of them contains more than 100 references.

We noticed that guaranteed timeliness of the production of the Handbook gave additional impetus and motivation to the authors. As an illustration of the timeliness, we only mention that detailed accounts about DNA computing appear here in a handbook form, less than two years after the first ideas about DNA computing were published.

Having discussed the reasons behind our most important editorial decision, let us still go back to formal languages in general. Obviously there cannot be any doubt about the mathematical strength of the theory – many chapters in Volume 1 alone suffice to show the strength. The theory still abounds with challenging problems for an interested student or researcher. Mathematical strength is also a necessary condition for applicability, which in the case of formal language theory has proved to be both broad and diverse. Some details of this were already mentioned above. As the whole Handbook abounds with illustrations of various applications, it would serve no purpose to try to classify them here according to their importance or frequency. The reader is invited to study from the Handbook older applications of context-free and contextual grammars to linguistics, of parsing techniques to compiler construction, of combinatorics of words to information theory, or of morphisms to developmental biology. Among the newer application areas the reader may be interested in computer graphics (application of L systems, picture languages, weighted automata), construction and verification of concurrent and distributed systems (traces, omega-languages, grammar systems), molecular biology (splicing systems, theory of deletion), pattern matching, or cryptology, just to mention a few of the topics discussed in the Handbook.

About Volume

Some brief guidelines about the contents of the present Volume 3 follow. Tree automata and tree languages were introduced already in the 1960s; Chapter 1 gives a comprehensive survey of this highly developed area. Chapter 2 deals with related issues, important in linguistics. Graph grammars generalize both string grammars and tree grammars and considerably extend possible application areas. Chapter 3 is a thorough survey of context-free graph grammars. Chapter 4 considers another generalization of string languages, namely two-dimensional languages where strings are replaced by rectangular arrays of symbols. Term rewriting plays an important role in the theory of programming languages – the basics of term rewriting are considered in Chapter 5. Considering infinite strings rather than finite ones is important for both theory and applications. Chapter 6 considers languages consisting of infinite strings (of order type ω), concentrating on the topological aspects of the theory of ω-languages. Chapter 7 considers formal languages in the framework of mathematical logic. The theory of traces based on partially commutative monoids forms a very successful framework for the study of concurrency; it is presented in Chapter 8.

The volume ends with beautiful applications of language theory to computer graphics. Chapter 9 applies L systems, and Chapter 10 automata. The diverse illustrations show the really many vistas and possibilities

Acknowledgements

We would like to express our deep gratitude to all the authors of the Handbook. Contrary to what is usual in case of collective works of this kind, we hoped to be able to keep the time schedule – and succeeded because of the marvellous cooperation of the authors. Still more importantly, thanks are due because the authors were really devoted to their task and were willing to sacrifice much time and energy in order to achieve the remarkable end result, often exceeding our already high expectations.

The Advisory Board consisting of J. Berstel, C. Calude, K. Culik II, J. Engelfriet, H. Jürgensen, J. Karhumäki, W. Kuich, M. Nivat, G. Păun, A. Restivo, W. Thomas, and D. Wood was of great help to us at the initial stages. Not only was their advice invaluable for the planning of the Handbook but we also appreciate their encouragement and support.

We would also like to extend our thanks from the actual authors of the Handbook to all members of the scientific community who have supported, advised, and helped us in many ways during the various stages of work. It would be a hopeless and maybe also unrewarding task to try to single out any list of names, since so many people have been involved in one way or another.

We are grateful also to Springer-Verlag, in particular Dr. Hans Wössner, Ingeborg Mayer, J. Andrew Ross, and Gabriele Fischer, for their cooperation, excellent in every respect. Last but not least, thanks are due to Marloes Boon-van der Nat for her assistance in all editorial stages, and in particular, for keeping up contacts to the authors.

September 1996 Grzegorz Rozenberg, Arto Salomaa