THE MATHEMATICAL THEORY OF L SYSTEMS

GRZEGORZ ROZENBERG
ARTO SALOMAA

ACADEMIC PRESS, INC.
111 Fifth Avenue, New York, New York 10003

United Kingdom Edition published by
ACADEMIC PRESS, INC. (LONDON) LTD.
24/28 Oval Road, London NW1 7DX

Library of Congress Cataloging in Publication Data

Rozenberg, Grzegorz.
The mathematical theory of L systems.

(Pure and applied mathematics, a series of monographs and textbooks ;)
Includes bibliographical references and indexes.
1. L systems. 2. Formal languages. I. Salomaa, Arto, joint author. II. Title. III. Series.
QA3.P8 [QA267.3] 510'.8s [511'.3] 79-25254
ISBN 0-12-597140-0

PRINTED IN THE UNITED STATES OF AMERICA

80 81 82 83 9 8 7 6 5 4 3 2 1

Preface

Formal language theory is by its very essence an interdisciplinary area of science: the need for a formal grammatical or machine description of specific languages arises in various scientific disciplines. Therefore, influences from outside the mathematical theory itself have often enriched the theory of formal languages.

Perhaps the most prominent example of such an outside stimulation is provided by the theory of L systems. L systems were originated by Aristid Lindenmayer in connection with biological considerations in 1968. Two main novel features brought about by the theory of L systems from its very beginning are (i) parallelism in the rewriting process—due originally to the fact that languages were applied to model biological development in which parts of the developing organism change simultaneously, and (ii) the notion of a grammar conceived as a description of a dynamic process (taking place in time), rather than a static one. The latter feature initiated an intensive study of sequences (in contrast to sets) of words, as well as of grammars without nonterminal letters. The results obtained in the very vigorous initial period—up to 1974—were covered in the monograph "Developmental Systems and Languages" by G. Herman and G. Rozenberg (North-Holland, 1975).

Since this initial period, research in the area of L systems has continued to be very active. Indeed, the theory of L systems constitutes today a considerable body of mathematical knowledge. The purpose of this monograph is to present in a systematic way the essentials of the mathematical theory of L systems. The material common to the present monograph and that of Herman and Rozenberg

quoted above consists only of a few basic notions and results. This is an indication of the dynamic growth in this research area, as well as of the fact that the present monograph focuses attention on systems without interactions, i.e., context-independent rewriting.

The organization of this book corresponds to the systematic and mathematically very natural structure behind L systems: the main part of the book (the first five chapters) deals with one or several iterated morphisms and one or several iterated finite substitutions. The last chapter, written in an overview style, gives a brief survey of the most important areas within L systems not directly falling within the basic framework discussed in detail in the first five chapters.

Today, L systems constitute a theory rich in original results and novel techniques, and yet expressible within a very basic mathematical framework. It has not only enriched the theory of formal languages but has also been able to put the latter theory in a totally new perspective. This is a point we especially hope to convince the reader of. It is our firm opinion that nowadays a formal language theory course that does not present L systems misses some of the very essential points in the area. Indeed, a course in formal language theory can be based on the mathematical framework presented in this book because the traditional areas of the theory, such as context-free languages, have their natural counterparts within this framework. On the other hand, there is no way of presenting iterated morphisms or parallel rewriting in a natural way within the framework of sequential rewriting.

No previous knowledge of the subject is required on the part of the reader, and the book is largely self-contained. However, familiarity with the basics of automata and formal language theory will be helpful. The results needed from these areas will be summarized in the introduction. Our level of presentation corresponds to that of graduate or advanced undergraduate work.

Although the book is intended primarily for computer scientists and mathematicians, students and researchers in other areas applying formal language theory should find it useful. In particular, theoretical biologists should find it interesting because a number of the basic notions were originally based on ideas in developmental biology or can be interpreted in terms of developmental biology. However, more detailed discussion of the biological aspects lies outside the scope of this book. The interested reader will find some references in connection with the bibliographical remarks in this book.

The discussion of the four areas within the basic framework studied in this book (single or several iterated morphisms or finite substitutions) builds up the theory starting from the simple and proceeding to more complicated objects. However, the material is organized in such a way that each of the four areas can also be studied independently of the others, with the possible exception of a few results needed in some proofs. In particular, a mathematically minded reader might find the study of single iterated morphisms (Chapters I and III) a very

interesting topic in its own right. It is an area where very intriguing and mathematically significant problems can be stated briefly *ab ovo*.

Exercises form an important part of the book. Many of them survey topics not included in the text itself. Because some exercises are rather difficult, the reader may wish to consult the reference works cited. Many open research problems are also mentioned throughout the text. Finally, the book contains references to the existing literature both at the end and scattered elsewhere. These references are intended to aid the reader rather than to credit each result to some specific author(s).

Acknowledgments

It is difficult to list all persons who have in some way or other contributed to this book. We have (or at least one of us has) benefited from discussions with or comments from K. Culik II, J. Engelfriet, T. Harju, J. Karhumäki, K. P. Lee, R. Leipälä, A. Lindenmayer, M. Linna, H. Maurer, M. Penttonen, K. Ruohonen, M. Soittola, A. Szilard, D. Vermeir, R. Verraedt, P. Vitanyi, and D. Wood. The difficult task of typing the manuscript was performed step by step and in an excellent fashion by Ludwig Callaerts and Leena Leppänen. T. Harju and J. Mäenpää were of great help with the proofs. We want to thank Academic Press for excellent and timely editorial work with both the manuscript and proofs.

A. Salomaa wants to express his gratitude to the Academy of Finland for good working conditions during the writing of this book. G. Rozenberg wants to thank UIA in Antwerp for the same reason and, moreover, wants to express his deep gratitude to his friend and teacher Andrzej Ehrenfeucht, not only for outstanding contributions to the theory of L systems but especially for continuing guidance and encouragement in scientific research.

List of Symbols

Contents

Introduction

This introduction gives a summary of the background material from automata and formal language theory needed in this book. It is suggested that the introduction be consulted only when need arises and that actual reading begin with Chapter I.

An *alphabet* is a set of abstract symbols. Unless stated otherwise, the alphabets considered in this book are always finite nonempty sets. The elements of an alphabet Σ are called *letters* or *symbols*. A *word* over an alphabet Σ is a finite string consisting of zero or more letters of Σ, whereby the same letter may occur several times. The string consisting of zero letters is called the *empty word*, written Λ. The set of all words (resp. all nonempty words) over an alphabet Σ is denoted by Σ^* (resp. Σ^+). Thus, algebraically, Σ^* and Σ^+ are the free monoid and free semigroup generated by Σ.

For words w_1 and w_2, the juxtaposition w_1w_2 is called the *catenation* (or concatenation) of w_1 and w_2. The empty word Λ is an identity with respect to catenation. Catenation being associative, the notation w^i, where i is a nonnegative integer, is used in the customary sense, and w^0 denotes the empty word. The *length* of a word w, in symbols $|w|$, means the number of letters in w when each letter is counted as many times as it occurs. A word w is a *subword* of a word u if there are words w_1 and w_2 such that $u = w_1ww_2$. If, in addition, $w \neq u$ and $w \neq \Lambda$, then w is termed a *proper* subword of u.

Furthermore, if $w_1 = \Lambda$ (resp. $w_2 = \Lambda$), then w is called an *initial* subword or *prefix* of u (resp. a *final* subword or a *suffix* of u).

Subsets of Σ^* are referred to as *languages* over Σ. Thus, if L is a language over Σ, it is also a language over Σ_1, provided $\Sigma \subseteq \Sigma_1$. However, when we speak of the alphabet of a language L, in symbols $alph(L)$, then we mean the smallest alphabet Σ such that L is a language over Σ. If L consists of a single word w, i.e., $L = \{w\}$, then we write simply $alph(w)$ or $alph\ w$ instead of $alph(\{w\})$. (In general, we do not make any distinction between elements x and singleton sets $\{x\}$.)

Various unary and binary *operations* for languages will be considered in the sequel. Regarding languages as sets, we may immediately define the Boolean operations of union, intersection, complementation (here it is essential that $alph(L)$ is considered) and difference in the usual fashion. The *catenation* (or *product*) of two languages L_1 and L_2 is defined by

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

The notation L^i is extended to apply to the catenation of languages. By definition, $L^0 = \{\Lambda\}$. The *catenation closure* or *Kleene star* (resp. Λ-free catenation closure, or Kleene *plus* or *cross*) of a language L, in symbols L^* (resp. L^+) is defined to be the union of all nonnegative (resp. positive) powers of L.

We now define the operation of *substitution*. For each letter a of an alphabet Σ, let $\sigma(a)$ be a language (possibly over a different alphabet). Define, furthermore,

$$\sigma(\Lambda) = \{\Lambda\}, \qquad \sigma(w_1w_2) = \sigma(w_1)\sigma(w_2),$$

for all w_1 and w_2 in Σ^*. For a language L over Σ, we define

$$\sigma(L) = \{u \mid u \in \sigma(w) \text{ for some } w \in L\}.$$

Such a mapping σ is called a *substitution*. Depending on the languages, $\sigma(a)$, where a ranges over Σ, we obtain substitutions of more restricted types. In particular, if each of the languages $\sigma(a)$ is finite, we call σ a *finite substitution*. If none of the languages $\sigma(a)$ contains the empty word, we call σ a Λ-*free* or *nonerasing* substitution.

A substitution σ such that each $\sigma(a)$ consists of a single word is called a *homomorphism* or, briefly, a *morphism*. If each $\sigma(a)$ is a word over Σ, we call σ also an *endomorphism*. (Algebraically, a homomorphism of languages is a monoid morphism linearly extended to subsets of monoids.) According to the convention above (identifying elements and their singleton sets), we write $\sigma(a) = w$ rather than $\sigma(a) = \{w\}$. A homomorphism σ is Λ-*free* or *nonerasing* if $\sigma(a) \neq \Lambda$ for every a. A letter-to-letter homomorphism will often in the sequel be called a *coding*. *Inverse homomorphisms* are inverses of homomor-

phisms, regarded as mappings. They will be explicitly defined below, in connection with transductions.

The *mirror image* of a word w, in symbols $\mathrm{mir}(w)$, is the word obtained by writing w backward. The mirror image of a language is the collection of the mirror images of its words, that is,

$$\mathrm{mir}(L) = \{\mathrm{mir}(w) | w \text{ in } L\}.$$

The cardinality of a finite set S is denoted by $\#(S)$. Similarly, $\#_\Sigma(w)$ or $\#_\Sigma w$ denotes the number of occurrences of letters from Σ in the word w. If Σ consists of one letter a_1, this notation reads $\#_{a_1} w$, meaning the number of occurrences of a_1 in w. If the alphabet considered is $\{a_1, \ldots, a_n\}$, we write simply $\#_{a_i} w = \#_i w$. The notation $pres_\Sigma w$ means the word obtained from w by erasing all letters not in Σ. (Thus, only letters "present" in Σ are considered.) Clearly,

$$\#_\Sigma w = |pres_\Sigma w|.$$

If w is a word and $1 \leq i \leq |w|$, then $w(i)$ denotes the ith letter of w.

The main objects of study in formal language theory are finitary specifications of infinite languages. Most such specifications are obtained as special cases from the notion of a rewriting system. By definition, a *rewriting system* is an ordered pair (Σ, P), where Σ is an alphabet and P a finite set of ordered pairs of words over Σ. The elements (w, u) of P are referred to as *rewriting rules* or *productions* and usually denoted $w \to u$. Given a rewriting system, the (binary) *yield relation* $\Rightarrow$ on the set Σ^* is defined as follows. For any words α and β, $\alpha \Rightarrow \beta$ holds if and only if there are words x_1, x_2, w, u such that

$$\alpha = x_1 w x_2 \qquad \text{and} \qquad \beta = x_1 u x_2,$$

and $w \to u$ is a production in the system. The reflexive transitive (resp. transitive) closure of the relation $\Rightarrow$ is denoted $\Rightarrow^*$ (resp. $\Rightarrow^+$). If several rewriting systems $G, H, \ldots$ are considered simultaneously, we write $\underset{G}{\Rightarrow}$ to avoid confusion when dealing with G.

A *phrase structure grammar* or, briefly, *grammar* is an ordered quadruple $G = (\Sigma, P, S, \Delta)$, where Σ and Δ are alphabets and $\Delta \subsetneq \Sigma$ (Δ is called the alphabet of *terminals* and $\Sigma \backslash \Delta$ the alphabet of *nonterminals*), S is in $\Sigma \backslash \Delta$ (the *initial* letter), and P is a finite set of ordered pairs (w, u), where w and u are words over Σ and w contains at least one nonterminal letter. Again, the elements of P are referred to as rewriting rules or productions and written $w \to u$. A grammar G as above defines a rewriting system (Σ, P). Let $\Rightarrow$ and $\Rightarrow^*$ be the relations determined by this rewriting system. Then the language $L(G)$ *generated* by G is defined by

$$L(G) = \{w \in \Delta^* | S \Rightarrow^* w\}.$$

For $i = 0, 1, 2, 3$, a grammar $G = (\Sigma, P, S, \Delta)$ is of *type i* if the restrictions (i) on P, as given below, are satisfied:

(0) No restrictions.
(1) Each producion in P is of the form $w_1 A w_2 \rightarrow w_1 w w_2$, where w_1 and w_2 are arbitrary words, A is a nonterminal letter, and w is a nonempty word (with the possible exception of the production $S \rightarrow \Lambda$ whose occurrence in P implies, however, that S does not occur on the right-hand side of any production).
(2) Each production in P is of the form $A \rightarrow w$, where A is a nonterminal letter and w is an arbitrary word.
(3) Each production in P is of one of the two forms $A \rightarrow Bw$ or $A \rightarrow w$, where A and B are nonterminal letters and w is an arbitrary word over the terminal alphabet Δ.

A language is of type i if and only if it is generated by a grammar of type i. Type 0 languages are also called *recursively enumerable*. Type 1 grammars and languages are also called *context-sensitive*. Type 2 grammars and languages are also called *context-free*. Type 3 grammars and languages are also referred to as *regular*. The four language families thus defined are denoted by $\mathscr{L}$(RE), $\mathscr{L}$(CS), $\mathscr{L}$(CF), $\mathscr{L}$(REG). Furthermore, the family of all finite languages is denoted by $\mathscr{L}$(FIN). These families form a strictly increasing hierarchy, usually referred to as the *Chomsky hierarchy:*

$$\mathscr{L}(\text{FIN}) \subsetneq \mathscr{L}(\text{REG}) \subsetneq \mathscr{L}(\text{CF}) \subsetneq \mathscr{L}(\text{CS}) \subsetneq \mathscr{L}(\text{RE}).$$

(The reader is referred to [S4] for a more-detailed discussion, as well as for all proofs of the facts listed in this introduction.)

Two grammars G and G_1 are termed *equivalent* if $L(G) = L(G_1)$. This notion of equivalence is extended to apply to all devices defining languages; two devices are *equivalent* if they define the same language. To avoid awkward special cases we make the *convention* that two languages differing by at most the empty word Λ are considered to be equal. We also make the convention that whenever new letters are introduced in a construction they are distinct from the letters introduced previously.

For a grammar G, every word w such that $S \Rightarrow^* w$ is referred to as a *sentential form* of G. Hence, a sentential form need not be over the terminal alphabet Δ. A context-free grammar is termed *linear* if the right-hand side of every production contains at most one nonterminal. A language is *linear* if it is generated by a linear grammar.

The *length set* of a language L is defined by

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

Consider the alphabet $\Sigma = \{a_1, \ldots, a_n\}$. The mapping ψ of Σ^* into the set N^n of ordered n-tuples of nonnegative integers defined by

$$\psi(w) = (\#_1(w), \ldots, \#_n(w))$$

is termed the *Parikh mapping* and its values *Parikh vectors*. The *Parikh set* of a language L over Σ is defined by

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

A subset K of N^n is said to be *linear* if there are finitely many elements $c, b_1, \ldots, b_r$ of N^n such that

$$K = \left\{ c + \sum_{i=1}^{r} m_i b_i \,\middle|\, m_i \text{ a nonnegative integer}, i = 1, \ldots, r \right\}.$$

A subset of N^n is said to be *semilinear* if it is a finite union of linear sets.

The Parikh set of a context-free language is always semilinear. Consequently, the length set of a context-free language, ordered according to increasing length, constitutes an almost periodic sequence. We often want to exclude the "initial mess" from the language we are considering: if L is a language and r a positive integer, we denote by $less_r(L)$ the subset of L consisting of words of length less than r.

The family of regular languages over an alphabet Σ equals the family of languages obtained from "atomic" languages $\{\Lambda\}$ and $\{a\}$, where $a \in \Sigma$, by a finite number of applications of *regular operations*: union, catenation, and catenation closure. The formula expressing how a specific regular language is obtained from atomic languages by regular operations is termed a *regular expression*.

The families of type i languages, $i = 0, 1, 2, 3$, defined above using generative devices can be obtained also by recognition devices or *automata*. A recognition device defining a language L receives arbitrary words as inputs and "accepts" exactly the words belonging to L. We now define in detail the class of automata-accepting regular languages.

A rewriting system (Σ, P) is called a *finite deterministic automaton* if (i) Σ is divided into two disjoint alphabets Q and V (the *state* and the *input* alphabet), (ii) an element $q_0 \in Q$ and a subset $F \subseteq Q$ are specified (*initial state* and *final state set*), and (iii) the productions in P are of the form

$$q_i a_k \to q_j, \quad q_i, q_j \in Q, \quad a_k \in V,$$

and, for each pair (q_i, a_k), there is exactly one such production in P.

The language *accepted* or recognized by a finite deterministic automaton FDA is defined by

$$L(\text{FDA}) = \{w \in V^* \mid q_0 w \Rightarrow^* q_1 \text{ for some } q_1 \in F\}.$$

A finite deterministic automaton is usually defined by specifying a quintuple (V, Q, f, q_0, F), where f is a mapping of $Q \times V$ into Q, the other items being as above. (Clearly, the values of f are obtained from the right-hand sides of the productions $q_i a_k \to q_j$.)

A finite *nondeterministic* automaton FNA is defined as a deterministic one with the following two exceptions. In (ii) q_0 is replaced by a subset $Q_0 \subseteq Q$. In (iii) the second sentence ("and for each pair...") is omitted. The language accepted by an FNA is defined by

$$L(\text{FNA}) = \{w \in V^* \mid q_0 w \Rightarrow^* q_1 \text{ for some } q_0 \in Q_0 \text{ and } q_1 \in F\}.$$

A language is regular if and only if it is accepted by some finite deterministic automaton if and only if it is accepted by some finite nondeterministic automaton.

We omit the detailed definition of the three classes of automata (pushdown automata, linearly bounded automata, Turing machines) corresponding to the language families $\mathscr{L}(\text{CF})$, $\mathscr{L}(\text{CS})$, $\mathscr{L}(\text{RE})$. (The reader is referred to [S4].) In particular, a Turing machine is the most general type of an automaton: it is considered to be the formal counterpart of the informal intuitive notion of an "effective procedure." (Hence, this applies also to type 0 grammars because they have the same language-accepting capability as Turing machines.) The addition of new capabilities to a Turing machine does not increase the computing power of this class of automata. In particular—as in connection with finite automata—deterministic and nondeterministic Turing machines accept the same class of languages. As regards pushdown automata, deterministic automata accept a strictly smaller class of languages than nondeterministic ones: $\mathscr{L}(\text{CF})$ is accepted by the nondeterministic ones. As regards linear bounded automata, the relation between deterministic and nondeterministic ones constitutes a very famous open problem, often referred to as the LBA problem.

Acceptors have no other output facilities than being or not being in a final state after the computation, i.e., they are capable only of accepting or rejecting inputs. Sometimes devices (transducers) capable of having words as outputs, i.e., capable of translating words into words, are considered. We give next the formal definition for the transducer corresponding to a finite automaton. In particular, its simplified version (gsm) will be needed in this book quite often.

A rewriting system (Σ, P) is called a *sequential transducer* if each of the following conditions (i)–(iii) is satisfied:

(i) Σ is divided into two disjoint alphabets Q and $V_{\text{in}} \cup V_{\text{out}}$. (The sets Q, V_{in}, V_{out} are called the *state*, *input*, and *output* alphabet, respectively. The latter two are nonempty but not necessarily disjoint.)

(ii) An element $q_0 \in Q$ and a subset $F \subseteq Q$ are specified (*initial state* and *final state set*).

(iii) The productions in P are of the form

$$q_i w \to u q_j, \qquad q_i, q_j \in Q, \quad w \in V_{\text{in}}^*, \quad u \in V_{\text{out}}^*.$$

If, in addition, $w \neq \Lambda$ in all productions, then the rewriting system is called a *generalized sequential machine* (gsm). If, in addition, always $u \neq \Lambda$, we speak of a Λ-free gsm.

For a sequential transducer ST, words $w_1 \in V_{\text{in}}^*$ and $w_2 \in V_{\text{out}}^*$, and languages $L_1 \subseteq V_{\text{in}}^*$ and $L_2 \subseteq V_{\text{out}}^*$, we define

$$\begin{aligned}
\text{ST}(w_1) &= \{w \mid q_0 w_1 \Rightarrow^* w q_1 \text{ for some } q_1 \in F\}, \\
\text{ST}(L_1) &= \{u \mid u \in \text{ST}(w) \text{ for some } w \in L_1\}, \\
\text{ST}^{-1}(w_2) &= \{u \mid w_2 \in \text{ST}(u)\}, \\
\text{ST}^{-1}(L_2) &= \{u \mid u \in \text{ST}^{-1}(w) \text{ for some } w \in L_2\}.
\end{aligned}$$

Mappings of languages thus defined are referred to as (*rational*) *transductions* and *inverse* (*rational*) *transductions*. If ST is also a gsm, we speak of *gsm mappings* and *inverse gsm mappings*. In what follows, a generalized sequential machine is usually defined by specifying a sixtuple $(V_{\text{in}}, V_{\text{out}}, Q, f, q_0, F)$, where f is a finite subset of the product set $Q \times V_{\text{in}}^+ \times V_{\text{out}}^* \times Q$, the other items being as above.

A homomorphism, an inverse homomorphism, and a mapping $f(L) = L \cap R$, where R is a fixed regular language, are all rational transductions, the first and the last being also gsm mappings. The composition of two rational transductions (resp. gsm mappings) is again a rational transduction (resp. gsm mapping). Every rational transduction f can be expressed in the form

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

where h_1 and h_2 are homomorphisms and R is a regular language.

These results show that a language family is closed under rational transductions if and only if it is closed under homomorphisms, inverse homomorphisms, and intersections with regular languages. Such a language family is referred to as a *cone*. A cone closed under regular operations is termed a *full AFL*. (If only nonerasing homomorphisms are considered in the homomorphism closure, we speak of an *AFL*.) A family of languages is termed an *anti-AFL* if it is closed under none of the six operations involved (i.e., union, catenation, catenation closure, homomorphism, inverse homomorphism, intersection with regular languages).

Each of the families $\mathscr{L}$(REG), $\mathscr{L}$(CF), $\mathscr{L}$(CS), and $\mathscr{L}$(RE) is closed under the following operations: union, catenation, Kleene star, Kleene plus, intersection with a regular language, mirror image, Λ-free substitution, Λ-free homomorphism, Λ-free gsm mapping, Λ-free regular substitution, inverse homomorphism, inverse gsm mapping. With the exception of $\mathscr{L}$(CS), these