CHAPTER 2.
DEFINITIONS AND PRELIMINARIES

We shortly review the more needed formal language theoretical machinery and the most generally used concepts of L system theory. For formal language theory and automata theory we use notation and so forth from Hopcroft and Ullman [1969], and for L system theory we sometimes depart somewhat from the (nonuniform) notation in the literature, so as to obtain some unity in our treatment.

2.1. FORMAL GRAMMARS

Formal grammars originate from Chomsky [1957], who introduced them for largely linguistical reasons. In essence, a formal grammar is a string rewriting system which transforms strings into strings. The purpose is to define in a finite way (by means of the grammar) an infinite number of strings (the language), such that the particular definition of each string (the derivation) yields some structural information about it. For proofs of lemmas and theorems in this section consult any textbook on the subject, e.g., Hopcroft and Ullman [1969].

We denote generally, with or without indices, symbols (equivalently, letters) by $a, b, c, \ldots$; strings (or words) of letters by $u, v, w, x, y, z$ or $a, b, \gamma, \ldots, \omega$; sets of letters (alphabets) by $A, B, U, V, W$; sets of strings (languages) by $L, X, Y, Z$; numbers by $i, j, k, l, m, n, p, q, r, s, t$. We will not always strictly adhere to these conventions, but then the context will allay confusion. The set of natural numbers $\{0, 1, 2, \ldots\}$ is denoted by $\mathbb{N}$; the set of reals by $\mathbb{R}$; and the set of positive reals by $\mathbb{R}^+$. If $X$ is a set then $\#X$ denotes the cardinality of $X$; if $x$ is a string then $lg(x)$ or $|x|$ denotes the length (number of occurrences of letters) of $x$.

The set of all strings over some finite alphabet of letters $W$ is denoted by $W^*$, e.g., $W^* = \{\lambda, a, b, aa, ab, ba, bb, aaa, \ldots\}$ for $W = \{a, b\}$, and $\lambda$ denotes the empty string (the string consisting of no letters at all).
$W^*$ is customarily called the free monoid finitely generated by $W$. If $X$ and $Y$ are two sets then

$$X \cup Y = \{ x \mid x \in X \text{ or } x \in Y \},$$

$$X \cap Y = \{ x \mid x \in X \text{ and } x \in Y \},$$

$$X - Y = \{ x \mid x \in X \text{ and } x \notin Y \},$$

$$XY = \{ w \mid w = xy \text{ and } x \in X \text{ and } y \in Y \},$$

$$x^i = \begin{cases} \{ \lambda \} & \text{if } i = 0 \\ x x^{i-1} & \text{if } i > 0, \end{cases}$$

$$x^* = \bigcup_{i=0}^{\infty} x^i,$$

$$x^+ = \bigcup_{i=1}^{\infty} x^i,$$

and $\emptyset$ denotes the empty set. The class of regular sets over an alphabet $W$ is formed as follows:

(i) $\emptyset$ and the singleton sets of elements in $W$ are regular sets.

(ii) If $R_1$ and $R_2$ are regular sets then so are $R_1 \cup R_2$ and $R_1 R_2$.

(iii) If $R$ is a regular set then so is $R^*$.

(iv) Only sets formed by application of (i) - (iii) are regular sets.

**DEFINITION 2.1.** A generative or formal grammar is an ordered quadruple

$G = <V_N, V_T, S, P>$ where $V_N$ and $V_T$ are finite nonempty alphabets, $V_N \cap V_T = \emptyset$, $S \in V_N$, and $P$ is a finite set of ordered pairs $(\alpha, \beta)$ such that $\beta$ is a word over the alphabet $V = V_N \cup V_T$ and $\alpha$ is a word over $V$ containing at least one letter of $V_N$. The elements of $V_N$ are called nonterminals and those of $V_T$ terminals; $S$ is called the start symbol. Elements $(\alpha, \beta)$ of $P$ are called rewriting rules or productions and are written $\alpha \Rightarrow \beta$.

$P$ induces a relation "⇒" on $V^*$ as follows. $v'$ is directly produced from $v$: $v \Rightarrow v'$ if there are $\gamma_1, \gamma_2, \alpha, \beta \in V^*$ such that $v' = \gamma_1 \beta \gamma_2$, $v = \gamma_1 \alpha \gamma_2$, and there is a $\alpha' \beta' \in P$. The transitive reflexive closure of $\Rightarrow$ is $\Rightarrow^*$ and the transitive irreducible closure of $\Rightarrow$ is $\Rightarrow^+$.

If $v \Rightarrow^* v'$ or $v \Rightarrow^+ v'$ we say $v$ produces or derives $v'$. If $v_0 \Rightarrow^*_1 \cdots \Rightarrow^*_n v$, we write $v_0 \Rightarrow^n v$, and say $v_0$ produces $v$ in $n$ steps. The language produced by $G$ is defined by

$$L(G) = \{ v \in V_T^* \mid S \Rightarrow^* v \}.$$

A family of languages is a nonempty set of languages closed under isomor-
phism (with respect to the operation of concatenation in this case), i.e., renaming of letters.

**THEOREM 2.2.** The family of languages \( \{L \mid L = L(G) \text{ for some generative grammar } G \} \) equals the class of recursively enumerable languages.

According to Church's thesis, the class of recursively enumerable languages is the largest class of sets obtainable by effective means. By successive restrictions on the form of the production rules, we obtain successively restricted classes of grammars.

**DEFINITION 2.3.** A grammar \( G = \langle V_N, V_T, S, P \rangle \) is of type \( i \) if the restrictions

1. on \( P \), as given below, are satisfied.
2. No restrictions.
3. Each production in \( P \) is of the form \( a_1Xa_2 \rightarrow a_1va_2, \ a_1, a_2 \in V^*, \ X \in V_N \) and \( v \) is a nonempty word over \( V \), with the possible exception of the production \( S \rightarrow \lambda \) whose occurrence in \( P \) implies, however, that \( S \) does not occur on the righthand side of any other production in \( P \).
4. Each production in \( P \) is of the form \( X \rightarrow \beta \) where \( X \in V_N \) and \( \beta \in V^* \).
5. Each production in \( P \) is of one of the two forms \( X \rightarrow Y \alpha \) or \( X \rightarrow \alpha \) where \( X, Y \in V_N \) and \( \alpha \in V_T^* \).

We call the grammars of types 0,1,2, and 3, recursively enumerable, context sensitive, context free and regular, respectively. We denote the corresponding families of languages by \( \text{RE} \), \( \text{CS} \), \( \text{CF} \) and \( \text{REG} \).

**THEOREM 2.4.** \( \text{REG} \) equals the class of regular sets.

**THEOREM 2.5.** \( \text{REG} \subset \text{CF} \subset \text{CS} \subset \text{RE} \) where "\( \subset \)" denotes strict inclusion.

These (by inclusion) nested language families make up the so-called Chomsky hierarchy. We call languages in the difference \( X-Y \), \( Y \) and \( X \) in sequence as in Theorem 2.5, strictly \( X \) where \( Y \) is understood.

**EXAMPLE 2.6.**

\( L_0 = \{ f^n(n) \mid n \geq 0 \} \in \text{RE} - \text{CS} \), if \( f: \mathbb{N} \rightarrow \mathbb{N} \) enumerates some nonrecursive, but recursively enumerable, set.

\( L_1 = \{ a^n b^n c^n \mid n \geq 1 \} \in \text{CS} - \text{CF} \).

\( L_2 = \{ a^n b^n \mid n \geq 1 \} \in \text{CF} - \text{REG} \).

\( L_3 = \{ a^n \mid n \geq 1 \} \in \text{REG} \).
L₄ = {aⁿ | n is a prime number} ∈ CS - CF, to give a feeling of the power of context sensitivity as opposed to context freeness.

Above it was shown how the four main language families of the Chomsky hierarchy are derived by classes of generating devices, viz., by suitable restrictions on the form of the production rules in grammars. They can also be characterized by accepting devices, i.e., classes of machines which accept exactly the languages generated by a class of grammars. By acceptance we mean, that if L is the language accepted by a machine M then M enters an accepting configuration after reading a word v iff v ∈ L. So RE is accepted by Turing machines, CS is accepted by Linear Bounded Automata, CF is accepted by Pushdown Automata and REG is accepted by Finite Automata. When, where and if, necessary we shall introduce these devices.

2.2. LINDENMAYER SYSTEMS

As we have seen in the previous section, a generative grammar is a sequential rewriting system, i.e., in each production step part of the string is rewritten. L systems are rewriting systems where we rewrite all letters in a string simultaneously in each production step. Moreover, they have no terminal symbols in the sense of formal grammars as defined above.

EXAMPLE 2.7. The production rule a → aa yields, if we start with the string a, the string sequence a, aa, a₁, a₂, a₃, ..., and the produced language is {aⁿ | n = 2ⁱ, i ∈ IN}. This system is context free (each letter is rewritten independent of the context in which it occurs) and deterministic (letters can be rewritten in but one way). In a context sensitive L system the letters in a string are rewritten by the production rules, according to the context in which they occur. In an (m, n) L system this context consists of the m left- and n right letters of the letter to be rewritten, and we rewrite letters according to the production rules which are applicable to the letter with its m letter left- and n letter right context in the string before the rewriting. Formally,

DEFINITION 2.8. An (m, n) L system is a triple G = <W, P, w> where W is a finite nonempty alphabet, w ∈ W⁺ is the initial string, and

$$
P ≤ \bigcup_{i=0}^{m} (W W^i × W × \bigcup_{i=0}^{n} W W^i) × W^*$$
is a finite set of production rules. We write an element of $P$ also as $(u,a,v) \rightarrow a$ where $u \in \bigcup_{i=0}^{m} W^{i}$, $a \in W$, $v \in \bigcup_{i=0}^{n} W^{i}$ and $a \in W^{*}$.

We derive strings by the system as follows. $\rightarrow$ induces a relation $\Rightarrow$ on $W^{*}$ defined by

$$a_{1}a_{2}\ldots a_{k} \Rightarrow a_{1}a_{2}\ldots a_{k}$$

$$a_{1}a_{2}\ldots a_{k} \in W \text{ and } a_{1}a_{2}\ldots a_{k} \in W^{*},$$

if

$$(a_{i-m}a_{i-m+1}\ldots a_{i-1},a_{i+1}a_{i+2}\ldots a_{i+n}) \rightarrow a_{i} \in P$$

for all $i$, $1 \leq i \leq k$, where we take $a_{j} = \lambda$ whenever $j < 1$ or $j > k$. If $v_{0} \Rightarrow v_{1} \Rightarrow v_{2} \Rightarrow \ldots \Rightarrow v_{\ell}$ for some $v_{0}, v_{1}, \ldots, v_{\ell} \in W^{*}$ we write $v_{0} \Rightarrow^{\ell} v_{\ell}$ and say $v_{0}$ derives or produces $v_{\ell}$ in $\ell$ steps. As usual, $\Rightarrow^{*}$ and $\Rightarrow^{\ast}$ are the transitive reflexive closure and the transitive irreflexive closure of $\Rightarrow$, respectively. (I.e., $\Rightarrow^{*} = \bigcup_{\ell=0}^{\infty} (\Rightarrow^{\ell})$ and $\Rightarrow^{\ast} = \bigcup_{\ell=1}^{\infty} (\Rightarrow^{\ell})$.)

A sequence produced by $G$ is a sequence $v_{0}, v_{1}, \ldots, v_{t}, \ldots$ where $v_{i} \Rightarrow v_{i+1}$ for all $i \geq 0$. In case $G$ is deterministic the produced sequence is unique relative to $v_{0}$, and if, moreover, $v_{0} \leftrightarrow w$ then the produced sequence is called the string sequence $S(G)$ associated with $G$. The language produced by $G$ is:

$$L(G) = \{v \in W^{*} | w \Rightarrow^{\ast} v\}.$$  

We subscript the relations $\rightarrow$, $\Rightarrow$, $\Rightarrow^{*}$, $\Rightarrow^{\ast}$, (i) with the appropriate identifiers when necessary. Similarly to the generative grammars in the previous section we obtain classes of $L$ systems by imposing restrictions on the form of the production rules.

**DEFINITION 2.9.** Let $G = \langle W, P, w \rangle$ be an $(m,n)$ $L$ system.

(i) Without any restriction $G$ is called context sensitive, or interacting, and the corresponding class of $L$ systems is denoted as IL systems.

With fixed $m$ and $n$ we call the corresponding class the class of $(m,n) L$ systems.

(ii) If $(u,a,v) \rightarrow a \in P$ implies that $u,v = \lambda$ then $G$ is context free, or interactionless, and the corresponding class of systems is denoted as OL systems. For ease of notation we write rules in $P$ as $a \rightarrow a$.

(iii) If for each $(u,a,v) \rightarrow a \in P$ either always $v = \lambda$ or always $u = \lambda$ then $G$ is left- or right context sensitive, and the corresponding class of systems is denoted as IL$^{-}$ or IL$^{+}$ systems. For ease of notation we
write rules in \( P \) as \((u,a) \rightarrow a\) or \((a,v) \rightarrow a\), respectively.

(iv) If \((u,a,v) \rightarrow a\) and \((u,a,v) \rightarrow a'\) imply that \(a = a'\) then \( G \) is called deterministic, and we indicate this property by prefixing a "D" in the denotation of the class of systems. We also denote the set of production rules \( P \) by a function \( \delta \), and write \( \delta(u,a,v) = a \) for \((u,a,v) \rightarrow a\) \( \in P \). We extend \( \delta \) to \( W^* \) by defining \( \delta(v) = v' \) if \( v \Rightarrow v' \), \( v,v' \in W^* \). \( \delta^i \) is defined as the \( i \)-fold composition of \( \delta \): \( \delta^0(v) = v \) and \( \delta^i(v) = \delta(\delta^{i-1}) \), \( i \geq 1 \).

(v) If \((u,a,v) \rightarrow a\) implies \(a \neq \lambda\) the system is nonerasing or propagating and we denote this property by prefixing a "P" in the denotation of the class.

Hence we have, e.g., PDIL systems, DOL systems, D(m,n)L systems, DL systems etc. The following notation is standard throughout the literature and partly follows from above.

\[
\begin{align*}
(0,0)L \text{ systems} & \equiv OL \text{ systems}, \\
(1,0)L \text{ systems} & \text{ or } (0,1)L \text{ systems} \equiv IL \text{ systems}, \\
(1,1)L \text{ systems} & \equiv 2L \text{ systems}.
\end{align*}
\]

The notion of \( L \) systems has been extended to the important table \( L \) systems.

**DEFINITION 2.10.** A table \( L \) system with \( q \) tables, \( T L \) system, is a triple \( G = <W,P,w> \) where \( P = \{P_1,P_2,\ldots,P_q\} \), \( W \) and \( w \) are as before, and \( P_1, \ldots, P_q \), \( 1 \leq i \leq q \), is as \( P \) in Def. 2.8. Therefore, a table \( (m,n)L \) system is a triple \( G = <W,P,w> \) with \( P = \{P_1,P_2,\ldots,P_q\} \) such that for each \( i, 1 \leq i \leq q \), \( G_i = <W,P_i,w> \) is an \((m,n)L\) system.

Strings are derived in a table \( L \) system \( G = <W,P,w> \) as follows.

\[
a_{i_1}a_{i_2}\ldots a_k \xrightarrow{G} a_{i_1}a_{i_2}\ldots a_k' \quad a_{i_1}a_{i_2}\ldots a_k \in W \text{ and } a_{i_1}a_{i_2}\ldots a_k \in W^*,
\]

if there is a table \( P_{i_1} \) in the set of tables \( P \) such that

\[
a_{i_1}a_{i_2}\ldots a_k \xrightarrow{G} a_{i_1}a_{i_2}\ldots a_k' \quad G \text{ and } G' \text{ are the usual closures of } G.
\]

The language generated by \( G \) is defined by \( L(G) = \{v | w \xrightarrow{G} v\} \), etc. \( G \) is a XYZTL system if, for \( P = \{P_1,P_2,\ldots,P_q\} \) each \( G_i, 1 \leq i \leq q \), is an XYZTL system. E.g., PDT1L systems, TOL systems etc.
(No subscript on T means that \( q \geq 1 \), no \( T \) at all means that \( q = 1 \). E.g., \( PD_{1,OL} \) systems are PDOL systems.) The family of languages generated by \( X^q Y^1 Z^q \) systems is denoted as \( X^q Y^1 Z^q \).

**Definition 2.11.** A semi \( X^q Y^1 Z^q \) system is a \( X^q Y^1 Z^q \) system without the initial string.

We can squeeze languages out of L systems in various ways. One way is (as above) to consider all strings generated from the initial string: the pure L language of the system. By dividing the alphabet into a set of terminals and a set of nonterminals, we can consider the language consisting of all strings over the terminals occurring in the pure L language. Such a language is called an extension language, since the terminal–nonterminal mechanism extends the generating power of a class of L systems. Another device is to take a homomorphism *) of a pure L language or extension language. A third method we shall meet is to consider the stable string language of an L system. That is, the set of all strings, occurring in the pure L language, which are invariant under the rewriting rules. Accepting devices for families of L languages, similar to machine type characterizations of families of languages produced by classes of generative grammars like in Section 2.1, are not treated in this work. They have, however, been studied in van Leeuwen [1974], Rozenberg [1974] and Savitch [1975].

2.3. BIBLIOGRAPHICAL COMMENTS

Generative grammars were introduced by Chomsky [1957]. The concepts and results in Section 2.1 are treated in any textbook on the subject, like Hopcroft and Ullman [1969] or Salomaa [1973a]. Lindenmayer systems were

*) By homomorphisms we will mean monoid homomorphisms which are mappings between monoids in which the operation is concatenation. More precisely, the free monoid \( S \) finitely generated by a finite alphabet \( W \) is \( W^* \), i.e., if \( x, y \in S \) then so does \( xy \). The operation of concatenation is associative; \((xy)z = x(yz)\) and the identity element of \( S \) is the empty word \( \lambda \): the word with no letters. If \( W^* \) and \( V^* \) are two monoids and \( h: W^* \rightarrow V^* \) is a homomorphism between them then \( h(\lambda) = \lambda, h(xy) = h(x)h(y) \) for all \( x, y \in W^* \). We extend the concept of homomorphisms to sets by defining \( h(L) = \{ h(w) | w \in L \} \) for each subset \( L \subseteq W^* \).
proposed by LINDENMAYER [1968a,b] and a first formal language type of treatment was given by HERMAN [1969] and van DALEN [1971]. Table L systems were introduced by ROZENBERG [1973a] and the concept of stable string languages of L systems is due to WALKER [1974a,b]. Extension languages of L systems were first introduced by van LEEUWEN (unpublished) who, with an alternative interpretation of the concept, called them restriction languages. A textbook covering most of the research (done in collaboration with the authors) in L system theory up to 1972–1973 is HERMAN and ROZENBERG [1975]. Collections of research papers and tutorials presented at L system conferences are contained in the PROCEEDINGS of an Open House in Unusual Automata Theory [1972], PROCEEDINGS of the IEEE Conference on Biologically Motivated Automata Theory [1974], ROZENBERG and SALOMAA [1974] and LINDENMAYER and ROZENBERG [1976].