TECHNICAL REPORT 400-165

LANGUAGES, AUTOMATA AND CLASSES OF CHAIN - ENCODED PATTERNS

Jerome Feder

August, 1967

Sponsored by
AIR FORCE OFFICE OF SCIENTIFIC RESEARCH
INFORMATION SCIENCES DIRECTORATE

Prepared under
GRANT AF-AFOSR-24-66

NEW YORK UNIVERSITY
SCHOOL of ENGINEERING and SCIENCE
DEPARTMENT of ELECTRICAL ENGINEERING
UNIVERSITY HEIGHTS BRONX, NEW YORK 10453

LABORATORY FOR
ELECTROSCIENCE RESEARCH

LANGUAGES, AUTOMATA AND CLASSES OF CHAIN-ENCODED PATTERNS

Jerome Feder

New York University
New York, N. Y.

August 1967

TECHNICAL REPORT 400-165

LANGUAGES, AUTOMATA AND CLASSES OF CHAIN-ENCODED PATTERNS

Jerome Feder

August 1967

NEW YORK UNIVERSITY
SCHOOL OF ENGINEERING AND SCIENCE
DEPARTMENT OF ELECTRICAL ENGINEERING
Laboratory for Electroscience Research

University Heights
Bronx, New York 10453

ACKNOWLEDGMENT

This research was sponsored by the Air Force Office of Scientific Research, Directorate of Information Sciences, AFOSR Grant AF-AFOSR-24-66.

ABSTRACT

By treating patterns as statements in a two-dimensional language, it is possible to apply linguistic theory to pattern analysis and recognition. In this report, patterns are encoded into string form using the chain code developed by Freeman. A class of patterns, or pattern language, encodes to a set of strings that is analyzed using the large body of theory that exists for string languages. The known relationships between classes of string languages and classes of automata are applied to determine the computational power required to recognize various patterns. Pattern languages formed on the basis of equations in two variables, pattern properties, and various notions of pattern similarly to an arbitrary given pattern are related to the hierarchy of string language classes. The extension of results to other pattern encoding schemes is considered.

		Page
I.	INTRODUCTION	1
II.	LANGUAGES AND AUTOMATA	4
	2.1 Classification of String Languages	4
	2.2 Classes of Automata	8
	2.3 Relations Between Classes of Languages and Automata	12
	2.4 Deterministic and Nondeterministic Automata	13
	2.5 Boolean Functions of Languages	14
	2.6 Product Languages	15
III.	PATTERN LANGUAGES BASED ON EQUATIONS IN TWO VARIABLES	17
	3.1 Straight Lines	20
	3.2 Circles and Circular Arcs	24
	3.3 Other Curves	29
IV.	PATTERN LANGUAGES BASED ON PATTERN PROPERTIES	
	4.1 Closure	31
	4.2 Self Intersection	31
	4.3 Convexity	32
	4.4 Periodicity	33
V.	PATTERN LANGUAGES CONSISTING OF CHAINS THAT ARE SIMILAR IN SOME SENSE TO A GIVEN CHAIN	41
	5.1 Languages Consisting of Chains Similar in Size, Shape and Orientation	42
	5.2 Languages Based on Other Definitions of Similarity	46
VI.	OTHER METHODS OF PATTERN ENCODING	48
VII.	CONCLUSIONS	50
REFERENCES		51

LANGUAGES, AUTOMATA AND CLASSES TO CHAIN-ENCODED PATTERNS

I. INTRODUCTION

A great deal of work has been done in relating classes of string languages to the types of automata required to generate or recognize the strings of the languages. It is possible to consider patterns as being a form of two-dimensional language. A pattern language, L, can be defined to be any subset of the universe of all patterns, U. A previous report[1] by the author has suggested the possibility of classifying pattern languages in the same manner as string languages, according to the type of automata required to accomplish pattern generation and recognition. A pattern language classification scheme of this sort would provide information as to what pattern analysis and recognition requirements could be fulfilled with various types of programs and computation facilities, and in some cases, on whether pattern recognition could be accomplished at all.

This report is a start in the direction of providing such a classification system for pattern languages. Consideration is restricted to line patterns encoded in the chain code developed by Freeman.[2] This encoding method represents a line pattern by a sequence of octal digits called a chain. The results obtained can be extended to other forms of encoding provided that a translator between codes can be built that satisfies certain conditions. The extension of results to other pattern encoding schemes is considered in the report.

The work described here is intended to form a linguistic basis for the detection of basic object types in the table-driven pattern analysis

system. This system has been outlined in a previous report,[1] and when completed will be able to analyze a broad class of line patterns according to prespecified grammar rules. Ultimately, it is hoped that examination of the linguistic power of each of the operations performed by the pattern analysis system will lead to a form of analytic grammar[3] for pattern languages. Such a grammar will be able to formally specify and analyze a much richer class of patterns than can be accomplished using operations on chains alone.

An introduction to the classes of automata and string languages to which pattern classes are related is given in Section II. Since the number of different types of languages and automata that have been delimited is rather large, only some of the more important of these have been selected for comparison with pattern languages. Languages formed by Boolean functions of languages and by the concatenation of strings of a number of languages are discussed in this section. The latter are related to problems that arise in the recognition of patterns that occur as parts of chains.

Section III examines pattern languages based on families of equations in two variables. Each of the languages consists of the set of chains formed by the chain encoding of a class of patterns. Pattern languages formed from chains of straight lines, circles, and circular arcs are related to string language classes. In Section IV, pattern languages based on pattern properties such as closure, self intersection, convexity, and periodicity are examined. Pattern languages formed on the basis of a

number of pattern properties are expressed as Boolean functions of the languages based on the individual properties. Section V examines the languages consisting of chains similar in various ways to an arbitrary given chain. The extension of results to other forms of pattern encoding is considered in Section VI.

For background material and information on what others have done in applying linguistics to pattern recognition and analysis, the reader is referred to a separate literature survey report by the author.[4]

II. LANGUAGES AND AUTOMATA

This section contains an introduction to some of the classes of string languages and automata to which the pattern languages in Sections III, IV and V are related and gives some linguistic results that are of use in dealing with such languages. An outline of some major classes of string languages is given first in Section 2.1. Section 2.2 contains a description of some of the more important classes of automata. Important relations between classes of string languages and classes of automata are given in Section 2.3. Section 2.4 discusses some problems concerning non-deterministic automata. In Section 2.5, some linguistic results pertaining to Boolean functions of languages are given. Finally, the recognition of languages whose strings are formed by concatenating the strings of other languages is discussed in Section 2.6.

Most of the material in Section II is well known to linguists. Proofs of all the stated results would render the size of this report prohibitive. Consequently, results are stated without proof and references are given.

2.1 Classification of String Languages

The pattern languages considered in this report are composed of sets of chains. Since a chain is a string, these pattern languages can be treated as string languages. Classes of string languages and automata have been related according to the type of automaton required to accomplish string recognition. Therefore, if it can be shown that the set of chains that forms a particular pattern language falls into one of the

existing string language classes, then the automaton required for the detection or recognition of this pattern language is immediately known. For this reason it is useful to examine the main classes of string languages and the methods of generating these classes.

A string language is defined as follows.[5] An alphabet, $\mathcal{A}$, is a finite set of symbols such that any string in $\mathcal{A}$ can be uniquely decomposed into its component symbols. For any such alphabet there is an infinite set, U, of finite strings of symbols from $\mathcal{A}$. A specification that will select a subset, L, of U is called a formal syntax or grammar of the string language, L. Although a formal syntax or grammar may be any method for selecting L, of particular interest are the phrase structure grammars (also called constituent structure grammars). The notation and terminology used in the following is that of Chomsky.[6,7]

A phrase structure grammar, G, is a four-tuple

$$G = [V_T, V_N, P, S]$$

where:

V_T is a finite nonempty set called the terminal vocabulary

V_N is a finite nonempty set called the nonterminal vocabulary

$V_T \cap V_N = \emptyset$

P is a finite set of rewrite rules or productions of the form, $\phi \rightarrow \omega$, where ϕ and ω are strings of symbols $\in V_T \cup V_N$

$S \in V_N$ is called the sentence of G.

The terminal vocabulary is the alphabet ($\mathcal{A}$) from which the sentences of the

language are constructed. The nonterminal vocabulary is the set of symbols used to represent intermediate syntactic entities used in the generation of the strings of the language. The relation "$\rightarrow$" is read "is rewritten as," and is specified by a finite set of pairs $(\psi,\omega) \in P$. It is customary also to write $\varphi \rightarrow \chi$ when only a substring of φ is rewritten i.e. $\varphi \rightarrow \chi$ when $\varphi = \varphi_1 \psi \varphi_2$, $\chi = \varphi_1 \omega \varphi_2$, and $(\psi,\omega) \in P$. S serves as an initial symbol. A string of L is generated by writing down the initial symbol, S, applying one of the rewrite rules to form a new string, ψ_1, applying another rule to form a new string, ψ_2, and so on until a string, ψ_n, is obtained that consists only of terminal symbols and cannot be further rewritten.

A number of notational conventions are adopted to facilitate the exposition of a phrase structure grammar and the proofs in Sections III, IV, and IV. Small roman letters are used for denoting strings in V_T; capital letters for strings in V_N; Greek letters for arbitrary strings; early letters of all alphabets for single symbols; late letters of all alphabets for strings.

A sequence of strings $(\psi_1 \ldots \psi_n)$ $(n \geq 1)$ with $\psi_1 = \chi$, $\psi_n = \omega$ and $\psi_i \rightarrow \psi_{i+1} (1 \leq i < n)$ is called a <u>χ-derivation of ω</u>. The existence of a χ-derivation of ω is denoted $\chi \Rightarrow \omega$. A χ-derivation is <u>terminated</u> if it is not a proper initial subsequence of any χ-derivation. The set of strings, z, of elements of V_T, such that there exists a terminated S-derivation of z, forms the language, L, generated by G.

Phrase structure grammars may be classified into five types by imposing the following restrictions:

Type 0 no restrictions

Type 1 $(\gamma,\omega) \in P$ implies that there are A, γ_1, γ_2 and χ such that $\gamma = \gamma_1 A \gamma_2$, $\omega = \gamma_1 \chi \gamma_2$ where $\chi \neq$ null

Type 2 only rules of the form (A, χ) where $\chi \neq$ null are permitted

Type 3 only rules of the form (A,χ) are permitted, where for all rules χ is of the form (i) or (ii), or for all rules χ is of the form (i) or (iii):

(i) a
(ii) aB
(iii) Ba

Type 4 only rules of the form (S,z) are permitted

For $j = 0,1,2,3,4$, a language is a <u>type j language</u> if there is a type j grammar that can generate it. Type 0 grammars are called <u>unrestricted grammars</u>. Type 1 grammars are called <u>context-sensitive</u> since the symbol replacement depends on the symbols adjacent to (the context) of the symbol being replaced. It has been shown[6] that the class of languages that can be generated by a type 1 grammar is not changed if the restriction is weakened to requiring only that if $\gamma \rightarrow \omega$, then $\ell(\omega) \geq \ell(\gamma)$, where ℓ denotes the length of its string argument. Type 2 grammars and languages are called <u>context-free</u> since the symbol replacement is performed independent of the context of the symbol. Type 3 grammars are called <u>one-sided linear</u> and the languages generated by such grammars are called <u>regular languages</u>. Type 4 languages can only contain a finite number of strings.

Such languages are said to be finite.

THEOREM 1. (Chomsky)[6] For both languages and grammars type 0 ⊇ type 1 ⊃ type 2 ⊃ type 3 ⊃ type 4

2.2 Classes of Automata

The following major classes of automata are discussed:

A. Turing machines
B. Linear bounded automata
C. Pushdown-storage automata
D. Finite automata
E. K-limited automata

Each of these classes of automata is less powerful and constitutes a subclass of the class of automata preceding it on the list. The automata class descriptions that follow are intended to form an outline only. Strict mathematical definitions of these classes of automata as well as proofs of the results that are stated can be found in the literature.[7,8,9]

A. Turing Machines

The most powerful automata are the Turing machines. In its simplist form a Turing machine consists of a control unit with a finite number of states, a tape that can move to the left or right, and a read-write head for reading and writing characters of the output alphabet, a_0 ($a_0 \supset I$) on the tape. The behavior of the device is specified by the five-tuples [i,j,k,l,m]. This rule states that if the device is in state S_j reading the symbol $a_i \in a_0$, it will switch to state S_k, write the symbol $a_m \in a_0$ in the space formerly occupied by a_i, and then move the tape l squares to the left. The rules of a Turing machine permit the rewriting of blank

squares of the tape so that an infinite amount of storage is available. Increasing the number of tapes that a Turing machine can have does not change its basic computation power.

A Turing machine is used to decide whether a given string, z, is a member of the language, L, as follows. The control unit begins computation in an initial state, S_0, with its read-write head scanning the left-most symbol of z. If at the end of computation the device returns to S_0 and halts, it is said to have accepted or detected z. If the device halts in a state other than S_0 or does not halt at all, it is said to have rejected z. The language, L, consists of the set of all strings that are accepted by the given Turing machine. A set of strings defined in such a manner by a Turing machine is said to be recursively enumerable.

There exists a general problem concerned with the use of recursively enumerable sets of strings as languages. Since there is no time limit on the computation of the Turing machine, there is no way of telling after any length of time whether the Turing machine will eventually stop and either accept or reject the input string, or just keep running forever. All that is known is that if the input string is going to be accepted then the Turing machine will halt in a finite amount of time. If both a set and its complement are recursively enumerable then the set is called recursive. If a set is recursive then there is a Turing machine that will either accept or reject an element of the set in a finite amount of time. Recursive sets of strings can thus be used as languages.

The question arises as to what the relationship of a Turing machine is to commonly occurring "general purpose" computers. It is possible to

construct a Turing machine called a universal Turing machine that can be programmed to carry out any computation that can be carried out on any Turing machine or computer. All general purpose computers that have provision for changing tapes or disks, or for other arbitrary increases in storage during computation, are such universal Turing machines.

B. Linear Bounded Automata[9]

If in the definition of the single-tape Turing machine the additional restriction is made that the device cannot rewrite blank squares of the tape, the class of linear bounded automata (LBA) is obtained. Such automata can write on, and thus use for computation, only the portions of the tape occupied by the input string. The amount of storage available for computation is a function of the length of the input string, z, and is given by $c\ell(z) + q$, where q is the fixed storage of the control unit and c is a constant determined by the size of the output alphabet of the device. A LBA tests an input string for membership in a language and signifies acceptance and rejection of the string in the same manner as a Turing machine. For a LBA, however, if the computation time is finite, it has an upper limit, t_M, expressible as a function of the length of the input string. Therefore, it is not necessary to wait forever to be sure that a LBA will reject an input string; when computation time exceeds t_M it is certain that the LBA will never halt and the input string can be rejected. All sets of strings that are specified by LBA are thus recursive.

LBA are shown in Sections III, IV, and V to be capable of detecting a number of pattern languages. In demonstrating this capability the

following theorem concerning storage available for computation is used. The proof of this theorem is given in the Appendix.

THEOREM 2. A LBA has sufficient storage for the execution of a program if there exist integers, b, m, such that for all inputs, z, the storage of no more than m integers is required, and none of these integers is larger than $b^{\ell(z)}-1$.

C. Pushdown-Storage Automata

The pushdown-storage automata (PDSA) are obtained by appropriately restricting the LBA. In order to do this it is useful to view a LBA as possessing two separate infinite tapes, one solely for input, and the other a storage tape used solely for the computation; the storage tape having as many squares available for computation as are occupied by the symbols appearing on the input tape. This change does not affect the computational power of the LBA.

A PDSA is a LBA meeting the following restrictions:

(i) The input tape can move in only one direction.

(ii) Everything to the right of the read-write head of the storage tape is automatically erased.

The storage tape is equivalent to a pushdown-stack, a storage mechanism analogous to a cafeteria well for storing dishes. The pushdown-stack organization of memory has found wide application in programming. A PDSA accepts an input string, z, if it returns to S_0 after reading z and halts with the storage tape blank.

D. Finite Automata

By eliminating the storage tape from a PDSA, an automaton that

consists of a finite state control unit and a tape that can move in only one direction is obtained. Since the input tape can move in only one direction, it is useless for storage purposes. The only storage available to this device is that of the finite state control unit; hence the device is called a finite automaton. A finite automaton accepts an input string by simply returning to S_0 and halting. If only reading of the input tape is permitted then allowing the input tape to move in both directions does not change the class of languages that can be recognized.

E. K-Limited Automata

A K-limited automaton is a finite state automaton for which the state of the control unit is dependent only upon the previous K symbols of the input that have been accepted, for some fixed K. The behavior of a K-limited automaton can be specified by a D^K X D matrix, where D is the number of symbols in A.

2.3 Relations Between Classes of Languages and Automata

Close correspondences have been found between the hierarchy of language classes described in Section 2.1 and the classes of languages that can be detected by the hierarchy of automata given in Section 2.2. The following grammars and automata are equivalent in linguistic power.[7,8,10]

Phrase structure grammar	Class of automata
type 0 (unrestricted)	Turing machines
type 1 (context-sensitive)	LBA
type 2 (context-free)	PDSA
type 3 (one-sided linear)	finite automata

2.4 Deterministic and Nondeterministic Automata

The preceding discussion has not distinguished automata for which behavior is completely determined and automata for which it is not. Automata that have tape-machine configurations to which more than one transition rule can apply are said to be nondeterministic. Automata in which the transition rule that applies to a given tape-machine configuration is unique are called deterministic. Nondeterministic automata, although useful for theoretical purposes, cannot actually be constructed since their operation is not fully specified. A language definition by a nondeterministic automaton is a statement that the automaton has transition rules that would lead to the acceptance of all strings of the language and no others. To any string of a language defined by a nondeterministic automaton there exists a set of transition rule choices for ambiguous situations that would lead to the acceptance of the string. Acceptance of the string is dependent upon the correct transition rule being chosen in all ambiguous situations encountered; if the correct rules are not chosen, then the string may very well be rejected by the automaton.

Since to a large extent the concern here is with realizable systems, care must be exercised with proofs that a language can be detected by a class of automata unless it can be also established that this detection can be accomplished with a deterministic automaton. If it is shown, however, that a pattern language cannot be detected by a class of automata, then it is known that this language cannot be detected by such automata, deterministic or nondeterministic.

For finite automata and Turing machines, problems of this type do not occur. Nondeterministic finite automata have deterministic equivalents; in the remainder of this report, deterministic and nondeterministic finite automata are not distinguished. The term "Turing machine" customarily refers to a deterministic device. Since the linguistic power of deterministic and nondeterministic devices of this class is equivalent, no problems arise. With LBA and PDSA, however, care must be exercised. A result of some use is that every context-free language is accepted by a deterministic LBA.[10]

2.5 Boolean Functions of Languages

This section states without proof some results of mathematical linguistics pertaining to the complement, union and intersection of languages. The complement of a language, L, denoted $\bar{L}$, is the set of all strings on the alphabet of L that are not members of L. The union of two languages, L_1 and L_2, denoted $L_1 \cup L_2$, is the set of all strings that are members of L_1 or of L_2. The intersection of two languages, denoted $L_1 \cap L_2$, is the set of all strings that are members of L_1 and of L_2. Complex pattern languages can be expressed in terms of relatively simple ones using the concepts of language complement, union and intersection. The theorems that follow show how the language classification of these more complex languages can be found from the language classifications of their constituents.

THEOREM 3. (Chomsky and Miller)[11] The set of regular languages is closed under the Boolean operations. If L_1 and L_2 are regular languages, then $\bar{L}_1$, $L_1 \cup L_2$ and $L_1 \cap L_2$ are regular languages.

THEOREM 4. (Bar-Hillel, Perles and Shamir)[12] If L_1 and L_2 are context-free languages, then $L_1 \cup L_2$ is a context-free language.

The set of context-free languages is not closed under intersection and complementation.[12] Therefore, that L_1 and L_2 are context-free does not necessarily imply that $\bar{L}_1$ or $L_1 \cap L_2$ are context-free.

THEOREM 5. (Bar-Hillel, Perles and Shamir)[12] If L_1 is a regular language and L_2 is a context-free language, then the intersection $L_1 \cap L_2$ is a context-free language.

THEOREM 6a. (Landweber)[13] If L_1 and L_2 are context-sensitive languages, then the intersection $L_1 \cap L_2$ is a context-sensitive language.

THEOREM 6b. (Kuroda)[10] The set of languages that are accepted by deterministic LBA is closed under the Boolean operations. If L_1 and L_2 are accepted by deterministic LBA, then $\bar{L}_1$, $L_1 \cup L_2$ and $L_1 \cap L_2$ are accepted by deterministic LBA.

2.6 Product Languages

The preceding has been concerned with languages comprised of whole strings. Often a chain can be divided into a number of smaller chains, with each of the smaller chains being itself a member of a pattern class. For instance, a chain representing a straight line joined to the tip of a circular arc or another straight line can advantageously be broken into parts for analysis. There is a linguistic concept that is of great use in this regard. The language, $L = L_1 * L_2 = \{z | z = z_1 z_2,$ where $z_1 \epsilon L_1$ and $z_2 \epsilon L_2\}$ is called the language product of L_1 and L_2. Product languages can be formed from any number of languages. The following

theorems give the language class membership of product languages in terms of the language class membership of their constituents.

THEOREM 7. (Kleene)[14] The product of two regular languages is a regular language.

THEOREM 8. (Bar-Hillel, Perles and Shamir)[12] The product of two context-free languages is a context-free language.

An important problem arises in connection with product languages. If the product strings are ambiguous (i.e., if there is more than one way of generating or analyzing them), then a unique separation into strings of component languages may not be possible. This problem can arise even if the component languages are themselves non-ambiguous and recognizable with deterministic automata. As a result, the analysis of chains composed of smaller chains belonging to a number of pattern languages is more difficult than the determination of whether an entire chain belongs to a given pattern language.

III. PATTERN LANGUAGES BASED ON EQUATIONS IN TWO VARIABLES

The pattern languages considered in this report are composed of the sets of chains representing various classes of patterns. The alphabet, $\mathcal{A}$, of such languages is the set of eight octal digits from which chains are formed. If a bound, M, is assumed on the length of the chains that comprise a pattern language, L, then this language can have at most $8^M + 8^{M-1} + \ldots + 8^0$ chains. A finite list can be made of the chains that belong to L and a K-limited automaton with K = M-1 can be constructed that will exhaustively check any input chain against this list. Hence the following theorem is obtained.

THEOREM 9. A pattern language consisting of chains, all of length less than or equal to some bound, M, is a finite language and can be detected by a K-limited automaton with K = M-1.

In order to show the characteristics inherent in the various pattern languages considered, it is necessary that no bound be assumed on chain length. For the remainder of the report this assumption is made.

Section III considers pattern languages consiting of chains of continuous curves that can be represented by a family of equations in two variables. The curves are specified by the three-tuple [f,R,Q] where:

$f(x,y) = 0$ is an equation in two variables describing a continuous curve.

R is a set of regions $(r_1, r_2, \ldots, r_k)$ of the curve $f(x,y) = 0$. To each r_i correspond two coordinate pairs, $(\underline{x}_i, \underline{y}_i)$ and $(\overline{x}_i, \overline{y}_i)$. The region of the curve described by r_i is $\underline{x}_i \leq x \leq \overline{x}_i$, $\underline{y}_i \leq y \leq \overline{y}_i$.

Q is a set of parameter ranges $(\underline{q}_1, \overline{q}_1)$, $(\underline{q}_2, \overline{q}_2), \ldots, (\underline{q}_l, \overline{q}_l)$. Each of these pairs is a coefficient or simple function of coefficients appearing in $f(x,y)$. The coefficient or function of coefficients corresponding to $(\underline{q}_i, \overline{q}_i)$ is denoted by q_i.

It is desirable to examine two separate methods for defining a pattern language, L, on the basis of a given $[f,R,Q]$.

Strict Definition: A chain, C, belongs to the pattern language defined by $[f,R,Q]$ if and only if there exist values of coefficients within Q and a position of the encoding grid such that if a region of $f(x,y)$ within R is chain encoded, the resultant chain is identical to C.

The superscript, s, (L^s) is used to designate languages defined according to the strict definition. For the second type of definition, an algorithm, Γ, called the RQ-algorithm, is introduced. This algorithm assumes that a given chain, C, is a member of L and attempts to calculate the coefficients and region of $f(x,y)$, respectively, that yield a curve that is within some tolerance, T, of C. The test for tolerance is made by superimposing the curve and chain, with the initial points of curve and chain coinciding. If throughout their length the curve and chain are never separated by more than T squares, then it is said that the curve and chain conform within T. The complete language specification is a five-tuple, $[f,R,Q,\Gamma,T]$.

The pattern language definition is as follows.

Weak Definition: A chain, C, belongs to the pattern language specified by [f,R,Q,Γ,T] if and only if the following conditions are satisfied.

(i) The RQ-algorithm, Γ, is successful in its calculation of coefficients and range of f(x,y) for C.

(ii) The coefficients and range calculated in (i) yield a curve that conforms within T to C.

Languages defined according to this definition are designed by the superscript, w, (L^W).

Given a chain, a "maze" can be constructed within which all curves that encode to this chain must lie.[15] Determining whether a given chain, C, is a member of a particular L^S is equivalent to determining if there exists a curve or curves in the [f,R,Q] for L^S that can be drawn entirely within the maze defined by C. In many instances it is possible to find algorithms to accomplish this, but such algorithms tend to be relatively complex and of little practical use. Such algorithms are not examined in this report. Detection algorithms for the L^W are incorporated into the language specification. By adjusting T in this specification, it is possible to include in the language chains with small amounts of pattern distortion or "noise."

The next two sections examine some particular pattern languages and place them in the hierarchy of language and automata classes given in Section 2.3. Section 3.1 treats the language consisting of chains of straight lines of arbitrary slope. Section 3.2 examines languages formed from chains of circles and circular arcs. An attempt to generalize some of the results of these two sections is given in Section 3.3.

3.1 Straight Lines

Consider the family of curves specified by $[f,R,Q]_{ST}$ where

$$f(x,y) = y - ax$$

$$R = r_1 \begin{cases} (\underline{x}_1,\underline{y}_1) = (-\infty,-\infty) \\ (\bar{x}_1,\bar{y}_1) = (+\infty,+\infty) \end{cases}$$

$$Q = (\underline{q}_1,\bar{q}_1) \begin{matrix} \underline{q}_1 = a \\ = (-\infty,+\infty) \end{matrix}$$

The family of curves described is the family of straight lines of arbitrary length and slope.

The chain code does not store information as to the location with respect to the origin of the encoded curves. In the above, for convenience, $f(x,y)$ has been selected such that all straight lines pass through the origin. The languages based on $[f,R,Q]_{ST}$ include chains of straight lines at all locations in the coordinate system. If necessary, a chain can be preceded by a pair of coordinates giving the starting point of the chain. Location restrictions can then be incorporated into R and Q.

Let L^{s}_{ST} and L^{w}_{ST} be straight-line languages based on $[f,R,Q]_{ST}$, defined according to the strict and weak definitions, respectively. L^{s}_{ST} and L^{w}_{ST} will be related to the hierarchy of string language and automata classes. An RQ-algorithm for the detection of L^{w}_{ST} is as follows.

Let a chain, C, be denoted by

$$C = c_1 c_2 c_3 \cdots c_i \cdots c_N$$

where the c_i are the octal digits that are the elements of the chain.

Assume the starting point of element c_1 to lie at the origin, (0,0). Denote the coordinates of the tip of element c_i by (x_i,y_i). The slope of the straight line represented by the chain is taken to be $a = y_N/x_N$. The test for language membership is $|x_N y_i - y_N x_i| \leq NT_{ST}$ for all i. For $T_{ST} \geq 1$, $L^S_{ST} \subset L^W_{ST}$; values of the tolerance as small as one square result in the acceptance of all chains that are straight lines according to the strict definition (as well as some other chains not satisfying this definition). The RQ-algorithm operates as follows.

First pass over chain:

Compute (i) $x_N = \sum_{i=1}^{N} \gamma_x(c_i)$

(ii) $y_N = \sum_{i=1}^{N} \gamma_y(c_i)$

(iii) NT_{ST} (compute by adding T_{ST} N times)

where $\gamma_x(c_i)$ and $\gamma_y(c_i)$ are given by

c_i	γ_x	γ_y
0	1	0
1	1	1
2	0	1
3	-1	1
4	-1	0
5	-1	-1
6	0	-1
7	1	-1

Second pass over chain:

For each successive i

(i) Find $x_N y_i = x_N y_{i-1} + x_N \gamma_y(c_i)$

(ii) Find $y_N x_i = y_N x_{i-1} + y_N \gamma_x(c_i)$

(iii) Test $|x_N y_i - y_N x_i| \leq NT_{ST}$

If for any i the above test fails, then the chain being tested is immediately rejected. If the test is satisfied for all i, then the chain is accepted.

THEOREM 10. L_{ST}^{W} is context-sensitive and can be detected by a deterministic LBA.

Proof: The program and all fixed storage occupy the control unit of the LBA. The tape of the LBA is used for the storage of quantities whose magnitude increases with increasing chain length. The program requires subroutines for the addition, subtraction and comparison of numbers of arbitrary size appearing on the tape. These subroutines can be programmed in the finite control unit, but may require working storage for quantities not exceeding the magnitude of the numbers calculated or compared. The tape of the LBA is used for this working storage. The largest number that must be stored in the first pass is $\leq NT_{ST}$. The largest number that must be stored during the second pass is $\leq 2N^{3}$. Consequently, the storage requirements of Theorem 2 are satisfied and L_{ST}^{W} can be detected by a deterministic LBA. It follows from Section 2.3 that L_{ST}^{W} is context-sensitive.

The next theorem sets a lower bound for L_{ST}^{S} and L_{ST}^{W} in the language-automata hierarchy.

THEOREM 11. L_{ST}^{S} and L_{ST}^{W} are not context-free languages and cannot be detected by PDSA.

Proof: In the proof of this theorem and a number of others a special form of context-free grammar, G, defined in the following lemma is used.

Lemma 1. (Chomsky)[6] If L is a context-free language, then there exists a context-free grammar, G, of L with the following properties:

(i) For each A in the nonterminal vocabulary of $\underline{G}$ there are infinitely many z's such that $A \Rightarrow z$.

(ii) Let η be the length of the longest string, u, such that $A \rightarrow u$ is a rule of $\underline{G}$. If $v \epsilon L$ then there is a substring, v_1, of η or less symbols of v and an $A \epsilon V_N$ of $\underline{G}$ such that $A \rightarrow v_1$. Replacement of v_1 by any of the infinite number of strings, z, such that $A \Rightarrow z$ does not alter the language membership of v in L.

Proof of Lemma: Let G be a context-free grammar of L, and let A be an element of the nonterminal vocabulary of G that does not generate an infinite number of z. Then A can be eliminated from G in favor of a finite number of rules of the form $B \rightarrow \psi_1 x \psi_2$ whenever G contains the rule $B \rightarrow \psi_1 A \psi_2$ and $A \Rightarrow x$. Carrying out this procedure for all such A the context-free grammar, $\underline{G}$, is obtained satisfying property (i). There must be at least one nonterminal symbol, A, in the next-to-last line of the derivation of every terminal string, $w \epsilon L$. The properties stated in (ii) automatically follow for a context-free grammar.

Now assume that L^s_{ST} and L^w_{ST} are context-free languages. Then there must exist context-free grammars, $\underline{G}^s_{ST}$ and $\underline{G}^w_{ST}$, of the form described in Lemma 1 for these languages. Let $C^s_{ST} \epsilon L^s_{ST}$ and $C^w_{ST} \epsilon L^w_{ST}$ be arbitrary chains of their respective languages. It is possible to choose C^s_{ST} and C^w_{ST} sufficiently long that the slope of the straight line represented by either is defined to any arbitrary degree of precision, even if any substring of length η of either chain is not specified. (For L^w_{ST} the length required depends on T_{ST}.) By Lemma 1 there must exist substrings of η or less elements in C^s_{ST} and C^w_{ST} that can be replaced by an infinite number of strings generated by a single element, A^s and A^w, in $\underline{G}^s_{ST}$ and $\underline{G}^w_{ST}$,

respectively. These replacement strings must themselves be chains of straight-line segments, and since there are an infinite number of them, some must be sufficiently long to define the slope of the straight line represented to any arbitrary degree of precision. It is not difficult to see that as original chains and replacement sections are taken longer and longer, their slopes must approach each other if the resultant chain is to be that of a straight line. Since the number of possible slopes for straight lines is infinite, and since there exist arbitrarily long C^{s}_{ST} and C^{w}_{ST} for straight lines of every slope, there must be an infinite number of nonterminal elements, A, in the grammars of both G^{s}_{ST} and G^{w}_{ST}; one A corresponding to each possible slope. A phrase-structure grammar, however, cannot have an infinite nonterminal vocabulary. Hence there do not exist context-free grammars of L^{s}_{ST} or L^{w}_{ST}. The second part of the theorem follows from Section 2.3.

For L^{w}_{ST}, the proof is dependent on the notion of tolerance rather than a particular RQ-algorithm. Placing restrictions on the slope, a, in defining L^{s}_{ST} or L^{w}_{ST} does not affect this result as long as the number of values of slope permitted remains infinite. Theorems 10 and 11 establish the position of L^{w}_{ST} in the language-automata hierarchy and set a lower limit for the language class of L^{s}_{ST}.

3.2 Circles and Circular Arcs

This section considers some languages consisting of chains of circles and circular arcs. The family of curves is described by $[f,R,Q]_{CI}$, where

$$f(x,y) = (x-a_x)^2 + (y-a_y)^2 - a_r^2$$

$$R = r_1 \begin{cases} (\underline{x}_1, \underline{y}_1) = (-\infty, -\infty) \\ (\overline{x}_1, \overline{y}_1) = (+\infty, +\infty) \end{cases}$$

$$q_1 = a_x^2 + a_y^2$$

$$q_2 = a_r^2$$

$$Q = \begin{cases} (\underline{q}_1, \overline{q}_1) = (a_r^2, a_r^2) \\ (\underline{q}_2, \overline{q}_2) = (0, a_M) \end{cases}$$

No restrictions are made on x or y for the circles. The restriction on q_1 requires that the circles pass through the origin. This is for convenience in detection only since location of curves with respect to the origin is disregarded by the chain code. The restriction on q_2 places a bound, a_M, on the radius of the circles. The language, L_{CI}^B, consisting of chains of circles and circular arcs of bounded radius (defined according to either the strict or weak definition) is considered first. Then the effects of loosening the bound on radius are considered.

A. Bounded Radius ($a_r \leq a_M$)

THEOREM 12. L_{CI}^B can be detected by a K-limited automaton.

Proof: Chains that trace a closed curve more than once are called repetitive. Chains that trace a closed curve only once or do not trace it completely are called nonrepetitive. The length of nonrepetitive chains of circles with $a_r \leq a_M$ is bounded by a bound, M_r, that is a function of a_M. A finite list can be made of all such nonrepetitive chains and a K-limited automaton with $K = M_r - 1$ can be used for their detection. In defining

L_{CI}^{B} the bound on circle radius is retained, but the set of chains is extended to include repetitive as well as nonrepetitive chains. The number of chains in L_{CI}^{B} is infinite. It can be seen that a chain is in L_{CI}^{B} if and only if all sections of this chain of length M_r describe circles. A K-limited automata with $K = M_r - 1$ similar to that used for the detection of nonrepetitive chains can be used to detect L_{CI}^{B}.

The reasoning in the foregoing is not dependent on a particular geometric figure but rather on the boundedness of the length of the line describing the figure. Hence Theorem 12 can be generalized to the following.

THEOREM 13. If the nonrepetitive chains of a class of closed curves are all of length $\leq M$, then the language that consists of the repetitive and nonrepetitive chains of this class of curves can be detected by a K-limited automaton with K = M-1.

B. Unbounded Radius ($0 \leq a_r \leq \infty$)

Let L_{CI}^{S} and L_{CI}^{W} be languages defined according to the strict and weak definitions, respectively, consisting of the chains of circles and circular arcs of unbounded radius. A preliminary RQ-algorithm, Γ_ρ, for the detection of L_{CI}^{W} is now given. It is later shown that this algorithm needs modification to take care of the case where $a_r = \infty$ (straight line). The chain is assumed to begin at (0,0). (x_i, y_i) are the coordinates of the tip of chain element c_i. The first step of Γ_ρ is to determine whether the input chain (assuming that the curve that it represents is a circle) represents a full half circle. This is accomplished by determining if there is a value of $i \leq N$ such that $(x_i^2+y_i^2) < (x_{i-1}^2+y_{i-1}^2)$. If such

a value exists, then (0,0) and (x_{i-1},y_{i-1}) are taken to be the coordinates of the extreme points of the diameter of the circle. If such a value of i does not exist, then it is assumed that the input chain does not represent a full half-circle. The centerpoints, a_x and a_y, and radius, a_r, are then calculated by substituting (0,0), (x_N,y_N), and the coordinates of the midpoint of the chain, in $f(x,y) = 0$. The last step of Γ_ρ is a test to determine whether the input chain lies within T_{CI} squares of a circle with the calculated centerpoint and radius. If this test is satisfied, then the chain is accepted.

Some problems are apparent concerning the detection of L_{CI}^{W} with this algorithm. For any m in Theorem 2, for a sufficiently short chain of a circle of sufficiently large radius, $a_r > m^{\ell(C)}$. For the RQ algorithm given, a LBA does not have sufficient storage for the detection of L_{CI}^{W}. This result is not general; there may exist other RQ-algorithms for the detection of L_{CI}^{W} that can be programmed on an LBA (although the author has not been able to find any). For the case where a_r is infinite, even a Turing machine cannot be used unless a modification of Γ_ρ is made. The modification consists of combining the RQ-algorithm for straight line detection with Γ_ρ. The straight-line algorithm examines an input chain first. If the input chain is that of a straight line, then the chain is accepted and the procedure is halted. If the input chain is not that of a straight line, then Γ_ρ examines the chain. Γ_ρ is modified slightly so that when three points on the chain are to be used to calculate a_x, a_y, and a_r, a check is made first that these three points do not lie in a straight line. If they do, the input chain is rejected. The resulting

combined algorithm is the final RQ-algorithm used to define L^{W}_{CI}. This algorithm provides an answer in finite time as to whether or not any given chain is in L^{W}_{CI}. Hence the following theorem is obtained:

THEOREM 14. L^{W}_{CI} is a recursive language.

If a restriction is made that chains ϵL^{W}_{CI} must represent at least some fixed fraction of the total length of arc of a circle, the detection can be accomplished by a deterministic LBA and the restricted language is context-sensitive. The following theorem sets a lower bound on the language class membership of L^{S}_{CI} and L^{W}_{CI}.

THEOREM 15. L^{S}_{CI} and L^{W}_{CI} are not context-free languages and cannot be detected by PDSA.

Proof: Assume that L^{S}_{CI} and L^{W}_{CI} are context-free. Then there exist context-free grammars, G^{S}_{CI} and G^{W}_{CI}, of the form described in Lemma 1 for these languages. Let $\bar{C} = C_1, C_2, \ldots C_j \ldots$ be a set of nonrepetitive chains, each representing a complete circle with $\ell(C_1) \geq 2$ and the diameter of each succeeding chain $[4T_{CI}]$ squares greater than that of the preceding chain. There are an infinite number of chains in $\bar{C}$ and all of these chains are in both L^{S}_{CI} and L^{W}_{CI}. (It is assumed that T_{CI} is large enough so that $L^{S}_{CI} \subset L^{W}_{CI}$.) Now from Lemma 1, [illegible] or less elements in any $C_j \in \bar{C}$ can be replaced by an infinite number of chains all generated by some A_j in the nonterminal alphabet of G^{S}_{CI} or G^{W}_{CI} without destroying the language membership of C_j. The A_j for the various C_j must all be different since the C_j have different diameters. There must, therefore, be an infinite number of distinct A_j. Since a phrase structure grammar cannot

have an infinite number of nonterminal symbols, it must be concluded that there do not exist context-free grammars of L_{CI}^{S} or L_{CI}^{W}. The second part of the theorem follows from Section 2.3.

3.3 Other Curves

Although some generalizations can be made concerning chain pattern languages based on curves described by [f,R,Q], for the most part each new language must be examined by itself to determine its characteristics and position in the language-automata hierarchy. Theorems 9 and 13 are general in scope and do not depend on any particular [f,R,Q]. The detection of languages defined according to the strict definition and not satisfying the conditions of Theorems 9 or 13 will generally be difficult. It should be possible to construct proofs similar to those for Theorems 11 and 15 to prove that other languages are not context-free. Restrictions on the range are not used for either L_{ST} or L_{CI} but in general these may prove useful for language definition.

IV. PATTERN LANGUAGES BASED ON PATTERN PROPERTIES

Given a set $(p_1, p_2, \ldots p_n)$ of pattern properties, any of which may or may not be ascribed to a given pattern, $\underline{P}$, the variables $\xi_1, \xi_2, \ldots, \xi_n$, may be assigned as follows:

$\xi_i = 1$ $\quad \underline{P}$ has property p_i

$\xi_i = 0$ $\quad \underline{P}$ does not have property p_i

Let $g(\xi_1, \xi_2, \ldots \xi_n)$ be a Boolean function of its respective arguments. A pattern language, L^g, can be defined on the basis of g and $(p_1, p_2, \ldots p_n)$ as follows:

$\underline{P} \in L^g$ if $g(\xi_1, \xi_2, \ldots \xi_n) = 1$

$\underline{P} \notin L^g$ if $g(\xi_1, \xi_2, \ldots \xi_n) = 0$

By taking $g'(\xi_i) = \xi_i$, a separate language $L_i^{g'}$ can be defined for each individual property p_i. Let $G(L_1^{g'}, L_2^{g'}, \ldots L_n^{g'})$ be the result of replacing ξ_i by $L_i^{g'}$, "or" by set union (U), "and" by set intersection ($\cap$), and "not" by set complement ($\bar{}$), in $g(\xi_1, \xi_2, \ldots \xi_n)$. Then the following relation is obtained.

$$L^g = G(L_1^{g'}, L_2^{g'}, \ldots L_n^{g'})$$

By application of the theorems in Section 2.5 concerning the complement, union and intersection of languages, it is often possible to find the language class of L^g in terms of the language classes of the $L_i^{g'}$. The remainder of this section is devoted to the examination of pattern languages formed on the basis of the properties, closure, self-intersection, convexity

and periodicity. No bound is assumed on chain length, for any of the pattern languages; in cases where such a bound exists, Theorem 9 can be used.

4.1 Closure

A chain is said to be closed if its beginning and endpoint coincide.

THEOREM 16. The language, L_{CL}, consisting of those chains that are closed, is a context-sensitive language and can be detected by a deterministic LBA.

Proof: L_{CL} can be detected by an automaton that assumes an input chain, C, to start at the origin (0,0) and uses two counters to keep track of the coordinates (x_i,y_i) of c_i. If $(x_N,y_N) = (0,0)$, then C is accepted. The largest number that must be stored is N. By Theorem 2 a LBA can be used for detection.

The author has not been able to determine a lower bound on the linguistic class membership of L_{CL}.

4.2 Self Intersection

A chain, C, is said to be self-intersecting if two or more points along the chain coincide.

THEOREM 17. The language, L_{SI}, consisting of those chains that are self intersecting, is context-sensitive and can be detected by a deterministic LBA.

Proof: In testing for self-intersection, the input chain is assumed to begin at the origin (0,0). (x_i,y_i) are the coordinates of chain element

c_i. For each i, j, $1 \leq i < j \leq N$, it is determined whether either of the following two conditions exists.

(i) $(x_i,y_i) = (x_j,y_j)$

(ii) Elements c_i and c_j are diagonal and cross each other.

If either of the above conditions is found, then C is immediately accepted and the process is terminated. This algorithm is easily carried out by an LBA.

The author has not been able to determine a lower bound for the linguistic class membership of L_{SI}.

4.3 Convexity

A chain, C, is said to be convex if it satisfies the following conditions.

(i) It is closed.

(ii) If the first element is removed, then the resulting chain is not self-intersecting.

(iii) For all i, j, $1 \leq i < j \leq N$, no point on the straight line defined by the points (x_i,y_i), (x_j,y_j) is external to the figure described by C.

Two theorems are given that place the language, L_{CO}, consisting of those chains that are convex, in the language-automata hierarchy.

THEOREM 18. L_{CO} is context-sensitive and can be detected by a deterministic LBA.

Proof: The tests for conditions (i) and (ii) are made with deterministic LBA using the algorithms given in Sections 4.1 and 4.2. The test for condition (iii) requires the comparison for all i, j, $1 \leq i < j \leq N$, of

successive nodes of the chain segment, $c_i, c_{i+1} \ldots c_j$, with the straight line $(x_j-x_i)y = (y_j-y_i)x$. The algorithm is easily carried out by a LBA. The automata for the detection of (i), (ii) and (iii) can all be combined into a single LBA using a larger alphabet and control unit.

THEOREM 19. L_{CO} is not a context-free language.

Proof: Assume that L_{CO} is context-free. Then there exists a context-free grammar, G_{CO}, of L_{CO} of the form described in Lemma 1. Let $C \epsilon L_{CO}$ be the chain of a circle with radius, $a_r = 2\eta/\pi$. By Lemma 1, η or less symbols of C can be replaced by an infinite number of chains without destroying the language membership of C in L_{CO}. Since there are an infinite number of replacement chains, some of the replacement chains must be of length greater than $2a_r$. Replacement chains of length greater than $2a_r$, however, cannot yield a convex figure. Therefore, there does not exist a context-free grammar of L_{CO}.

4.4 Periodicity

An arbitrary continuous curve can be described by

$$x = h_x(s) \qquad x_a \leq x \leq x_b$$
$$y = h_y(s) \qquad y_a \leq y \leq y_b$$
$$0 \leq s \leq s_d$$

where h_x and h_y are functions giving the abscissa and ordinate, respectively, of the curve as functions of the distance along the curve, s. x_a and y_a are the coordinates of the point at which the curve begins, corresponding to $s = 0$. x_b and y_b correspond to $s = s_d$ and are the coordinates

of the endpoint of the curve. A curve is said to be <u>two-dimensionally periodic</u>, or simply <u>periodic</u>, if there are constants, ρ_s, $(0 < \rho_s \leq \frac{s_d}{2})$, ρ_x and ρ_y such that

$$\left.\begin{aligned} h_x(s+\rho_s) &= h_x(s) + \rho_x \\ h_y(s+\rho_s) &= h_y(s) + \rho_y \end{aligned}\right\} 0 \leq s \leq s_d-\rho_s$$

It is assumed that ρ_s, ρ_x and ρ_y are the minimum values such that the above relations hold. ρ_x and ρ_y are called the <u>abscissa period</u> and <u>ordinate period</u>, respectively.

The notion of periodicity that has been defined is a rather general one. The following special cases are obtained.

(i) If $\rho_y = 0$ but $\rho_x \neq 0$, then the curve is said to be <u>periodic along the x-axis</u>.

(ii) If $\rho_x = 0$ but $\rho_y \neq 0$, then the curve is said to be <u>periodic along the y-axis</u>.

(iii) If $\rho_x = \rho_y = 0$, then the curve is repeated over and over again with increasing s and is said to be repetitive. (Repetitive curves are analogous to repetitive chains.)

A chain can be described by the function

$$h_c(i) = c_i \qquad 1 \leq i \leq N$$

where c_i is the i^{th} element of the chain. A chain is periodic if there exists an integer ρ_c such that

$$h_c(i+\rho_c) = h_c(i) \qquad 1 \leq i \leq N-\rho_c$$
$$1 \leq \rho_c \leq \frac{N}{2}$$

ρ_c is assumed to be the minimum integer such that the above relation holds and is called the <u>period</u> of the chain.

Correspondence between periodicity of chains and (two-dimensional) periodicity of curves will now be discussed. A periodic curve will encode to a periodic chain if and only if there is an integer, t, such that $t\rho_x$ and $t\rho_y$ are both integers. The minimum such t is the ratio of the chain period to the curve period. Of primary interest is the case in which t = 1. In this case the changes in abscissa and ordinate for ρ_c elements of the periodic chain are exactly ρ_x and ρ_y, respectively. Periodic curves for which ρ_x and ρ_y are not rational numbers do not encode to periodic chains. The languages consisting of periodic chains only, and the languages consisting of chains (both periodic and nonperiodic) of periodic curves, are treated separately.

A. Languages Consisting of Periodic Chains

The cases of bounded ρ_c ($\rho_c \leq M$) and nonbounded ρ_c are distinguished.

THEOREM 20. The language, L_P^B, consisting of chains that are periodic with period, $\rho_c \leq M$, can be detected with a K-limited automaton with K = 2M-1.

Proof: If $\rho_c \leq M$, then a chain is periodic if and only if all subsequences of this chain of length 2M are periodic. Since the number of possible periodic sequences of length 2M is finite, these sequences can be listed and incorporated into a finite automaton. 2M-1 past symbols in conjunction with the symbol presently scanned by the automator constitute sufficient information for the automaton to decide whether to reject the chain or to continue testing the remaining symbols of the chain.

Theorems 21 and 22 treat the case where ρ_c is not bounded.

THEOREM 21. The language, L_p, consisting of those chains that are periodic is context-sensitive and can be detected with a deterministic LBA.

Proof: The detection of periodic chains involves the determination of whether there exists an integer, ρ_c, ($\rho_c \leq N/2$) such that for all i, $1 \leq i \leq N-\rho_c$

$$c_i = c_{i+\rho_c} \tag{1}$$

The testing procedure starts with $\rho_c = 1$ and tests successive integer values of ρ_c until it finds one such that (1) holds for all i. If no such ρ_c is found after checking all integer $\rho_c \leq N/2$, then the chain being tested is rejected. This algorithm can easily be programmed on a LBA.

THEOREM 22. L_p is not a context-free language.

Proof: Assume that L_p is context-free. Then there exists a context-free grammar, G_p, of the form described in Lemma 1 for L_p. Consider periodic strings of the form

$$C = \underbrace{ww\ldots w}_{n}$$

where

$$w = \underbrace{abaabaaabaaaab\ldots a\ldots b}_{\rho_c}$$

a and b, $a \neq b$, are chain elements (octal digits). Clearly, there are an infinite number of such w. By Lemma 1 there is a sequence of η', $\eta' \leq \eta$ elements of C that can be replaced by an infinite number of strings, z,

all generated by the same nonterminal symbol $A \epsilon \underline{G}_P$. It can be seen that if $\ell(z) < n\rho_c - \eta'$, then the new chain, like the old chain, must be a periodic repetition of w (The new chain does not necessarily have to consist of an integral number of repetitions of w.) If $2\rho_c \leq \ell(z) < n\rho_c - \eta'$, then z must itself be a periodic repetition of w (although not necessarily containing an integral number of repetitions of w). Thus the Λ that generates replacement strings for C can generate only strings that consist of periodic repetitions of w when $2\rho_c \leq \ell(z) < n\rho_c - \eta'$. Since for any w, n may be chosen arbitrarily large, it can be seen that there must exist a separate $\Lambda \epsilon \underline{G}_P$ for each w. Since the number of different w is infinite, and since a phrase structure grammar must have a finite nonterminal vocabulary, it must be concluded that there does not exist a context-free grammar of L_P.

B. Languages Consisting of Chains of Periodic Curves

Define $L_{P'}$ to be the language consisting of chains of periodic curves. $L_{P'}$ includes nonperiodic as well as periodic chains and its detection is considerably more difficult than the detection of L_P. Two methods for detecting $L_{P'}$ will be examined.

The first method is based on a technique for comparing chains called chain correlation, which was developed by Freeman.[16] Let C and C' be arbitrary chains given by

$$C = c_1 c_2 \cdots c_N$$

$$C' = c'_1 c'_2 \cdots c'_{N'}$$

No loss in generality follows the assumption that $N' \geq N$. The chain

correlation function, $\Phi_{CC'}$, of chains C and C' is defined as

$$\Phi_{CC'}(j) = \frac{1}{N} \sum_{i=1}^{N} \cos(c_i - c'_{i+j}) \frac{\pi}{4} \qquad 0 \leq j \leq N'-N-1$$

$$-1 \leq \Phi_{CC'} \leq 1$$

$\Phi_{CC'}(j)$ is a measure of the mean coherence between the elements of chains C and C'. j is the relative displacement between the elements of the two chains; the i^{th} element of chain C is compared to the $(i+j)^{th}$ element of chain C'.

A chain autocorrelation function, Φ_{CC}, can be formed as follows.

$$\Phi_{CC}(j) = \frac{1}{N-j} \sum_{i=1}^{N-j} \cos(c_i - c_{i+j}) \frac{\pi}{4} \qquad 0 \leq j \leq \frac{N}{2}$$

In this case chain C is compared to a version of itself displaced by j elements. A peak or series of equally spaced peaks at which $\Phi_{CC}(j) \geq D_\Phi$, where D_Φ is some threshold, indicates that C is periodic. (The period of C is the value of j, $j > 0$, at the first peak in D_Φ.) This algorithm can detect chains of arbitrary period and can be executed by a LBA. If the language L_P^B, is defined as consiting of all periodic chains of bounded period ($p_C \leq M$), then by a slight modification of the algorithm just given, L_P^B, can be detected by a finite state device. The summation in the autocorrelation function, $\Phi_{CC}(j)$ is evaluated over intervals of length M. A chain, C, is accepted as being periodic if and only if $\Phi_{CC}(j) \geq D_\Phi$ over all intervals of length M.

Various complications arise in the detection of periodic curves using chain correlation. Identical curves encoded at different positions on a grid usually result in chains that differ somewhat. Such chains often have slightly different numbers of elements. Differences in the number of elements of chains of identical curves cause a difficulty in chain correlation known as displacement error. A more complete description of chain correlation including a discussion of the problem of displacement error and some solutions is given in a separate report.[17,18]

A second method for comparing curves and for detecting periodic chains is called coordinate matching. The coordinate matching function, $\Psi_{CC'}$, of chains C and C' is given by

$$\Psi_{CC'}(j) = \left[(x_i - x_0) - (x'_{i+j} - x'_j)\right]^2 + \left[(y_i - y_0) - (y'_{i+j} - y'_j)\right]^2$$

where (x_0, y_0) and (x'_0, y'_0) are the coordinates of the beginning of chains C and C', and (x_i, y_i) and (x'_i, y'_i) are the coordinates of elements c_i and c'_i, of chains C and C', respectively. This method recognizes a periodic chain by comparing the chain to a displaced version of itself. A chain is accepted as periodic if there is a j, $0 \leq j \leq N/2$, such that

$$\Psi_{CC}(j) = \left[(x_i - x_0) - (x_{i+j} - x_j)\right]^2 + \left[(y_i - y_0) - (y_{i+j} - y_j)\right]^2 \leq D_\Psi$$

for all i, where D_Ψ is some threshold. The required calculations can be carried out for chains of arbitrary period on a LBA. Displacement error problems also exist for coordinate matching methods and can seriously downgrade the results obtained.

The languages $L_{p'}$ and $L_{p'}^{B}$, unlike L_p are dependent on the algorithm used for their detection. The choice of detection method and threshold affect which chains are and are not included in these languages.

V. PATTERN LANGUAGES CONSISTING OF CHAINS THAT ARE SIMILAR IN SOME SENSE TO A GIVEN CHAIN

This section considers pattern languages consisting of chains that are similar in some sense to an arbitrary given chain, C. The chain obtained from a given curve generally depends on the position of the curve with respect to the encoding grid; identical curves can often encode to quite a number of different chains. Differences between chains of identical curves are considered to be manifestations of a form of "noise."[17,18] The strictest notion of chain similarity that can be formed requires the chains involved to be identical. Because of noise, however, the use of this notion of strict similarity is limited. It is desirable to have notions of similarity that require the chains involved to be alike in size, shape and orientation, and yet allow for the presence of noise and slight pattern distortion. Four definitions of similarity that require chains to conform in size, shape and orientation are given.

(i) $\overset{1}{=}$ $C \overset{1}{=} C'$ if and only if C and C' are identical.

(ii) $\overset{2}{=}$ $C \overset{2}{=} C'$ if and only if there exists a curve that can encode to both C and C'.

(iii) $\overset{3}{=}$ (chain correlation - See Section 4.4.) $C \overset{3}{=} C'$ if and only if for some chosen D_Φ

$$\Phi_{CC'} = \frac{1}{N} \sum_{i=1}^{N} \cos(c_i - c_i')\frac{\pi}{4} \geq D_\Phi$$

(iv) $\overset{4}{=}$ (coordinate matching - See Section 4.4) $C \overset{4}{=} C'$ if and only if for some chosen D_4 and all i.

$$\rho_{CC'}(i) = (x_i - x'_i)^2 = (y_i - y'_i)^2 \leq D_4$$

It is assumed that $(x_0, y_0) = (x'_0, y'_0)$

Section 5.1 examines languages consisting of chains that are alike in size, shape and orientation. In Section 5.2 the concepts of similarity are relaxed somewhat.

5.1 Pattern Languages Consisting of Chains Similar in Similar in Size, Shape and Orientation

Given an arbitrary chain, $\underline{C}$, the language $L^j_{\underline{C}}$ can be defined to consist of those chains, C', satisfying $\underline{C} \overset{j}{=} C'$, where $1 \leq j \leq 4$. There are two basic types of automata that can be used for the recognition of such languages. Automata of the first type are constructed about the particular chain, $\underline{C}$. Since $\mathcal{L}(\underline{C})$ is finite and known in advance the chains in $L^j_{\underline{C}}$ can be placed in a finite list. Recognition is accomplished by comparing an input chain to this list. The device is a K-limited automaton with K equal to the length of the longest chain in $L^j_{\underline{C}}$.

Of far more interest are recognition devices of the second type, in which the chain, $\underline{C}$, appears on the input tape preceding the chain to be tested. In this case a single device or program is used for the recognition of $L^j_{\underline{C}}$ for all $\underline{C}$. The remainder of Section V is concerned with devices of this second type.

If the restriction is made that for some bound, M, $\ell(\underline{C}) \leq M$, then for j=1,2,3,4 and all $\underline{C}$ the length of the chains in $L^j_{\underline{C}}$ is bounded. By Theorem 9, $L^j_{\underline{C}}$ is a finite language and can be detected by a K-limited automaton. If no such bound is assumed then the $L^j_{\underline{C}}$ must be examined separately for each j. The language, $L^1_{\underline{C}}$, is the least interesting since it consists of only the single chain, $\underline{C}$. Detection of this language can be accomplished by a PDSA provided that the elements of $\underline{C}$ appear on the input tape in reversed order. The reversed version of $\underline{C}$ is stored in the stack of the automaton and then unstacked element by element for comparison with the chain being tested.

The detection of $L^2_{\underline{C}}$ can be accomplished by superimposing the mazes[15] defined by $\underline{C}$ and the chain being tested. If there exist relative positions of the two mazes that permit a curve to be drawn between them, then the curve will encode to both $\underline{C}$ and the chain being tested. Numerous problems arise in reducing this procedure to an algorithm, a major problem being the necessity for trying all relative positions of the two mazes before some input chains can be rejected. The detection of $L^2_{\underline{C}}$ is not further examined in this report.

$L^3_{\underline{C}}$ can be detected by a PDSA. The chain, $\underline{C}$, appears on the input tape with elements in reversed order. The chain correlation summation is performed over intervals so that the sum obtained does not become too large. The acceptance criterion is that $\Phi_{CC}(j) \geq D_\Phi$ over all the intervals. If chain correlation is not done over intervals then a LBA can be used for the detection of $L^3_{\underline{C}}$. $L^4_{\underline{C}}$ can be detected by a LBA. The automata for the detection of the $L^j_{\underline{C}}$ are summarized in Table I.

	Automata for detection of $L_{\underline{C}}^{J}$ as whole string		Automata for detection of $L_{\underline{C}}^{J}$ as a substring	
Language	$\ell(\underline{C}) \leq M$	$\underline{C}$ not bounded	$\ell(\underline{C}) \leq M$	$\underline{C}$ not bounded
$L_{\underline{C}}^{1}$	K-limited	*PDSA	K-limited	LBA
$L_{\underline{C}}^{2}$	K-limited	-----	K-limited	------
$L_{\underline{C}}^{3}$	K-limited	**PDSA LBA	K-limited	LBA
$L_{\underline{C}}^{4}$	K-limited	LBA	K-limited	LBA

* $\underline{C}$ must appear on input tape with elements in reversed order

** Chain correlation over intervals. $\underline{C}$ must appear on input tape with elements in reversed order.

TABLE I

AUTOMATA FOR THE DETECTION OF $L_{\underline{C}}^{J}$

Consider now the problem of recognizing the chains, C", such that $C' \epsilon L_{\underline{C}}^{j}$ is a substring of C". Some of the general aspects of such recognition have been discussed in Section 2.6. This problem is related to the so called "segment fitting problem" which involves the determination of the point at which a short curve (the "segment") "best" fits a longer curve (the "curve"). The results of the application of chain correlation to this problem (including the effects of noise and displacement error) are given in a separate report.[17, 18] The solution to this problem involves "sliding" the segment along the curve and evaluating the chain correlation coefficient at each point. The point at which the chain correlation coefficient is greatest is considered to be the point at which the segment fits best. If the chain correlation coefficient at this point is below a certain threshold then the segment is considered not to fit the curve at all. The recognition of the chains, C", such that $C' \epsilon L_{\underline{C}}^{j}$ is a substring of C" is accomplished in a similar manner using the index of chain similarity associated with the particular j. The automata for such recognition are given in Table I. As can be seen, the classes of automata used for the detection of whole strings also suffice when the strings appear as substrings of other strings, except in cases where the whole strings would be detected by PDSA. In these cases LBA are now required.

It must be borne in mind that $L_{\underline{C}}^{3}$ and $L_{\underline{C}}^{4}$ are dependent on the thresholds, D_{ξ} and D_{ψ}, respectively. The definitions of similarity for j = 3 and 4 were devised because of the limited usefulness of the similarity definition for j = 1, and the intractability of the definition for j = 2. The similarity definitions for j = 3 and 4 are both affected

by displacement error, although this error can be minimized by modifying the definitions of these forms of similarity somewhat.

5.2 Pattern Languages Based on Other Definitions of Similarity

This section examines the formation of pattern languages when the notion of similarity is viewed in a broader sense than in the previous section. The existence of algorithms for rotating, scaling and performing other operations on chain encoded curves simplifies the detection of such languages. Chain operators can be defined that alter the pattern represented by a chain in some manner and then recode the result using grid-intersect chain encoding.[2] In this manner the following chain operators are defined.

$\tau_r(\theta)$	rotation by angle θ
$\tau_s(k)$	scale change by factor k
$\tau_{s'}(k_x,k_y)$	scale change by factors k_x and k_y in x and y directions, respectively
$\tau_\rho(n)$	periodic repetition n times

The use of an operator τ on a chain C is denoted $\tau \cdot C$. Using the above operators, languages can be defined consisting of the chains, C', such that for an arbitrary given chain, $\underline{C}$, the following relations are satisfied.

(i) $C' \overset{j}{=} \tau_r(\theta) \cdot \underline{C}$ (chains alike except for angle)

(ii) $C' \overset{j}{=} \tau_s(k) \cdot \underline{C}$ (chains alike except for size)

(iii) $C' \overset{j}{=} \tau_{s'}(k_x, k_y) \cdot \underline{C}$ (chains alike except for nonuniform size change in x and y directions)

(iv) $C' \overset{j}{=} \tau_{\rho}(n) \cdot \underline{C}$ (chains alike except in the number of repetitions that appear)

This language specification format is quite flexible. By placing restrictions on the permissible values of θ, k, k_x, k_y, and n, the class of languages that can be defined is made considerably richer. It is possible to combine operators in defining languages. Thus the language defined by $C' \overset{j}{=} \tau_r(\theta) \cdot \tau_s(k) \cdot \underline{C}$ contains chains alike in shape but differing in size and orientation. Additional operators can be defined to augment the class of languages that can be specified.

The linguistic classification of the languages that are defined is dependent on j and the particular operator, τ. Unfortunately the detection of many of these languages is difficult especially when the strings of the languages appear as substrings of larger strings.

VI. OTHER METHODS OF PATTERN ENCODING

Chain coding is a convenient method for putting classes of patterns into string form for linguistic examination. The linguistic results obtained for classes of chain encoded patterns can be extended to the same pattern classes encoded using other methods. This report restricts consideration to pattern codes obtainable from chains using a translating device known as a finite transducer. A finite transducer, T, is a pushdown store device satisfying the additional restriction that the storage tape can only move in one direction. The storage tape is used as an output tape and plays no essential role in determining the course of the computation. The set of strings, v, such that for some $u \in L$, T maps u into v is designated by T(L). A detailed description of finite transducers can be found in the literature.[7,19]

THEOREM 23. (Schützenberger and Chomsky)[7,19] If T is a finite transducer that maps L onto L' (T(L) = L'), then

(i) If L is a regular language, then L' is a regular language.

(ii) If L is a context-free language, then L' is a context-free language.

Therefore, given another method, E, of encoding patterns, if there exists a finite transducer, T, that will translate chains $\in L$ into strings in E, then for regular and context-free languages, the language class membership of T(L) is the same as that of L. Furthermore, let L_E be a pattern language in E. If there exists a finite transducer, $\overset{*}{T}$, that can translate from E into chain code, and if $\overset{*}{T}(L_E) = L'$ is a chain pattern language that has been shown in this report to be not regular, or not context-free, then

the same result holds for L_E.

If the finite transducer translating chain code into E effects a one-to-one mapping then results for context-sensitive languages as well as results for regular and context-free chain languages apply to the corresponding pattern languages in E. Further investigation is needed of the effects of different pattern codes on the pattern languages obtained for various classes of patterns, and of the linguistic aspects of different methods of translating between pattern codes.

VII. CONCLUSIONS

By encoding patterns into string form it is possible to place them in a conventional linguistic framework. The correspondence between the hierarchies of string language and automata classifications can be used to determine the computation power required to detect various classes of patterns. Pattern languages can be placed within the existing structure of string language classifications. Although the patterns considered in this report have been relatively simple ones, it should be possible by means of the language rules of the table-driven pattern analyzer to extend the approach to treat pattern classes containing much more complex patterns.

A great deal of work remains to be done. Only the most important classes of languages and automata were related to patterns. The placement of pattern languages in a more complete language-automata hierarchy would be interesting, as would a linguistic examination of additional pattern classes. The effects of pattern coding on linguistic classification need more study. Further investigations should lead to new methods for pattern recognition and analysis and an increased understanding of patterns in general.

REFERENCES

1. Feder, J. "Linguistic Specification and Analysis of Classes of Patterns," NYU Technical Report No. 400-147, Department of Electrical Engineering, New York University, Bronx, N.Y. 10453, October 1966.

2. Freeman, H. "On the Encoding of Arbitrary Geometric Configurations," IRE Trans. Electron. Comp., EC-10, (2), June 1961, pp. 260-268.

3. Gilbert, P. "On the Syntax of Algorithmic Languages," Journal ACM, 13, (1), January 1966, pp. 90-107.

4. Feder, J. "The Linguistic Approach to Pattern Analysis: Literature Survey," NYU Technical Report No. 400-133, Department of Electrical Engineering, New York University, Bronx, N.Y. 10453, February 1966.

5. Feldman, J.A. "A Formal Semantics for Computer Oriented Languages," Doctoral Thesis, Carnegie Institute of Technology, Pittsburgh, Pennsylvania, 1964.

6. Chomsky, N. "On Certain Formal Properties of Grammars," Inform. and Cont., 2, (1), 1959, pp. 137-167.

7. Chomsky, N. "Formal Properties of Grammars," in Luce, D., Bush, R. and Galanter, E. Handbook of Mathematical Psychology, Vol. 2, Wiley, 1963, pp. 323-418.

8. Rabin, M.O. and Scott, D. "Finite Automata and Their Decision Problems," IBM Journal Res. and Dev., 3, (2), April 1959, pp. 114-125.

9. Myhill, J. "Linear Bounded Automata," WADD Technical Note 60-165, Wright Air Development Division, Wright-Patterson Air Force Base, Ohio, 1960.

10. Kuroda, S.-Y. "Classes of Languages and Linear Bounded Automata," Inform. and Cont., 7, (2), 1964, pp. 207-223.

11. Chomsky, N. and Miller, G.A. "Finite State Languages," Inform. and Cont., 1, (2), 1958, pp. 91-112.

12. Bar-Hillel, Y. Perles, M. and Shamir, E. "On Formal Properties of Simple Phrase Structure Grammars," Technical Report No. 4, Office of Naval Research, Information Systems Branch, July 1960.

13. Landweber, P.S. "Three Theorems on Phrase Structure Grammars of Type 1," Inform. and Cont., 6, (2), 1963, pp. 131-136.

14. Kleene, S.C. "Representation of Events in Nerve Nets and Finite Automata," In Shannon, C.E. and McCarthy, J., Automata Studies, Princeton University Press, Princeton, N.J., 1956, pp. 3-41.

15. Glass, J.M. "A Criterion for the Quantization of Line-Drawing Data," NYU Technical Report No. 400-112, Department of Electrical Engineering, New York University, Bronx, N.Y. 10453, May 1965.

16. Freeman, H. "A Classification and Recognition Technique for Geometric Patterns," Proc. 3rd International Conference on Cybernetics, Namur, Belgium, September 1961.

17. Feder, J. and Freeman, H. "Segment Fitting of Curves in Pattern Analysis Using Chain Correlation," NYU Technical Report No. 400-108, Department of Electrical Engineering, New York University, Bronx, N.Y. 10453, April 1966.

18. Feder, J. and Freeman, H. "Digital Curve Matching Using a Contour Correlation Algorithm," Proc. IEEE International Conv., Part 3, March 1966, pp. 69-85.

19. Schützenberger, M.P. "A Remark on Finite Transducers," Inform. and Cont., 4, (2), 1961, pp. 185-196.

APPENDIX

Theorem 2 A LBA has sufficient storage for the execution of a program if there exist integers, b,m, such that for all inputs, z, the storage of no more than m integers is required, and none of these integers is larger than $b^{\ell(z)} - 1$.

Proof: An m+1 tape automaton will be given that fulfills the necessary storage requirements. It will then be shown that there exists a one-tape LBA that can perform the same computations.

Consider an automaton that has an input tape plus m storage tapes that operates on an alphabet, $\mathcal{A}_0$, containing b symbols. The use of no more than $\ell(z)$ locations on any of the tapes is permitted. The m storage tapes can store m integers any of which may be as large as $b^{\ell(z)}-1$.

To store this information on a single tape, the position of the read-write heads on the m+1 tapes must be stored in addition to the symbols appearing on these tapes. Let $a_{ij} \in A_0$ denote the contents of the j^{th} square of the i^{th} tape. The j^{th} square of the one-tape machine must store a 2(m+1)-tuple of the form

$$(a_{0j}, flag_{0j}, a_{1j}, flag_{1j}, \ldots, a_{mj}, flag_{mj})$$

The flags store the position of the tape head. $Flag_{ij}$ is set if and only if the read-write head of the j^{th} tape is scanning the i^{th} square. If an alphabet containing $b^{m+1} \times 2^{m+1}$ symbols is chosen for the LBA then all tape-machine configurations of the m+1 tape machine can be represented; the LBA need use only $\ell(z)$ squares of its tape. At the expense of a more complex control unit and greater computation time the single-tape LBA can duplicate the computations of the m+1 tape device and fulfill the conditions of the theorem.

Security Classification

DOCUMENT CONTROL DATA - R & D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1. ORIGINATING ACTIVITY *(Corporate author)*
New York University
Dept of Electrical Engineering
University Heights, Bronx, New York 10453

2a. REPORT SECURITY CLASSIFICATION
unclassified

2b. GROUP

3. REPORT TITLE

LANGUAGES, AUTOMATA, AND CLASSES OF CHAIN-ENCODED PATTERNS

4. DESCRIPTIVE NOTES *(Type of report and inclusive dates)*
scientific; interim

5. AUTHOR(S) *(First name, middle initial, last name)*

Jerome Feder

6. REPORT DATE	7a. TOTAL NO. OF PAGES	7b. NO. OF REFS
Aug. 1967	53	19

8a. CONTRACT OR GRANT NO. AF-AFOSR-24-66	9a. ORIGINATOR'S REPORT NUMBER(S) Tech. Rept. 400-165
b. PROJECT NO. 9769-02	
c. 6144501F	9b. OTHER REPORT NO(S) *(Any other numbers that may be assigned this report)* AFOSR-68-0794
d. 681304	

10. DISTRIBUTION STATEMENT

Distribution of this document is unlimited.

11. SUPPLEMENTARY NOTES	12. SPONSORING MILITARY ACTIVITY
TECH OTHER	Air Force Office of Scientific Research Directorate of Information Sciences Arlington, Va. 22209

13. ABSTRACT

By treating patterns as statements in a two-dimensional language, it is possible to apply linguistic theory to pattern analysis and recognition. This report presents an approach to a classification scheme for pattern languages that could provide information about types of programs and computation facilities capable of meeting particular pattern analysis and recognition requirements. Consideration is restricted to line patterns encoded in the chain code developed by Freeman. This encoding method represents a line pattern by a sequence of octal digits called a chain. Results can be extended to other forms of encoding when translators between codes can be built. The report compares languages formed by Boolean functions of languages and by the concatenation of strings of a number of languages with pattern languages. Pattern languages based on families of equations in two variables and formed from chains of straight lines, circles, and circular arcs are related to string language classes. Pattern properties, including closure, self-intersection, convexity, and periodicity, are examined. Pattern languages are also considered that are similar in various ways to an arbitrary given chain. Although the patterns considered in this report are relatively simple ones, it appears possible by means of language rules of a table-driven pattern analyzer to extend the approach to pattern classes containing more complex patterns.

DD FORM 1 NOV 65 1473

Security Classification

14 KEY WORDS	LINK A		LINK B		LINK C	
	ROLE	WT	ROLE	WT	ROLE	WT
Pattern recognition						
Language						
String languages						
Automata						
Pattern classification						
Computational power						
Chain codes						
Codes						
Grammar						
Hierarchies of language classes						