TR-79-003

On Syntactic Pattern Recognition

by

Patrick Shen-pei Wang

Institute of Information Science
Academia Sinica
Taipei, Taiwan 115
R. O. C.

Department of Computer Science
University of Oregon
Eugene, Oregon 97403
U. S. A.

On Syntactic Pattern Recognition

by

Patrick Shen-pei Wang

Institute of Information Science
Academia Sinica
Taipei, Taiwan 115
R. O. C.

Department of Computer Science
University of Oregon
Eugene, Oregon 97403
U. S. A.

Table of Contents

Abstract

This research paper deals with the development of syntactic pattern recognition by computers. A new device for generating formal languages, called " programmed grammar " is studied. In order to broaden its capability for 2-dimensional cases, the so-called "stochastic programmed array grammar" has been presented. Its alternative counterparts such as picture grammars, matrix grammars and fuzzy grammars are also investigated. Another approach from cellular automata point of view which is inherently faster than iterative acceptors is also introduced and is related to array automata. Finally the grammatical inference for discovering the grammars given a class of sampled data of a certain type of languages is studied and is applied to syntactic pattern recognition. The syntactic pattern recognition problems for 2-dimensional cases have been studied recently only in rudimentary exercises. There are still many open questions and formal works in this field need to be developed.

The general goal of the intended research is to develop a theory of syntactic pattern recognition through techniques from:

1. grammatical analysis, and
2. cellular automata analysis.

We intend to establish the following as subgoals:

1. to give a formal definition of " syntactic pattern recognition ",
2. to minimize the recognition (parsing) time with minimum errors,
3. to solve some incompleteness problems of tessellation automata, and
4. from syntactic pattern recognition to semantic pattern recognition.

Chapter 1 Introduction

In the past few years, two general approaches for solving pattern recognition problems have been developed, namely 1. decision-theoretical (or discriminant, geometrical) approach : In this approach a set of characteristic measurements called features are extracted from the patterns; the recognition of each pattern is usually made by partitioning the feature space. Once a pattern is transformed through feature extraction to a point or a vector in the feature space, its characteristics are expressed only by a set of numerical values. The information about the structure of each pattern is either ignored or not explicitly represented in the feature space. Successful applications of this approach to practical problems include character recognition, medical diagnosis and crop classification etc. (41,69).2. syntactic (or linguistic,structural) approach : This approach draws an analog between the structure of patterns and the syntax of languages. Pattern primitives are first selected and their relations in the patterns are described by a set of syntactic rules (or grammars). The recognition process is accomplished by performing a syntactic analysis (or parsing) to the sentence describing the given pattern. Initial applications of the syntactic approach to the recognition of pictorial patterns have given quite promising results. (21, 62).

In this research paper , our interest will be concentrated in the syntactic approach mainly due to the following motivations:

1. In some very practical applications of recognition problems (e.g. picture processing or more generally speaking, scene analysis) in which the structural information to describe each pattern is important,

the decision-theoretical approach has not been very effective and efficient. A typical example is the separation of two patterns which are very closely alike (e.g. the handwritten Latin letter "O" and "Q" or Russian letter "Ш" and "Щ"). It is difficult to divide the regions because they are too closely adjoined to one another and can only be well partitioned on subspace of small dimenionality. The number of features and/or possible descriptions is usually very large making it impractical to regard each (high-dimensional) descriptions as defining a class. However the application of the syntactic approach does not cause any difficulties. For instance, to describe the pattern of letters "O" and "Q", it is sufficient to have such simple concepts as the oval, "intersection","left and right", "up and down".2. Syntactic pattern recognition is an attempt to adapt the techniques of formal language theory, which provide both a notation (grammars) and an analysis mechanism (parsing) for such structures, to the problem of representing and analyzing patterns containing a significant syntactic content. As this apparently includes many kinds of patterns of interest, syntactic pattern analysis has recently become the focus of an increasing amount of pattern recognition research.
3. Compared with the decision-theoretic approach, the syntactic approach has not been as extensively investigated. The mathematical linguistics needed for the syntactic approach is not well developed yet.
4. However there are still some similarities between two approaches.. For instance, the feature extraction and selection problem in the decision-theoretic approach and the primitive selection problem in the syntactic approach are similar in nature.

It should be mentioned here that the terms "linguistic" and "syntactic" are used almost interchangeably herein, as has become common practice in the literature, although the former is a somewhat broader term. Strictly speaking, "syntactic" refers only to the structural aspects of languages; analytical techniques have been considered, both for natural languages (33) and for patterns (20) which may be considered as linguistic but not syntactic.

Formal languages, the generalized phrase-structure grammars and the corresponding automata (recognizers, acceptors) are the main tools for solving syntactic pattern recognition problems. Recent studies of cellular automata are also applied to recognition problems. Up to date, however, the theories and results for 2-dimensional formal language recognition problems are still very basic and only in rudimentary exercises.

This paper is roughly divided into two parts : (1) a general survey of the field (chapters 2-4) (2) a discussion of the area of future research the author intends to pursue (chapters 5-6).

Part I

A General Survey of the Field

Chapter 2 Stochastic Programmed Array Grammars

This chapter is concerned with the syntactic pattern recognition by stochastic programmed array grammars. Array grammars (§ 2,1) deal with two-dimensional formal languages. Once they are in the programmed form (§ 2,2), the grammars can be simplified and their generating abilities are widely broadened. The proposed stochastic grammars (§ 2,3) count in the unavoidable distortion and noise considerations in the real life, hence largely enhance the capabilities of solving syntactic pattern recognition problems (§ 2,4) in practical use.

§ 2,1 Array Grammars and Picture Grammars

In dealing with the generation of two-dimensional pattern (languages) array grammars and picture grammars [12 , 16 , 37 , 52] are widely used and shown to be very useful. There are several ways to generalize phrase-structure grammars whose rewriting rules allow the replacement of subarray of a picture with another subarray. Kirsch [33] developed a method of doing this and a language of right triangles is studied. A similar formal system is developed by Dacey [16,17] and grammars for languages consisting of classes of polygons are exhibited. A survey of this area of picture languages is given by Miller and Shaw [37].

Array grammars can be thought of as the two-dimensional generalization of context-sensitive grammars [27,30]. The formal definition is given as follows:

Dfn 2.1.1 An array grammar is a quintuple $G = (V, V_T, P, \#, S)$

where V : non-emty finite set of symbols called vocabulary

$V_T \subset V$: non;empty finite set of terminals

$\# \in V-V_T$: blank symbol

$S \in V-V_T$: start symbol

P : a non-empty finite set of structure-preserving productions or rewriting rules

The productions of P are of the form $\alpha \rightarrow \beta$ defined as follows :

Let J be a finite connected[+] subset of I^2 where I is the set of integers. Then α and β are mappings from J into V with the restriction that if $\alpha(i,j) = a \in V_T$ then $\beta(i,j) = a$ (i.e. terminals are never rewritten).

An array A is a mapping $I^2 \xrightarrow{\text{into}} V$. A production $\alpha \rightarrow \beta$ is applicable to A if there exists a translation τ of the domain J of α , such that $A \mid \tau J = \alpha$. Array A' is directly derivable from A, written as $A \Longrightarrow A'$, if for $\alpha \rightarrow \beta$ applicable to A, $A' \mid \tau J = \beta$ and $A' \mid (I^2 - \tau J) = A \mid (I^2 - \tau J)$. Let $\overset{*}{\Longrightarrow}$ be the transitive closure of $\Longrightarrow$. Then for $A \overset{*}{\Longrightarrow} B$, B is said to be derivable from A.

An initial array A_S is a mappint $I^2 \xrightarrow{\text{onto}} \{\#, S\}$ s.th. $\{ (i,j) \mid A_S(i,j) = S \}$ is a singletion. A terminal array A_T is a mapping $I^2 \xrightarrow{\text{into}} \{\#\} \cup V_T$ such that $\{(i,j) \mid A_T(i,j) \in V_T\}$ is connected.

The array language generated by an array grammar G is denoted by $L(G) = \{ B \mid A \overset{*}{\Longrightarrow} B \text{ where } A \in A_S \text{ and } B \in A_T \}$.

+ By connected we mean rookwise-connected, i.e. points (i,j) and (i',j') are connected iff $|i - i'| + |j - j'| \leq 1$. A subset K of I^2 is connected iff for any two points p and q in K , there is a sequence of points $P_1, P_2, \ldots, P_n$ in K with $p_1 = P$, $P_n = q$ and P_i connected to P_{i+1}, $1 \leq i < n$.

An array grammar G is said to be montonic if it cannot erase, i.e. if for arbitrary productions $\alpha \rightarrow \beta$, $\beta(i,j) = \#$ implies $\alpha(i,j) = \#$. In this case, L(G) is called a monotonic array language (or alternatively isotonic array language although montonic and isotonic have different senes in string grammars [36]).

The first example of an array grammar is described by Kirsch [33] for generating a class of labeled 45 right triangles. Then it is generalized by Dacey [16,17] to form polygons. The syntactic structure of these languages is analyzed and it is shown that a mathematical group summarizes the structure holding between languages constructed for polygons that are related by proper and improper rotations [17].

A possible modification to the definition of a derivation in a grammar (either a string or array grammar) is parallel rule application, i.e. all instances of the rule's antecedent are simultaneously replaced by the consequent (rather than just one instance). It should be emphasized that it is often useful to allow parallel application of rules in that, for a given language, a grammar that operates in parallel may be much simpler to write than one that operates sequentially. This modification is especially natural for array grammars since local picture operations (essentially equivalent to applying sets of context-sensitive productions) are often applied to digital picture in paralle l. The following simple illustration will describe the situation.

Exmaple 2.1.1. Suppose $G = (V, V_T, P, \#, S)$ be an array grammar

where $V = \{S, T, 1, \#\}$ $V_T = \{1\}$

and the productions in P are

(1) $\begin{matrix} \# & \\ S & \# \end{matrix} \rightarrow \begin{matrix} S & \\ 1 & T \end{matrix}$ (3) $\begin{matrix} T \\ \# \end{matrix} \rightarrow \begin{matrix} 1 \\ \# \end{matrix}$

(2) $S \rightarrow 1$ (4) $\begin{matrix} T & \\ 1 & \# \end{matrix} \rightarrow \begin{matrix} 1 & \\ 1 & T \end{matrix}$

Here L(G) is the set of all isosceles right triangles composed of 1's in a field of #'s. The triangles are oriented so that the right angle is at the lower left-hand corner of the array. The degenerate triangle consisting of a single 1 is also in L(G). One of the derivation results would look like :

```
                                                                 1
                S       S                                        1 1
   (1)S     (1)1  T (3)1  T (4)    (1)(4)(2)(3)(4)    (3)        1 1 1
 S ⇒ 1    T⇒1   T ⇒1   1 ⇒       ...............     ⇒        1 1 1 1
```

where the #'s are omitted. The operation is sequential, i.e. only one instance of the rules antecedent is replaced by the consequents in each step. Consider another simpler parallel grammar $G' = (\{S,1,\#\}, \{1\}, P, \#, S)$ where $P = \left\{ (1)\ \begin{matrix} \# \\ S\ \# \end{matrix} \rightarrow \begin{matrix} S \\ 1\ S \end{matrix}\ ,\quad (2)\ S \rightarrow 1 \right\}$

It is clear that $L(G') = L(G)$ i.e. $G \approx G'$ (grammar G is equivalent to G'). However to derive sequentially a triangle whose sides have length n an $n(n+1)/2$ -step derivation is required while to derive parallely only n-step derivation is required. It's obvious that the parallel grammar is much more simpler and the parallel processing is faster.

It should be very careful when parallel processing is applied. If instances of the antecedent cannot overlap, such aprallel application can lead only to sentential forms that are derivalbe by a succession of one-instance application. On the other hand, if instances can overlap, the notion of parallel application require further clarification [52]. In general, the languages generated by a given grammar when its rules are applied in parallel need not be the same as the language when they are applied sequentially [51]. Furthermore, the language parsed in aprallel by the grammar can be different from both of these. However it is shown [51] that any language is a parallel language and vice versa.

In addition to the array grammars there are still many other micellaneous methods for two-dimensional "pictures" [8,16,17,19,33,36,37, 39,46,51,52,53,56,60]. The following section is a brief discussion about picture-processing grammars " based upon Chang [12,13].

Picture processing grammar can be regarded as an extension of phrase-structure grammars to the two-dimensional case. The structure of pictures is usually hierachical, i.e. in a picture grammar higher level syntactic categories. For example, a line drawing is defined in terms of lines, and lines are in turn defined in terms of, say, points. Therefore, for line drawings, rules transforming points into lines can always be applied before rules transforming lines into more complicated patterns are applied.

The motivation for choosing this model-picture processing grammar is presented as follows. Consider the following context-free grammar as a model to describe horizontal lines :

(1) h-ds $\longrightarrow$ pt pt	(2) h-ln $\longrightarrow$ h-ds pt
(3) h-ln $\longrightarrow$ pt h-ds	(4) h-ln $\longrightarrow$ h-ds h-ds
(5) h-ln $\longrightarrow$ h-ln pt	(6) h-ln $\longrightarrow$ pt h-ln
h-ln $\longrightarrow$ h-ln h-ds	h-ln $\longrightarrow$ h-ds h-ln
h-ln $\longrightarrow$ h-ln h-ln	

where h-ds : horizontal dash, h-ln : horizontal line, pt : point

An element of a picture is a symbol together with its associate vector, and a picture is a collection of (symbol, asociate vector) pairs. In the above example we can associate (x,y,1) to the symbol "pt". (x,y) specifies the position of the " pt " on the grid. The third parameter is the number of squares occupied by the picture element "pt". Similarly (x,y,1) is associated with the symbol h-ln, where l can be regarded as the length of the h-ln and (x,y) is the coordinate of the left most element position. For instance Fig 2.1 represents the set $V = \{$ pt (2,1,1), h-ds (3,5,2) h-ln (2,3,4) $\}$

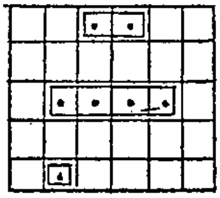


Fig 2.1 horizontal lines

For convenience, we'll write V as V = pt (2,1,1) $\wedge$ h-ds (3,5,2) $\wedge$ h-ln (2,3,4). It can also be represented in terms of points alone :

U = pt (2,1,1) $\wedge$ pt (2,3,1) $\wedge$ pt (3,3,1) $\wedge$ ... $\wedge$ pt (4,5,1)

we would like to reduce U to V via the following grammar :

(1') h-ds (x,y,z) $\rightarrow$ pt (x,y,1) $\wedge$ pt (x + 1,y,1)

(2') h-ln (x,y,3) $\rightarrow$ h-ds (x,y,2) $\wedge$ pt (x + 2,y,1)

(3') h-ln (x,y,3) $\rightarrow$ pt (x,y,1) $\wedge$ h-ds (x + 1,y,2)

(4') h-ln (x,y,δ +4) $\rightarrow$ h-ds (x,y,z) $\wedge$ h-ds (x + δ + 2,y,2)

(5') h-ln (x,y,l + 1) $\rightarrow$ h-ln (x,y,l) $\wedge$ pt (x + 1,y,1)

h-ln (x,y,l + δ +2) $\rightarrow$ h-ln (x,y,l) $\wedge$ h-ds (x + 1 + δ ,y,2)

h-ln (x,y,$l_1 + l_2 + \delta$) $\rightarrow$ h-ln (x,y,l_1) $\wedge$ h-ln (x + δ + l_1,y,l_2)

(6') h-ln (x,y,l + 1) $\rightarrow$ pt (x,y,1) $\wedge$ h-ln (x + 1,y,1)

h-ln (x,y,l + δ +2) $\rightarrow$ h-ds (x,y,2) $\wedge$ h-ln (x + δ +2,y,1)

where $x,y > 0$, $l_1,l_2,l > 2$, δ are variable (δ will be explained later.)

Let $W = pt(2,3,1) \wedge pt(3,3,1) \wedge pt(4,3,1) \wedge pt(5,3,1)$

$$\overset{(1')}{\Rightarrow} pt(2,3,1) \wedge pt(3,3,1) \wedge h\text{-}ds(4,3,2)$$

$$\overset{(1')}{\Rightarrow} h\text{-}ds(2,3,2) \wedge h\text{-}ds(4,3,2)$$

$$\overset{(4')}{\Rightarrow} h\text{-}ln(2,3,4)$$

Explanation of δ : if $\delta = 0$, the grammar describes only perfect h-ln's. If $\delta \neq 0$ (e.g. $\delta = 1$), then the imperfect line as shown in Fig2,2 can be recognized as h-lines. δ is a parameter which controls the gap width between lines. With a nonzero δ, gaps between lines can be filled.

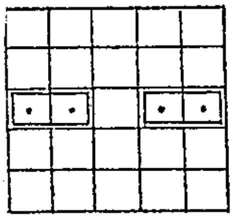


Fig 2.2 a noisy line

We are now ready to give a formal definition of picture processing grammars. Dfn 2.1.2 A picture processing grammar G is a quintuple (S,V,C,g,P) where S is the set of basic symbols, V is the set of vocabulary symbols, $C \subseteq V$ is the set of categorical symbols, g is a function from VUS into the set of natural numbers, and P is the set of grammar rules. Each rule in P is of the form $\alpha(f(x_1, x_2, \ldots, x_k)) \rightarrow \beta_1(x_1) \wedge \beta_2(x_2) \wedge \ldots \wedge \beta_k(x_k)$ where $\beta_1, \beta_2, \ldots, \beta_k \in V \cup S$, $\alpha \in V$ the number of parameters of the associate vector x_i is equal to $g(\beta_i)$, $1 \leq i \leq k$, and f is a partially computable function from $I^{\sum_{i=1}^{k \geq 1} g(\beta_i)}$ into $I^{g(\alpha)}$, whose completion is also computable. A G-picture is a finite set of symbol-associate vector pairs $\xi_1(x_1) \wedge \cdots \wedge \xi_n(x_n)$ such that (a) $\xi_i \in S$, $1 \leq i \leq n$ and (b) $x_i \in I^{g(\xi_i)}$, $1 \leq i \leq n$.

Dfn 2.1.3 A picture processing grammar $G = (S,V,C,g,P)$ is called hierachical iff there exists a nontrivial partition of the rules P into blocks $R_1, R_2, \ldots, R_n$, $n > 1$, s.th. if α appears as the left-hand symbol of a rule in R_i, then it will never appear as a right-hand symbol of any rule in R_j, provided that $j < i$.

Dfn 2.1.4 Given two grammars $G_1 = (S_1, V_1, C_1, g_1, P_1)$ and $G_2 = (S_2, V_2, C_2, g_2, P_2)$. Suppose (a) $S_2 \subseteq V_1 \cup S_1$, (b) $V_2 \cap (V_1 \cup S_1) = \emptyset$, and (c) $g_1(\alpha) = g_2(\alpha)$ if $\alpha \in S_2$, then the composition of G_1 and G_2, denoted by $G_1 \circ G_2$, is the grammar $(S_1, V_1 \cup V_2, C_2, g, P_1 \cup P_2)$, where g is defined by

$$g(\alpha) = \begin{cases} g_1(\alpha) & \text{if } \alpha \in V_1 \cup S_1 \\ g_2(\alpha) & \text{if } \alpha \in V_2 \end{cases}$$

Some theorems and important results will be described below [13] :

Theorem 2.1.1 If G is a hierachical picture processing grammar, then in the reduction of a G-picture, rules in block R_j can always be applied before rules in block R_i are applied, provided that $j < i$.

Theorem 2.1.2 If $G = (\ldots((G_1 \circ G_2) \ldots G_n) \ldots)$, $n > 1$, and $S_1, V_1, \ldots, V_n$ are pairwise disjoint, then G is hierachical. The converse is also true.

This theorem shows that the composition of several grammars results in a hierachical grammar. Conversely, given a hierachical grammar, one can decompose it into several grammars s.th. their composition is equivalent to the original grammar. In practice, we can design several pieces of grammars and then form their composition. The resulting grammar is of course hierachical. With such a hierachical grammar the picture analyzer can then process efficiently.

Theorem 2.1.3 There is a constructive procedure to decide whether a picture processing grammar is hierachical.

To summarize, we have shown that properly designed picture processing grammars can be used to (a) recognize pictures (b) generate or synthesize pictures (c) process or transform pictures and (d) perform inverse picture transformations. In some cases the grammar is able to handle noisy pictures.

§2.2. Programmed Grammars and Matrix Grammars

In 1965 Abraham [2,3] first introduced a new type of generative grammar called matrix grammar which can be described as follows :

Dfn 2.2.1 A matrix grammar is a quintaple $G = (V, V_T, \Sigma, F, F^*)$ where V is a finite set of symbols (letters) called dictionary, V_t is a proper subset of V called terminals. Σ is a finite set of sequences over V called initial sequences. F is a finite set of pairs (φ, ψ), where φ and ψ are sequences over V with restrict that $\varphi \in V - V_T$ and $|\varphi| = 1$. F* is a finite set of matrices (called matrix rules), defined as follows :

(i) f* is a matrix rule if it has the form

$$\begin{pmatrix} f_1 \\ \vdots \\ f_n \end{pmatrix}$$

(ii) f* is a matrix rule if it has the form

$$\begin{bmatrix} f_1^* \\ \vdots \\ f_n^* \end{bmatrix}$$

with $f_i \in F$ (not necessarily $f_i \neq f_j$) where f_i^*'s are matrix rules or $\in F$

To apply a matrix rule a f* to a string x means to apply all the context free rules which form it, to x, in the given order. The generative power of a matrix grammar can be shown in the following example :

Example 2.2.1 Consider the context-sensitive (type 1) language $L = \left\{ a^n b^n c^n \mid n \geq 1 \right\}$ we have the following matric grammar that generates it : $G = (V, V_T, \sum, F, F^*)$ with

$V = \left\{ S,X,Y,Z,a,b,c \right\}$, $V_T = \left\{ a,b,c \right\}$, $F^* : \left[S,abc \right] \left[S,aXbYcZ \right]$

$F = \left\{ (S,abc), (S, aXbYcZ), (X,aX), (Y,bY), (Z,cZ), (X,a), (Y,b), (Z,c) \right\}$ $\begin{bmatrix} X,aX \\ Y,bY \\ c,cZ \end{bmatrix}$ $\begin{bmatrix} X,a \\ Y,b \\ Z,c \end{bmatrix}$

In other words, by properly applying matrix grammars, the context-free production rules can generate context-sensitive languages. The following famous theorem (by Abraham) is very useful.

Theorem 2.2.1 For every given context-sensitive grammar G a strongly equivalent matrix grammar G_M can be constructed.

The generative capability of matrix grammars seem still so quite limited and awkward that intuitively they can be expanded and generalized in certain senses. Peter [54] deals with grammars where the productions are arranged cyclicly, and each production may either be applied once or as many times as possible. Ginsburg and Spair [3] have considered the classes of languages generated from phrase structure grammars by leftmost derivations whose production sequences lie in some language. Chomsky [14] has mentioned a model of natural languages where the grammar contains context-sensitive productions which are applied cyclicly. A group at MITRE [2] has written a program for analyzing English which utilizes productions of this form as part of its grammar. In 1969, Rosenkrantz [54] proposed a new idea of generative grammar called programmed grammars which largely enhance the generative power of context-free production rules. It can be described as follows :

Dfn 2.2.2 A programmed grammar is $G=(V,V_T,P,J,S)$ where V,V_T,P,S are as in §2.1. J is a set of production labels. With each r in J there is associated a unique production (r, φ, ψ, V, W). Here φ and ψ are same as in Dfn 2.2.1. V and W are subsets of J. The production is written in the following format : (r) $\varphi \rightarrow \psi$ S (V) F (W). Note that the format is somewhat similar to the instruction format of the SNOBOL programming language [54] and of Markov normal algorithm [54].

In applying the production to an intermediate string ξ, ξ is first scanned to see if it contains φ as a substring. If so, the leftmost occurrence of φ in ξ is replaced by ψ, and the next production to be applied to the ensuing string is selected from V (called success field). If ξ does not contain φ, then no chnage is made, and the next production is selected from W (called failure field). We'll use the Example 2.2.1 again to show the generative power of a programmed grammar.

Example 2.2.2 Let G be a programmed grammar

$$G = (\{S,B,C,a,b,c\}, \{a,b,c\}, P, \{1,2,3,4,5\}, S)$$

where P : (1) $S \rightarrow a\ B\ S$ (2,3) F (ε)

(2) $B \rightarrow a\ B\ B$ S (2,3) F (ε)

(3) $B \rightarrow C$ S (4) F (5)

(4) $C \rightarrow b\ C$ S (3) F (ε)

(5) $C \rightarrow c$ S (5) F (ε)

Clearly $L(G) = \{a^n b^n c^n \mid n \geq 1\}$ and the cores $(\varphi \rightarrow \psi)$ are all in context-free forms. Compare with Example 2.2.1 we have only 5 production rules which is simpler.

From this example it is known that a major advantage of using programmed grammar is that the grammars can often generate the sentences of a language in a manner which corresponds to the way in which humans would envision the generation. Among the many famous theorems derived by Rosenkrantz[54] I am particularly interested in the following one :

Theorem 2.2.2 The set of languages generated by programmed grammars all of whose rules have cores with a single symbol on the left-hand side and an arbitrary (possibly null) string on the right-hand side is identical to the set of recursively enumerable languages.

In summary, a key result is that programmed grammars whose cores have a single symbol on the left-hand side and an arbitrary string on the right-hand side can generate all recursively enumerable languages. The context-free programmed grammars generate a class of languages which properly contains the context-free languages and is properly contained within the context-sensitive languages. Cfpg's have considerable generative power and it is often comparatively easy to write a cfpg for a particular language. However several problems which are decidable for context-free grammars are undecidable for cfpg's [54].

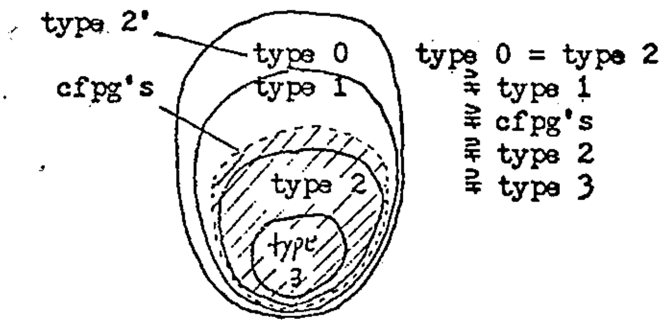


Fig 2.3 Hierachy of pgl

§2.3 Stochastic Grammars and Fuzzy Grammars

Since in syntactic approach, abstract primitive elements are usually selected, effects of noise and distortion in the measurement of patterns can be reduced only through extension preprocessing. From the view point of real-data processing, noise and distortion are, in general, unavoidable in the actual practice. In order to take them into consideration, the use os stochastic languages for pattern description have recently been proposed as a possible solution. This section mainly follows Fu[21,22,23,34,62], Booth[10], Turakainen [65], Paz [45] and Zadeh [72,73,74].

Dfn 2.3.1 A stochastic grammar is a 5-tuple $G_S = (V_N, V_T, P, S, D)$

where V_N : finite set of nonterminals

V_T : finite set of terminals $(V_N \cap V_T = \emptyset)$

P : finite set of productions

$S \in V_N$: start symbol

D : probability measure (assignment) over P

The generating process of a string $x \in L(G_S)$ can be represented as

$$S \stackrel{r_1}{\Longrightarrow} \gamma_1 \stackrel{r_2}{\Longrightarrow} \gamma_2 \quad \cdots \quad \stackrel{r_n}{\Longrightarrow} \gamma_n = x$$

where $r_i \in P$ and $\gamma_i \in (V_N \cup V_T)^*$. The probability associated with the generation of x is defined to be the product of conditional probabilities

$$p(x) = p(r_1)p(r_2|r_1) \cdots p(r_n|r_1, \ldots, r_{n-1})$$

If the string $x \in L(G_S)$ can be generated by m distinct sequences of productions, then the probablity associated with the generation of x is defined as

$$g(x) = \sum_1^m p(x) = \sum_1^m p(r_1)p(r_2|r_1) \cdots p(r_n|r_1, \ldots, r_{n-1})$$

A production probability assignment D is consistent provided

$$\sum_{x \in L(G_S)} g(x) = 1$$

The necessary and sufficient condition for a consistent stochastic context-free language (grammar) has been found by Booth [10] and Grenander [21,22]. But the conditions for a consistent stochastic context-sensitive grammar have yet to be found. Furthermore D is called an unrestricted production probability assignment provided $P(r_j \mid r_1, \ldots, r_{j-1}) = p(r_j)$ for all production sequences. Let p i be the associated probability of a production $\gamma \rightarrow \eta_i$, then a stochastic grammar is said to be normalized iff $\sum_i p_{\gamma_i} = 1$ for all i such that η_i is the consequence and γ is the premise from p. The following exampoes are given to demonstrate the potential for using stochastic languages for the description of distorted and noisy patterns.

Example 2.3.1 An equilateral triangle and eight other distorted versions are shown in Fig 2.4. The pattern primitives selected are given in Fig 2.5.

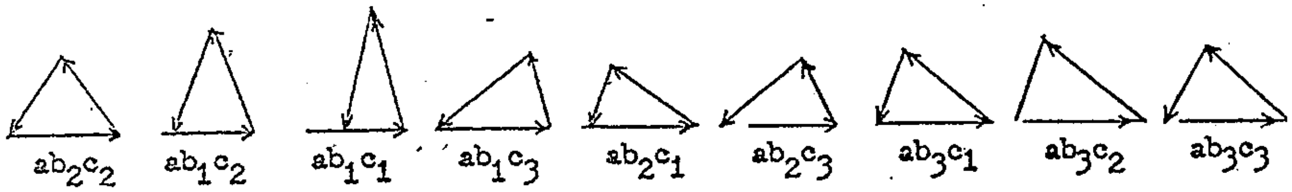


Fig 2.4 noisy triangles

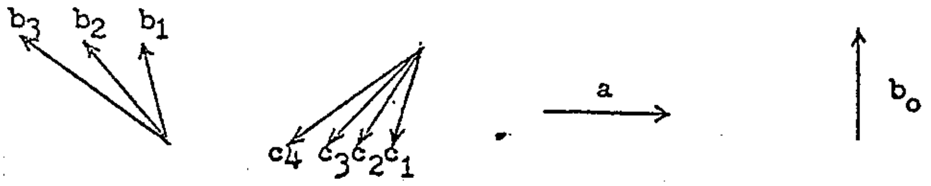


Fig 2.5 primitives

Table 2.1 shows the probabilities of these 9 different triangles. The stochastic finite-state grammar which will generate these strings with associated probabilities is $G_S = (V_N, V_T, P, S, D)$ where

$V_N = \{S, A_1, A_2, A_3, A_4\}$ $V_T = \{a, b_1, b_2, b_3, c_1, c_2, c_3\}$

and

P	D
$S \rightarrow a A_1$	1
$A_1 \rightarrow b_1A_2 \mid b_2A_3 \mid b_3A_4$	1/6,2/3,1/6
$A_2 \rightarrow c_1 \mid c_2 \mid c_3$	1/6,1/3,1/2
$A_3 \rightarrow c_1 \mid c_2 \mid c_3$	1/24,21/24,1/12
$A_4 \rightarrow c_1 \mid c_2 \mid c_3$	1/2,1/3,1/6

Note that here D is consistent, unrestricted and nomalized.

x	p(x)	x	P(x)
ab_1c_1	1/36	ab_2c_3	2/36
ab_1c_2	2/36	ab_3c_1	3/36
ab_1c_3	3/36	ab_3c_2	2/36
ab_2c_1	1/36	ab_3c_3	1/36
ab_2c_2	21/36		

Table 2.1

Example 2.3.2 A right triangle and eight other distorted versions are shown in Fig 2.6 based on the pattern primitives shown in Fig 2.5 and probabilities information listed in Table 2.2.

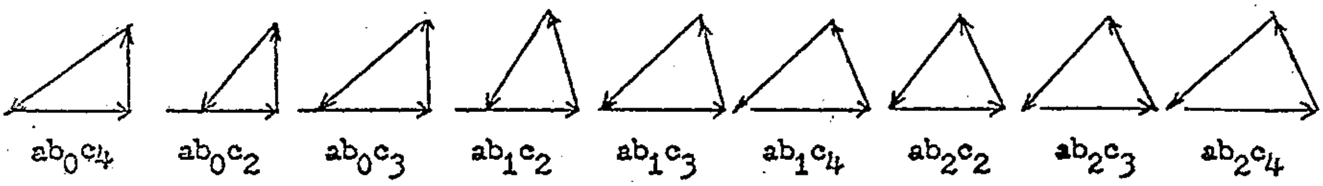


Fig 2.6 noisy triangles

If we define the strings (triangles) in Table 2.1 as forming the pattern class I and those in Table 2.2 as pattern class II, then class I $\cap$ class II = $\{ab_1c_2, ab_1c_3, ab_2c_2, ab_2c_3\}$.

X	p(x)	x	p(x)
ab_0c_2	1/36	ab_1c_4	1/36
ab_0c_3	2/36	ab_2c_2	2/36
ab_0c_4	21/36	ab_2c_3	3/36
ab_1c_2	1/36	ab_2c_4	1/36
ab_1c_3	4/36		

Table 2.2

In the dicision-theoretic approach, the classification problem with "overlapping" classes can be solved by applying statistical decision theory [69]. A similar idea is also applied here. The probability information p(x) of the strings belonging to each pattern class plays an important role in the classification problem. For example, suppose that the input pattern (a triangle) is represented by the string ab_1c_2. With the assumption of equal a priori probabilities (of the occurrences of each class) the information p(x) can be used for the maximum-likelihood [23] classification rule. That is, in this case, ab_1c_2 should be classified as belonging to class I since $P_I(x) = \frac{2}{36} > \frac{1}{36} = P_{II}(x)$ where $x = ab_1c_2 \in$ class I $\cap$ class II.

In [22] it has been shown that any normalized stochastic context-free grammar (ns fg) has its equivalent Chomsky Normal Form (CNF) and Greibach Normal Form (GNF). In a very similar way, Zadeh (72) has successfully derived the CNF and GNF of the so called fuzzy grammars (also known as weighted grammars) which can be described as follows :

Dfn 2.3.2 A fuzzy grammar is a quadruple $G_f = (V_N, V_T, P, S)$ where V_N, V_T, P, S are as usual. The elements of P are in the following form:

$$\mu(\alpha \rightarrow \beta) = \rho \quad , \qquad \rho > 0$$

where α and β are strings in $(V_T \cup V_N)^*$ and ρ is the grade (or weight) of memberships β geven α . A fuzzy languages $L(G_f)$ generated by G_f is a fuzzy set in V_T^*. It is a set of ordered pairs $L = \{(x, \mu_L(x))\}$, $x \in V_T^*$ where $0 \leq \mu_L(x) \leq 1$ is the grade of membership of X in L. The union of two fuzzy languages $L_1 + L_2$ in V_T^* is defined as $\mu_{L_1 + L_2}(x) = \max(\mu_{L_1}(x), \mu_{L_2}(x))$, $x \in V_T^*$ or $\mu_{L_1 + L_2} = \mu_{L_1} \vee \mu_{L_2}$ for short. The intersection of two fuzzy languages is $L_1 \cup L_2$ in which $\mu_{L_1 \cap L_2}(x) = \min(\mu_{L_1}(x), \mu_{L_2}(x))$, $x \in V_T^*$ or $\mu_{L_1 \cap L_2} = \mu_{L_1} \wedge \mu_{L_2}$. Thus μ can be thought of as a point in a lattice [38]. The concatenation of L_1 and L_2 is denoted by $L_1 \; L_2$ and if $x = uv$ then

$$\mu_{L_1 L_2}(x) = \sup \min(\mu_{L_1}(u), \mu_{L_2}(v))$$

$$\text{or} \quad \mu_{L_1 L_2} = \bigvee_u (\mu_{L_1}(u) \wedge \mu_{L_2}(v))$$

The grade of a string $x \in L(G_f)$ is represented by

$$\mu_{G_f}(x) = \sup \min(\mu(s \rightarrow \alpha_1), \mu(\alpha_1 \rightarrow \alpha_2), \ldots, \mu(\alpha_n \rightarrow x))$$

$$= \sup \min(\rho_1, \rho_2, \ldots, \rho_{m+1})$$

$$\text{or} \quad \mu_{G_f}(x) = \bigvee_k (\rho_1 \wedge \rho_2 \cdots \wedge \rho_{m+1})$$

$$= \bigvee_{l} (\bigwedge_{1}^{m+1} \rho_j)$$ for short, if there are k ways to derive

Note that here the fuzzy grammar G_f is defined almost the same way as the stochastic grammar G_s if we slightly modify our notation in Dfn 2.3.1 for the unrestricted associated generation probabilities:

$$g(x) = \sum_{1}^{m} \left(\prod_{j=1}^{n} p(\gamma_j) \right) \text{ if there are m ways to derive x.}$$

The theory of fuzzy languages offers what appears to be a fertile field for further study. It may prove to be of relevance in the construction of better models for natural languages and may contribute to a better understanding of the role of fuzzy algorithems and fuzzy automata in decision making, pattern recognition, and other processes involving the manipulation of fuzzy data. Further works can be found in [73,74].

§2.4 Stochastic Programmed Array Grammar and Syntactic Pattern Recognition

Now we have enough information ready to conbine the concepts of "programmed and stochastic" idea together into array grammars. So far no such a definition has been found yet. I try to define it in the following way.

Dfn 2.4.1 A stochastic programmed array grammar (SPAG) is a quadruple $G_{spa} = (G_A, I, M_s, M_f)$ where G_A is an array grammar as defined in Dfn 2.1.1, I is an initial rule choice vector, and M_s and M_f are programming matrices for success branch field and failure branch field respectively. Let G_A be a grammar with n rules, numbered 1,...,n. Let M_s and M_f be two n-by-n programming matrices, called the success and failure matrices, where each row of M_s and M_f sums to either 0 or 1. The scheme for selecting a production is ; at the beginning, the rule is selected according to the initial rule choice vector I, which is a 1 x n row matrix. Next, if rule k has just been selected but did not apply, then the next rule is chosen according to the probability density imposed by the weights in the kth row of the M_f if this row does not sum to zero. If, in either case, the kth row sums to 0, there is no next rule, and the derivation has halted. The language generated by G_{spa} is denoted by $L_g(G_{spa})$. Clearly, $L_g(G_{spa}) \subset L(G_A)$. The following example will describe the potential of generating array languages by a spag G_{spa} [52]

Example 2.4.1 Let $G_{spa} = (G_A, I, M_s, M_f)$ where

$G_A = (\{S,T,U,V,A,B,C,H,X,Y,I,\#\} , \{A,B,C,H,X,Y,I\} ,\#,P,S)$ where

$$P: (1)\ \begin{matrix}\#\#\\ S\#\end{matrix} \rightarrow \begin{matrix}UT\\ CT\end{matrix} \qquad (4)\ \begin{matrix}\#\\ UI\end{matrix} \rightarrow \begin{matrix}U\\ YI\end{matrix} \qquad (6)\ \begin{matrix}\#\\ UH\end{matrix} \rightarrow \begin{matrix}A\\ YH\end{matrix}$$

$$(2)\ \begin{matrix}\#\\ T\#\end{matrix} \rightarrow \begin{matrix}T\\ IT\end{matrix} \qquad (3)\ \begin{matrix}I\\ V\#\end{matrix} \rightarrow \begin{matrix}I\\ XV\end{matrix} \qquad (7)\ \begin{matrix}H\\ V\#\end{matrix} \rightarrow \begin{matrix}H\\ XB\end{matrix}$$

$$(3)\ T \rightarrow H$$

and $M_s = \begin{pmatrix} 0 & m & n & 0 & 0 & 0 & 0 \\ 0 & m & n & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$ $\quad M_f = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$

$I = (1\ 0\ 0\ 0\ 0\ 0\ 0)$

It can be shown, that L(Gspa) is the set of all labeled right triangles which would look like this :

```
A
Y B
T I B
Y I I B
C X X X B
```

Compared with Krisch [33], this is much more simpler. In general, if rule 2 is applied $k > 0$ times, then the number of derivation is $3k+4$, and the length of the side (including vertices) is k + 3. Let y = number of steps to derive a triangle, x = length of the side of a triangle, then y is linearly dependent on s, i.e. $y = az + b$. In this example, $y = 3k + 4$, $x = k + 3$ ∴ $y = 3x - 5$. Meanwhile for the non-parallel array grammars generating these triangles y is proportional to x quadratically.

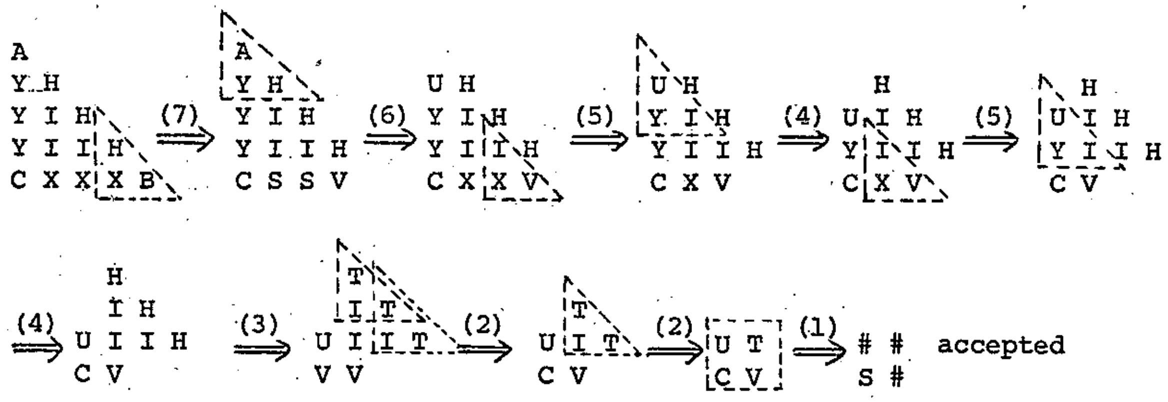
Given a language L and a spag G_{spa} generating L, it may be necessary to define a new spag G'_{spa} to recognize L, since the generating grammar may be such that no pair of matrices cause a parse to proceed correctly. Also since the language parsed by a spag need not be the same as the language generated [51], we define $L_p(G_{spa})$ to be the language parsed by G_{spa}. In this example the new spag, G'_{spa} is defined as follows :

$$G'_{spa} = (G_A, I', M'_s, M'_f) \text{ where } G_A \text{ is the same as before}$$

$$M'_s = \begin{pmatrix} 0&0&0&0&0&0 \\ 0&1&0&0&0&0 \\ 0&1&0&0&0&0 \\ 0&0&0&1&0&0 \\ 0&0&1&0&0&0 \\ 0&0&0&1&0&0 \\ 0&0&0&0&1&0 \end{pmatrix} \qquad M'_f = \begin{pmatrix} 0&0&0&0&0&0&0 \\ 1&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0 \\ 0&0&1&0&0&0&0 \\ 0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0 \end{pmatrix}$$

$$I = (0\ 0\ 0\ 0\ 0\ 0\ 1)$$

If we want to recognize a language L_p, we can parse it this way :



In [52] a generator programmar for stochastic ocntext-free programmed grammar languages is designed. In [23] a stochastic push down automata (spda)M_s is described and the language accepted by final state with a cut-point $0 \leq \lambda < 1$ is $T(M_s, \lambda) = \left\{ (x, p(x)) \mid x \in \Sigma^* : (q_0, z_0, 1) \mathrel{\vdash^*_{M_s}} (q_i, \gamma_i, p_i(x)),\ \text{for}\ \gamma_i \in \Gamma^*, q_i \in F, i = 1, ,,, , k,\ \text{and}\ p(x) = \sum_{i=1}^{k} p_i(x) > \lambda \right\}$

where k is defined as the number of districtively different derivations defined by δ. As suggested by Rabin [23], a stochastic experiment can be derived to test whether a sequence $x \in L(M_s, \lambda)$ is accepted by a given stochastic automaton $M_s = (\Sigma, \Phi, M, \Pi_0, F)$ with a cut-point λ. In this experiment, the Chebyshev's inequality [35] is applied to evaluate the confidence level ξ

$$P\left(\left| \frac{m}{n} - p(x) \right| \leq \frac{c}{\sqrt{n}} \right) \geq 1 - \frac{1}{4c^2} = \xi$$

where Mtimes (out of total n times) M_s is ended in final state and c is a parameter which controls the trade-off between and the expected theoretical termination time $T = \dfrac{c^2}{(p(x) - \lambda)^2}$. Fig 2.7 and Fig 2.8 show the relation between ξ and c, and the result of stochastic experiment with various confidence level. The conclusion of the experiment agrees with that of the theroetical analysis

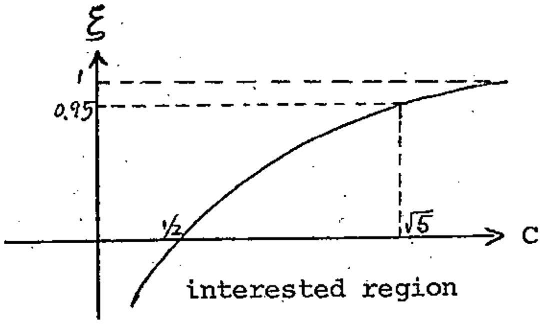


Fig 2.7

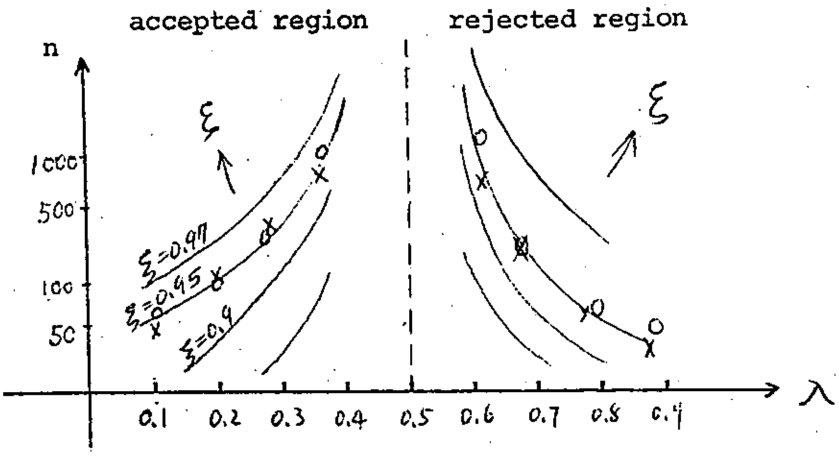


Fig 2.8 0 : theoretical x : experimental

The relationship between formal languages and λ- stochastic [21] languages can be roughly drawn in Fig 2.9 where :

λS : the set of λ- stochastic languages

R E : the set of recursively enumerable languages

F S : the set of finite-state languages

C F : the set of context-free languages

C S : the set of context-sensitive languages

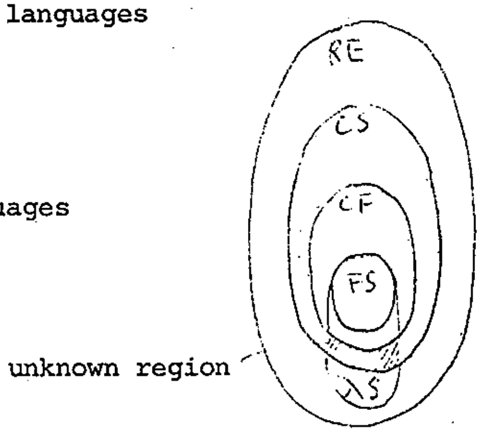


Fig 2.9

Note that $\lambda S \supset FS$

$\lambda S \cap DF \neq \emptyset$

$\lambda S \cap CS \neq \emptyset$

and the $\lambda S \cap (CS - CF)$ is still unknown.

In pattern recognition problems, the description of patterns can be viewed as languages generated by a certain stochastic grammar with the underlying statistical properties. Based on the knowledge of the stochastic grammar and the nature of classification, a stochastic automaton can be synthesized to relate the input descriptions and a priori knowledge, then, togheter with a threshold (e.g. cut-point), a decision (or classification) can be made. The value of the threshold can be adjusted dependent upon the actual output decision (refer to Fig 2.10)

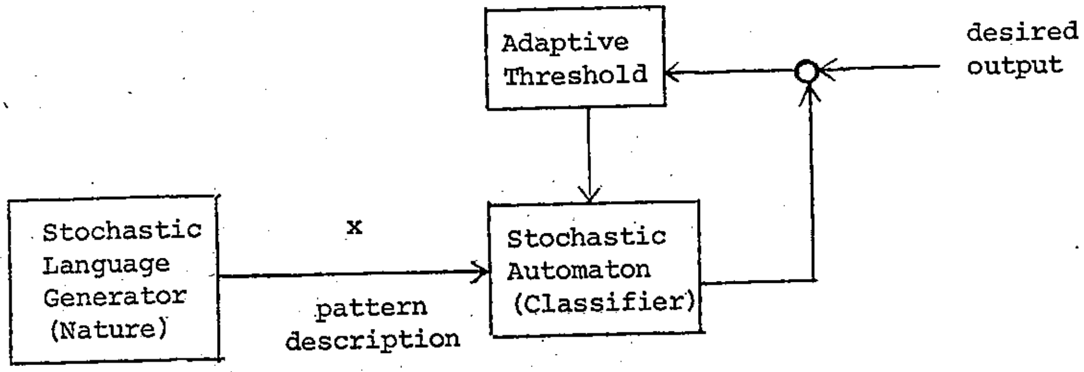


Fig 2.10 Formulation of Pattern Recognition

The concept of stochastic automata for pattern rrcognition seems very practical and powerful. However, unfortunately, the development of principles of stochastic automata applied to array grammar has not been found yet.

Chapter 3 Celluar Automata

The subject of cellular automata - also known variously as cellular space [15,64,66], modular computer [59], or tessellation automata [7,70,71], deals with large collection of interconnected finite-state Moore machines (automata), each finite automaton being thought of as a cell. It can be used as a medium for theoretical studies of pattern recognition, biological modeling and evolution processes, also as a foundation for a theory of logical design based on integrated circuits [15,66]. It can also be shown [60] that cellular automata are faster than iterative acceptors (in real time or linear time).

§ 3.1 Definitions of One-Dimensional Cellular Automata

One may envision a 1-D cellular automata as an infinite strip of film, each frame of which represents a copy of a single finite-state machine (or cell). Associated with each cell is a local transition function δ which obtains the next state of the cell as a function not only of the present state of the cell but also as a function of the present states of a specified set of neighboring cells in its neighborhood. It can be shown that a 3-cell neighborhood - a cell and its left and right neighbors - always suffices in the 1-D case; hence we assume this neighborhood. If Q is the state set of each cell, then the input set is Q x Q. That is, the out put of a cell is taken to be its state, and this output is used as input to the two nearest neighbors. Hence $\delta : Q^3 \rightarrow Q, (x,y,z) \rightarrow Y'$ is the local transition function for a cell in state y with left neighbor in state x and right neighbor in state z. A global transition function Δ can be defined as the simultaneous invocation of δ at each cell. Thus Δ maps a configuration, i.e. an assignment of states to each cell in a cellular space, into another configuration. There is a special state $q_0 \in Q$, called the quiescent state, such that $\delta(q_0, q_0, q_0) = q_0$. We define a pattern as the finite portion of a configuration between the two boundary cells.

Dfn 3.1.1 A deterministic bounded cellular space (DBCS) is a 1-D cellular space, denoted by the 4-tuple (X, Q, δ, b), with the 3-cell neighborhood, state set Q, and deterministic local transition function $\delta : Q^3 \rightarrow Q$ restricted as follows: (1) $b \in Q$ is a special boundary state, (2) $X \subset Q_b = Q - \{b\}$ is the initial alphabet (3) $\delta(q_i, b, q_j) = b$ for arbitrary $q_i, q_j \in Q$ (4) two and only two cells, the boundary cells, are in state b at time t = 0.

Dfn 3.1.2 The pattern transition function for a DBCS $Z = (X, Q, \delta, b)$ is the function $F : Q_b^* \rightarrow Q_b^*$ such that $F(q_1 q_2 \ldots q_n) = \delta(b, q_1, q_2)\ \delta(q_1, q_2, q_3) \ldots \delta(q_{n-2}, q_{n-1}, q_n)\ \delta(q_{n-1}, q_n, b)$ and $F(\Lambda) = \Lambda$, the empty string, where n is the number of cells in Z between the boundary cells.

Dfn 3.1.3 An element of Q_b^* is said to be a pattern for DBCS $Z = (X, Q, \delta, b)$. Let $R : Q_b^* \rightarrow Q$ be the extraction function which extracts the rightmost element of a finite pattern : $R(q_1 q_2 q_3 \ldots q_n) = q_n$ and $R(\Lambda) = b$.

Dfn 3.1.4 A DBCS $Z = (X, Q, \delta, b)$ is said to accept the language $L \subseteq X^*$ (on A) if, for arbitrary $x \in L$, there is a time t s.th. $R(F^t(x)) \in A$ where $A \subset Q$ is a set of accept states disjoint from X. We shall denote a DBCS used in this manner by the 5-tuple (X, Q, δ, b, A) and call it a DBCS acceptor. A is said to recognize L if it accepts L on A_1 and accepts $L' = X^* - L$ on

A_2 where $A_1 \cap A_2 = \emptyset$ and $A_1 \cup A_2 \subset Q$. If z recognizes L, we say Z rejects L'. Such a Z is called a DBCS recognizer.

A language accepting device is said to accept (recognize) a language L within time T(n) if, for any x of length n, it can determine whether (or not) $x \in L$ within T(n) steps, where $T: N \rightarrow N$ is a total time function on the positive integers. T(n) = n is called real time; T(n) = cn, c is a constant is called linear time.

Dfn 3.1.5 L is a DBCS language if there is a DBCS acceptor $Z_t = (X, Q, \delta, b, A)$ s.th. $L = L(Z) = \left\{ x \in X^* \ (\exists t) \ \left[R(F^t(x)) \in A \right] \right\}$. Similarly, L is a DBCS predicate if it is recognized by some DBCS recognizer. A real-time DBCS language (predicate) is a DBCS language (predicate) which is accepted (recognized within T(n) = T. Similarly, the adjective linear-time implies T(n) = cn.

Thus a string is accepted if, when embeded between two boundary cells in some DBCS acceptor, action of the pattern transition function caused the rightmost cell to eventually pass into an accept state.

§3.2 Definitions of Two-Dimensional Cellular Automata

A 2-D cellular automaton is an infinite array of finite-state machines (FSM), called cells, where each cell is assigned a point in I^2 (Fig 3.1). In general, a cell is non-deterministic. The local transition function of a cellular space obtains the next state the same way as defines in §3.1 except that the neighborhood is 2-dimensional. This neighborhood is sometimes called the von Neumann or H_1-neighborhood (Fig 3.2) where $H_k = \left\{ (i,j,) \in I^2 \mid |i+j| \leq k \right\}$ defines a general class of neighborhood. It can be shown [58] that there is no loss of generality in assuming the H_1-neighborhood. The configuration C, global transition function F and the quiescent state of a cellular space Z are defined similarly as in § 3.1 except that preimage is I^2 instead of I. The support of a configuration C is given by $\sup(C) = \{(i,j,) \mid C(i,j,) \neq q_0\}$. An initial configuration (i.e. at time 0) is assumed to have finite support.

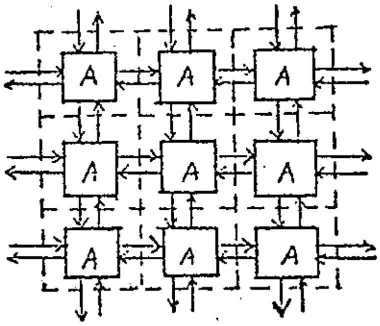


Fig 3.1 Cellular Space

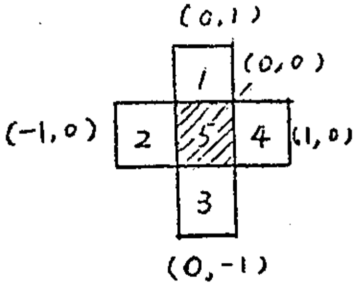


Fig 3.2 von Neumann Neighborhood (h_1- neighborhood)

Dfn 3.2.1 A (2-D) array bounded cellular space (BCS) is a cellular space for which each cell has a specially designed state B, called the boundary state, and a local transition function restricted to map a cell in state B into state B in all cases. Furthermore no boundary states can be created after time zero. The boundary cells at time zero in a BCS are assumed to delineate a connected subset of cells, called a retina of the BCS.

Dfn 3.2.2 A simply-connected BCS (SBCS) is a BCS for which each retina is simply-connected. A rectangular BCS (RBCS) is an SBCS for which each retina is restricted to form a rectangle. The rightmost cell in the uppermost row of a retina in a BCS Z is called the accept cell for that retina in Z. Notice that, since no boundary cells can be created after t = 0, a given retina in Z remains fixed in Z for all $t \geq 0$. Hence we shall call the accept cell of a retina in Z the accept cell of Z.

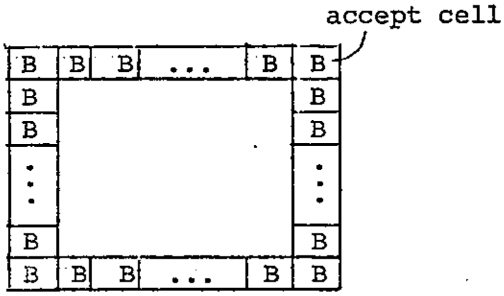


Fig 3.3 a RBCS retina

§3.3 Cellular Automata and Array Automata

We can define a Turing array acceptor (TAA) in parallel to the definition of an array grammar (§2.1) to be a 5-trple $T = (Q, \Sigma, \delta, q_s, F)$, where Q is a finite set of states of the form $Q' \times \Delta$ (where $\Delta = L,R,U,D$), Σ is a finite set of symboles, $(q_s, R) \in Q$ is a start symbol, $F \subseteq Q$ is a set of final states, δ is a mapping from $Q \times \Sigma$ into $2^{Q \times \Sigma \times \Delta}$ such that the triples in the image sets are all of the form $((q,Y),B,Y)$, $Y \in \Delta$. The interpretation of δ is as follows: If T is in internal state p and has just moved in direction X, and it reads symbol A, it goes into some internal state q, writes symbol B and moves in direction Y. T is called deterministic if the image under δ of every pair in $Q \times \Sigma$ is a singleton, otherwise, nondeterministic. T is called finite-state (FSAA) if it can never rewrite the symbols that it reads, in other words, if every triple in the image of any $((p,X), A)$ under δ is of the form $((q,Y),A,Y)$.

By an inut array on $\sum$ we-mean a mapping G from I^2 into $\sum$ s.th. the preimage P of $\sum - \{\#\}$ is finite and connected. We allow T to operate on G by starting with a pair whose terms are the start state of T and a point (i,j) of P (the initial "position" of T). The mapping δ is applied to the pair $((q_s,R),A)$. where A is the value of G at (i,j). If any such sequence of applications of δ leads to a triple whose first terms is in F, we say that T accepts G.

A TAA will be called array-bounded (an ABA) if it "bounces off" #'s, i.e. if every triples in the image of any $((p,X),\#)$ under δ is of the form $((q,X^{-1}),\#,X^{-1})$. It will be assumed that the input array of an ABA always contains a non-#. There are two famous therems related to array grammars and array automata by Milgram and Rosenfeld [36] as follows:

Theorem 3.3.1 Let $\mathcal{L}$ be the language of an AG; then there exists a TAA that accepts just the arrays of $\mathcal{L}$. Conversely, let $\mathcal{L}'$ be the set of input arrays accepted by a TAA; then there exists an AG whose language is $\mathcal{L}'$.

Theorem 3.3.2 Let $\mathcal{L}$ be the language of an MAG (monotonic array grammar); then there exists an ABA that accepts just the arrays of $\mathcal{L}$. Conversely, let $\mathcal{L}'$ be the set of input arrays accepted by an ABA, then there exists an MAG whose language is $\mathcal{L}'$.

Since cellular automata can both generate languages as well as recognize languages, if we can relate it with array automata (via array grammars) it would be very beautiful and pratical. So far.no such an equivalent relation has been found yet. Here i try to propose a pre-theorem which relates array automata with cellular automata (via array grammar) in some cases, the proof of which is still bo be worked out.

Pre-Theorem 3.3.3 For any MAG with minimal circumscribing rectangle [15] no greater than 2 x 2, there corresponds an equivalent cellular automaton Z' with an H_1-neighborhood which can simulate the MAG. The simulation time depends on the number of rewriting rules and the sizes of the minimal circumscribing rectangles of rules in the MAG. A more detailed discussion will be found in Chapter 5.

§3.4 Pattern Recognition by Cellular Automata

A cellular automaton is intuitively a pattern reocgnition receiving retina, especially in the 2-D case. Hence the pattern recognition capabilities of cellular automata is particularly interesting. Initial work [8,11,59] on this problem has shown that these devices can recognize a wide variety of topological invariants, including connectivity, in linear time. It has also been shown [57,58,59,60] that cellular automata are inherently faster than iterative acceptors. A non-deterministic bounded cellular space is defined just as is a DBCS with the exception that $\delta : Q^3 \rightarrow 2^Q$ is a non-deterministic local transition function with the restriction that $\delta(q_i,b,q_j) = \{b\}$, for arbitrary $q_i,q_j \in Q$. A language is accepted in this case if at some time t it is possible for the rightmost pattern cell, to enter an accept state. An n-D iterative automaton is an n-D cellular automaton with a distinguished cell which has a local transition function augmented to be a function of an external input also. A string is said to be accepted by an iterative automaton if the distinguished cell ultimately goes into an accept state and emits a corresponding output.

An iterative automaton used in this accepting mode is called an iterative acceptor. A real time iterative acceptor accepts within time T(n) = n. A real-space iterative acceptor uses no ohter cells than does a real-time iterative acceptor. That is for the 1-D case, only the distinguished cell, the n cells immediately to its right, and the cells immediately to its left can ever change state. Hence a real-time iterative acceptor is a special case of a real-space iterative acceptor. An interative acceptor is non-deterministic (deterministic) if its local transition function, including that of the distinguished cell, is non-deterministic (deterministic) - just as for cellular automata.

Now we shall make use of the following easily proved theorem[57,59]:
The Speed-Up Theorem (for DBCS) Let k be an arbitrary positive integer. For an arbitrary DBCS acceptor $Z = (X,Q, \delta, b,A)$ with $|Q| = r$, there exists a DBCS acceptor $Z' = (X,Q', \delta', b,A)$ with $|Q'| = 8r^k$ wuch that of Z accepts within time T(n) then Z' accepts within time (T(n)/k) + n.

There are micellaneous theorems concerning about the equivalence and complexity results of cellular automata, iterative acceptors, and linear-bounded automata [58,60]. Here we are going to see an example which shows the recongition capability of cellular automata.

Example 3.4.1 $L = \left\{ a^m b^m c^m \mid m \geq 1 \right\}$ is a real-time DBCS language. Consider Fig 3.4. To erase confusion, let B be the boundary state in this case. At the 1st step, each ab boundary, bc boundary, Ba boundary and cB boundary is specially marked. Each ab boundary sends a pulse to the right at 1/2 unit speed checking for all b's. Each bc boundary sends a pulse left at unit speed checking for all b's. The Ba boundary sends a pulse right at unit speed checking all a's. Shoud it collide with a bc boundary pulse at an ab boundary, then form Ba'b' is guaranteed. The cB boundary sends a pulse left at 1/2 unit speed checking for all c's. Should it collide with an ab boundary pulse at a bc boundary, then form $b^j c^j B$ is graranteed. Furthermore, should it also collide with the Ba boundary pulse, which has determined the existence of form $Ba^i b^i$, at the same bc boundary, then form $Ba^m b^m c^m B$ is guranteed and an accept pulse is sent right.

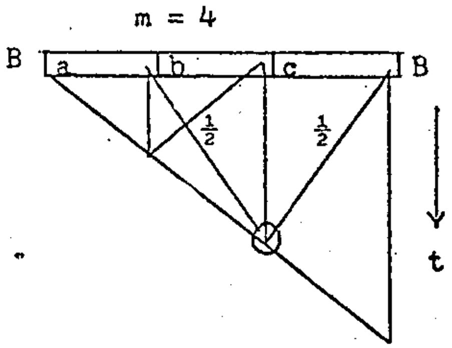


Fig 3.4 $a^m b^m c^m$ recognizer

For 2-D case, a BCS Z eith retina R is a pattern recognition device in the following sense : Let $X \subset Q - \{B\}$ be the initial alphabet of Z. Let X^* be the set of all connected words on alphabet X, where a connected word is a mapping from a finite, possibly empty, connected subset of I^2 into X. Assume that R is initially programmed with elements from X. Thus for initial configuration C_0, C_0 (R) $= w \in X^*$. The word w is said to be accepted by Z (on A) if there is a time t such that $[F^t(C_0)]$ $(i_R, j_R) \cap A \neq \emptyset$. where $A \subset Q - \{B\}$ is disjoint form X and (i_R, j_R) is the accept cell of Z. A language $L \subseteq X^*$ is accepted by Z (on A) if, for arbitrary $w \in L$, w is accepted by Z on A. A language $L \subseteq X^*$ is a BCS language if there is a BCS with initial alphabet X which accepts precisely L. Similarly a language

L is recognized by a BCS Z if it is not only accepted by Z but its complement $L' = X^* - L$ is rejected by Z, i.e. L' is accepted on A; where $A' \cap A = \emptyset$. A BCS predicate is a language L for which there is a BCS which recognizes L.

The speed of recognition of patterns by BCS shall be a major concern. For RBCS, a natural measure on the retinas is $m + n$, where m and n are the dimensions of a retina. Hence linear time will imply a number of time steps proportional to $m + n$, and area time is a number of time steps proportional to mn for RBCS and to the number of points in the retina for arbitrary BCS. A time measure of particular interest here will be called perimeter time, i.e. a number of time steps proportional to the perimeter of a given retina. At best, perimeter time is linear time. At worst, perimeter time is area time. Note that for RBCS, perimeter time is linear time. External perimeter time is a number of time steps proportional to only the external perimeter of a retina (i.e. addition to the perimeter due to holes in the retina are not included).

Previous work [60] in the area of pattern recognition by 2-D cellular automata has assumed rectangular retinas (i.e. RBCS). The following is a very important theorem by Smith [60] :

Theorem 3.4.1 There is a DBCS Z which recognizes the language L of all simply-connected words on arbitrary finite alphabet X within external perimet perimeter time.

A simple consequence from Smith is that if an ABA accepts a language L within time T, then there is a BCS which accepts L within time $2T + kp^2$ Theorem 3.4.1 brings the time limit down to $2T + kp$ where p is the perimeter. This is accomplished from the following lemma.

Lemma 3.4.1 Given a DBCS Z, there is a special cell in each retina R of Z which can be uniquely identified within perimeter time. That is, for C_o an initial configuration in Z entirely quiescent on R which has perimeter p, there is a special state \$ and a special cell $(i_R, j_R) \in R$, the accept cell, and a time t = kp, k constant, such that $[F^t(C_o)](i_R, j_R) = \$$ and such that $[F^{t'}(C_o)](i,j) \neq \$$ for all t' if $(i,j) \neq (i_R, j_R)$ or for all $t' < t$ if $(i,j) = (i_R, j_R)$.

This lemma is rather interesting in that it makes extensive use of the 1-D firing squad result (see Waksman[67]) and of the 1-D Dyck language recognizing BCS of Smith [57,59,60]. Both of these results are intrinsically cellular automata theoretic, and hence take on the appearance, at least, of basic theorems for cellular automata theory.

Chapter 4 Grammatical Inference

In Chapter 2 we see that the major tools for syntactic pattern recognition are various types of "grammars". However the ways how to derive them are not mentioned at all. The principal object of this chapter is to describe some recent studies of methods for the automatic inference on pattern grammars.

The grammatical infernece problem can be described as follows : a finite set of symbol strings from some language L and possibly a finite set of strings from the complement of L are known, and a grammar for the language is to be discovered. Precisely the same problem arises in trying to choose a model or theory to explain a collection of sample data. This is one of the most important information processing problems known and it is surprising that there has been so little work on its formalization [9,19,28,33,47,69]. The grammatical inference problem and its solution have implications for pattern recognition research for two reasons : (1) Considerable research has been invested in recent years into the development of linguistic methods for picture description and analysis, and the discovery of grammars for these systems has posed a problem. Various researchers [19,69] have indicated a need for improved methods for grammar discovery. (2) If pattern recognition is a research for structure in information space, then grammatical inference can be considered to be an example of pattern recognition in itself. In this case, the observed data is the pattern to be analyzed, and the inferred grammar can be thought of as its description or classification.

A grammatical inference problem is well-formed if we are given : (1) The hypothesis space, i.e. the class of grammars to be inferred, (2) The observation space, i.e. the form of the samples, and anything which is known about their structure, (3) The evaluation measure, i.e. an objective definition of what it means for grammars to be the "best" hypothesis on the basis of a sample, (4) The required performance, i.e. the criterion an accepted solution must satisfy.

Of course, not all well-formed problems have solutions, but unless an inference technique solves some well-formed problem, it is difficult to make any useful statement about its domain of applicability.

This chapter is roughly divided into two parts : the first part deals with the non-stochastic languages and the second part deals with stochastic languages.

For the non-stochastic language we'll give an example to demonstrate the derivation and reduction processes of context-free grammars.

Example 4.1 Find a ocntext-free grammar $G = (V_N, V_T, P, S)$ such that $L(G) = \left\{ ca^n b,\ bba^n b \;\middle|\; n \geq 0 \right\}$

First we introduce a predicate adj (x;y) which is true if the substring x is immediately left-adjacent to the substring y, then defining syntactic types in the usual way in terms of constituents, using the adj predicates [19,38]. Thus our first phase produces for each scene a grammar for each alternative possible structural description. To shorten the description, we shall look only at a 7-tuple of grammar (i.e. {caaab,bbaab,caab,bbab, cab,bbb,cb}) that will reduce to the final grammar. They are as follows, taking the strings in the order given above and write $C \rightarrow AB$ instead of

$C \to (x,y) : A(x), B(y) : adj\ (x : y)$. For instance we label the string caaab as follows :

caaab
12345

then 6 : (4,5) : adj(4;5) 8 : (2,7) : adj(2;7)

7 : (3,6) : adj(3;6) *9 : (1,8) : adj(1;8)

The asterisk 9 corresponds to the structural description that we are able to construct the string caaab. The structural tree is shown in Fig 4.1.

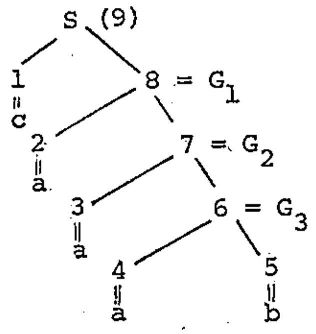


Fig 4.1 Tree of caaab

The productions are :

(a) $S \to cG_1$ $G_2 \to aG_3$

$G_1 \to aG_3$ $G_3 \to ab$

Similarly we obtain :

(b) for string bbaab

$S \to bbG_4$

$G_4 \to aG_5$

$G_5 \to ab$

(d) for string bbab

$S \to bbG_8$

$G \to ab$

(f) for string bbb

$S \to bbb$

(c) for string caab

$S \rightarrow cG_6$
$G_6 \rightarrow aG_7$
$G_7 \rightarrow ab$

(e) for string cab

$S \rightarrow cG_9$
$G_9 \rightarrow ab$

(g) for string cb

$S \rightarrow cb$

The first step of reduction process is to check : since substitute G_5, G_7, G_8, G_9 by G_3 we have identical rules like $G_3 \rightarrow ab$, so eliminating rules $G_{5,7,8,9} \rightarrow ab$ the resulting productions are :

$S \rightarrow cG_1$	$S \rightarrow cG_6$
$G_1 \rightarrow aG_2$	$G_6 \rightarrow aG_3$
$G_2 \rightarrow aG_3$	$S \rightarrow bbG_3$
$G_3 \rightarrow ab$	$S \rightarrow cG_3$
$S \rightarrow bbG_4$	$S \rightarrow bbb$
$G_4 \rightarrow aG_3$	$S \rightarrow cb$

The next pair of reductions identifies G_1, G_3 and G_6, resulting in

$S \rightarrow cG_1$	$S \rightarrow cG_1$
$G_1 \rightarrow aG_2$	$G_6 \rightarrow aG_1$
$G_2 \rightarrow aG_1$	$S \rightarrow bbG_1$
$G_1 \rightarrow ab$	$S \rightarrow cG_1$
$S \rightarrow bbG_4$	$S \rightarrow bbb$
$G_4 \rightarrow aG_3$	$S \rightarrow ab$

The next pair of reductions identifies G_1, G_2 and G_4 resulting in

$S \rightarrow cG_1$	$S \rightarrow bbG_1$
$G_1 \rightarrow aG_1$	$S \rightarrow bbb$
$G_1 \rightarrow ab$	$S \rightarrow cb$

The next reduction introduces $G_1 \rightarrow b$ and selectively substitutes G_1 for b in rules 3,5 and 6. So we have finally :

$$S \longrightarrow cG_1 \qquad S \longrightarrow bbG_1$$

$$G_1 \longrightarrow aG_1 \qquad G_1 \longrightarrow b$$

So the resulting context-free grmmar G is $G = (\{S, G_1\}, \{a,b,c\}, \{S \rightarrow cG_1 \mid bbG_1, G_1 \rightarrow aG_1 \mid b\}, S)$.

Some remarks of this mehtod should be mentioned here :

(1) Perhaps the most unsatisfactory disadvantage is that these grammars must be written in terms of a fixed set of predicates.

(2) Only positive instances, i.e. scenes to be fit by the final grammar, not a mixture of scenes labeled "yes" and "no" are shown in the grammar to be found.

(3) Only idealized "noise free" cases are to be taken into account. The statistical considerations are ignored.

The third problem may partially be resolved via the concept of stochastic grammatical inference proposed by Fu [21,22,23,34]. For a finite strings set over the set of symbols

$$X = \{x_1, \dots, x_n\} \qquad x_i \in \Sigma^*$$

let the given probability information be

$$R = \{f_1, \dots, f_n\} \qquad 0 \leq f_i \leq 1$$

where f_i is the probability or the relative frequency of occurrence of the string x_j. After (x_i, f_i) has been observed, let

$$h(z,X,k) = \{ (w, f_i) \mid zw = x_i \in X \text{ and } |w| \leq k \}$$

where $|w|$ denotes the length of the string w and $k \geq 0$. Based on the information (X,R), a stochastic automaton can be defined as

$$M\ (X,R,k) = (\Sigma, Q, \delta, q_o, F)$$

where Σ is the same set of input symbols

$$Q = \left\{ h(z,X,k) \;\middle|\; z \in \Sigma^* \right\} \quad \text{is the set of states,}$$

$$q_o = h(\lambda, X,k) \qquad \text{is the initial state (} \lambda \text{ is the}$$

empty string and

$$F = \left\{ h(z,X,k) \;\middle|\; h(z,X,k) = (\lambda, f_i) \right\} \text{ for some}$$

$f_i \in R$ is final state set .

Let $Q' = \left\{ h(z,X,k) \;\middle|\; (\lambda, f_i) \in h(z,X,k) \quad \text{for some} \quad f_i \in R \right\}$

be the set of states of which each serves at least as a final state and possibly also as a transition state. The transition from state $q = (w,f_i)$ $= h(z,X,k)$ to state $q' = (w',f_i) = h(za,X,k)$, $a \in X$, is defined as

$$(q,a) = \begin{cases} (a',p), \text{ if } q' \in (Q - Q') \text{ or } q' = q_f \in F \\ \left\{ (q',p_1),\ (q_f,p_2) \;\middle|\; q_f \in F \right\}, \text{ if } q' \in (Q' - F) \end{cases}$$

where p, P_1 and p_2 are transition probabilities which will be calculated from the relations $\quad \delta(q_o,s_i) = (q_f,\ f_i),\ \ i = 1, \ldots , n$.

It is anticipated that for a sufficiently large value of k (e.g. $k = \max x_i$), the above relation will give a unique solution of all the transition probabilities. In this case, the number of states would also be large enough so that the automaton would accept all the strings in X. Then the following condition may be realized :

$$L(M(X,R,k)) = (X,R) \qquad \sum_{i=1}^{n} f_i = 1$$

That is, the language accepted by the stochastic automaton M (S,R,k) will be exactly the same as X with the associated probability information R. However if $\sum_{i=1}^{n} f_i < 1$, there is a certain probability $1 - \sum_{i=1}^{n} f_i$ that other strings will also be accepted. (This is also one of the reasons that, instead of $\left\{p(x_1), \ldots , p(x_n)\right\}_n$, the notation $R = \left\{f_1, \ldots , f_n\right\}$ is used. Nevertheless, the quantity $\sum_{i=1}^{n} f_i$ may serve as a measure of confidence for the grammar inferred. In the following, without going through any further theoretical treatment, an example is given to illustrate the proposed procedure [21] .

Example 4.2 The strings and their associated probabilities listed in Table 2.1 are given as the input information , i.e.

$$(X,R) = \Big\{ (ab_1c_1,1/36),(ab_1c_2,2/36),(ab_1c_3,3/36),(ab_2,c_1,1/36), (ab_2c_2,21/36),(ab_2c_3,2/36),(ab_3c_1,3/36),(ab_3c_2,2/36), (ab_3c_3,1/36) \Big\}$$

Let $M(X,R,k) = (\sum, Q, \delta, q_o, F)$ where $\sum = \left\{a,b_1,b_2,b_3,c_1,c_2,c_3\right\}$

For $k = 3$ $z = \lambda$ $(|w| = 3)$ $q_o = (X,R)$

$z = a$ $(|w| = 2)$ $q_1 = \Big\{ (b_1c_1,1/36,(b_1,c_2,2/36), (b_1c_3,3/36),(b_2c_1,1/36),(b_2c_2,21/36),(b_2c_3,2/36),(b_3c_1,3/36), (b_3c_2,2/36),(b_3c_3,1/36) \Big\}$

$|w| = 1$ $z = ab_1$ $q_2 = \left\{(c_1,1/36),(c_2,2/36),(c_3,3/36)\right\}$

$z = ab_2$ $q_3 = \left\{(c_1,1/36),(c_2,21/36),(c_3,2/36)\right\}$

$z = ab_3,$ $q_4 = \left\{(c_1,3/36),(c_2,2/36),(c_3,1/36)\right\}$

$w = 0$ $z = ab_1c_1$, $q_5 = \left\{(\lambda, 1/36)\right\}$ $z = ab_1c_2$, $q_6 = \left\{(\lambda, 2/36)\right\}$

$z = ab_1c_3$, $q_7 = \left\{(\lambda, 3/36)\right\}$ $z = ab_2c_2$, $q_8 = \left\{(\lambda, 21/36)\right\}$

$z = ab_2c_1$, $q_9 = q_5$ $z = ab_2c_3$, $q_{10} = q_6$

$z = ab_3c_1$, $q_{11} = q_7$ $z = ab_3c_2$, $q_{12} = q_6$

$z = ab_3c_3$, $q_{13} = q_5$

$$Q = \left\{q_i \mid i = 0, 1, \dots, 8\right\}$$

$$F = (\lambda, 1/36), (\lambda, 2/36), (\lambda, 3/36), (\lambda, 21/36) = q_5, q_6, q_7, q_8$$

$$Q' = \left\{q_5, q_6, q_7, q_8\right\} \qquad Q - Q' = \left\{q_o, q_1, q_2, q_3, q_4\right\} \quad Q' - F = \emptyset$$

$$\delta(q_o, a) = (q_1, p_1) \quad \because q_o = h(\lambda, X, k), \quad q' = h(a, X, k) = q_1$$

$\delta(q_1, b_1) = (q_2, p_2)$ $\delta(q_3, c_1) = (q_5, p_8)$

$\delta(q_1, b_2) = (q_3, p_3)$ $\delta(q_3, c_2) = (q_8, p_9)$

$\delta(q_1, p_3) = (q_4, p_4)$ $\delta(q_3, c_3) = (q_6, p_{10})$

$\delta(q_2, cl) = (q_5, p_5)$ $\delta(q_4, c_1) = (q_7, p_{11})$

$\delta(q_2, c_2) = (q_6, p_6)$ $\delta(q_4, c_2) = (q_6, p_{12})$

$\delta(q_2, c_3) = (q_7, p_7)$ $\delta(q_4, c_3) = (q_5, p_{13})$

we then can construct the transition diagram of the stochastic automaton :

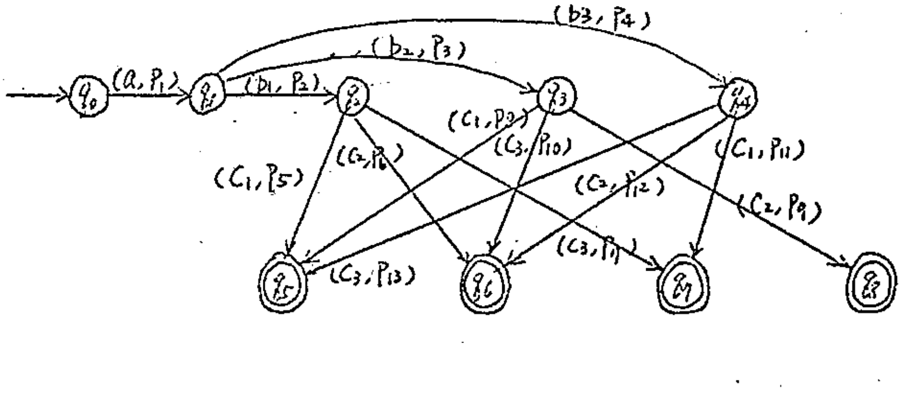


Fig 4.2 Transition Diagram

From Fig 4.2 and the information (X,R), the following relations can be established :

$$p_1p_2p_3 = 1/36 \quad p_1p_3p_8 = 1/36 \quad p_1p_4p_{11} = 3/36$$

$$p_1p_2p_6 = 2/36 \quad p_1p_3p_9 = 21/36 \quad p_1p_4p_{12} = 2/36$$

$$p_1p_2p_7 = 3/36 \quad p_1p_3p_{10} = 2/36 \quad p_1p_4p_{13} = 1/36$$

and the normalization conditions are :

$$p_1 = 1, \quad p_2 + p_3 + p_4 = 1, \quad p_5 + p_6 + p_7 = 1,$$

$$p_9 + p_{10} + p_8 = 1, \quad p_{11} + p_{12} + p_{13} = 1$$

we obtain

$$p_2 = p_{11} = p_7 = 1/2, \; p_3 = 2/3, \; p_4 = p_5 = p_{13} = 1/6,$$

$$p_6 = p_{12} = 1/3, \quad p_8 = 1/24, \; p_9 = 21/24, \; p_{10} = 1/12$$

After the stochastic automaton is synthesized, a corresponding stochastic finite-state grammar can be constructed using a procedure similar to the non-stochastic case [22,23].

$S \rightarrow aA$ p if $\delta(q\ ,a) = (q_1,p)$, $a \in \sum$, where A is the nonterminal associated with the state q and $q \notin F$

$A \rightarrow aA_2$ p if $\delta(q_1,a) = (q_2,p)$, $a \in \sum$, where A_1, A_2 are the nonterminals assoicated with the states q_1, q_2 respectively and $q_2 \notin F$.

$A \rightarrow b$ p if $\delta(q,b) = (q_f,p)$, $b \in \sum$; and $q_f \in F$ where A is the nonterminal assoicated with the state q.

Based on the above procedures we can find out the corresponding regular grammar as follows : $G_S = (C_N, V_T, P, S, D)$, $V_N = \{S_1, A_1, A_2, A_3, A_4\}$, $V_T = \sum$ and the rules of P are exactly the same as in § 2.3. This method, although very promising, is restricted regular grammars (via finite-state automata) only. The procedure for the construction of context-free sampled languages has not been found yet, let alone the context-sensitive grammars as well as the recursively enumerable grammars.

So far we have dealt with one-dimensional problems (i.e. string languages) only. The way to derive a grammar for two-dimensional languages (patterns) is partially successfully proposed by Evans [19] based on " descriptive-oriented" approach. The following example demonstrates the procedure:

Example 4.3 Suppose the input patterns are the scenes shown in Fig 4.3(a) and (b). The primitive object type is linesegment (seg) and the available predicates are join (x;y) and close (x;y), which apply to any object x,y made up of a sequence of line segments and test for situations like those shown in Fig 4.4(a) and (b).

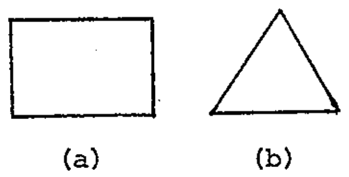


Fig 4.3

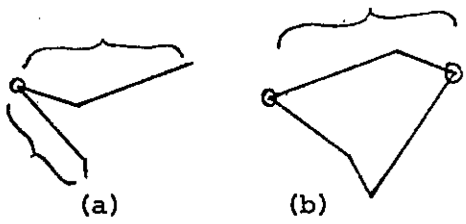


Fig 4.4

In a very similar way as in Example 4.1 we can find the production rules for (a) look like follows :

1 : $S \longrightarrow (x,y)$: seg (x), $G_1(y)$: close (x;y)

2 : $G_1 \longrightarrow (x,y)$: seg (x), $G_2(y)$: join (x;y)

3 : $G_2 \longrightarrow (x,y)$: seg (x), seg(y) :join (x;y)

and for (b)

4 : $S \longrightarrow (x,y)$: seg (x), $G_3(y)$: close (x;y)

5 : $G_3 \longrightarrow (x,y)$: seg (x), seg (y) : join (x;y)

We take the union of these two and after the reduction process (similar as in Example 4.1) we get the final result :

$$S \longrightarrow (x,y) \quad : \quad seg\ (x),\ G_1(y) \ :\ close\ (x;y)$$

$$G_1 \longrightarrow (x,y) \quad : \quad seg\ (x),\ G_1(y) \ :\ join\ (x;y)$$

$$G_1 \longrightarrow (x,y) \quad : \quad seg\ (x),\ seg\ (y) \ :\ join\ (x;y)$$

The result is a grammar that will recognize any polygon.

Strictly speaking, Example 4.2 also deals with tow-dimensional patterns (noisy triangles). However, the grammatical inference procedure for stochastic programmed array grammars have not been found yet. Even the array grammatical inference alone is unknown.

PART II

Discussions and Future Research

Chapter 5 Discussions and Open problems

Through the investigations from Chapters 2 - 4 we already have an introductory idea about two main tools for 2-D syntactic pattern recognition problems, namely, stochastic programmed array grammars and cellular sutomata. In chapter 2, it has been argued that effective methods for syntactic pattern recognition will require grammatical formalism which (1) generate rich class of languages, (2) have an associated probabilistic mechanism, and (3) are amenable to effective and efficient analysis procedures. The stochastic programmed array grammar is proposed as one possible solution of such formalism. The power of the context-free programmed grammar (cfpg) is reviewed and illustrated by examples (§ 2.2). A stochastic programmed array grammar is defined (§ 2.4) and some of its properties are considered. In short the context-free grammar is considerably more compact than the finite-state (regular) grammar. The programmed grammar is still more compact than the context-free grammar. Examples of a stochastic cfpg which generates noisy triangles are presented (§ 2.3). In parallel to the stochastic, programmed, array grammars, the so called fuzzy, matrix, picture processing grammars are also investigated. Their applications to recognition problems, advantages, disadvantages and some open problems are discusse (§ 2.1,2.2,2.3,2.4). In Chapter 4 an algorithm for discovering grammars given a set of sampled data languages is described as grammatical inference. Its capability limitation is also discussed (pp43 -47). Two big open problems should be emphasized here : (1) In §2.1 it is noted that the language generated by a given grammar when they are applied in parallel need not be the same as the language when they are applied sequentially and that parallel grammar is faster than sequential grammars. It's already known [51] that a context-free parallel language is not necessarily a context-free sequential language. However it is still an open question whether any context-free array language is a context-free parallel array [1]. (2) The algorithm described in Chapter 4 is applicable to descriptive patterns only. Perhaps the worst defficiency is that it can not grow. The formal algorithm to derive array grammars (which eliminate such defficiency) is still an open question.

In Chapter 3 we have approached pattern recognition with cellular automata via formal language theory. A special interest in time and memory requirements led us to introduce and study the real-time DBCS languages (§3.1 , §3.2), those languages accepted in real time by deterministic cellular automata which are bound to use only "real memory", i.e. only the memory of those cells to which an input is presented. Cellular automata and array automata are related via a Pre-Theorem (§3.3). A more detailed discussion will be described as follows :

Suppose AG is a monotonic array grammar, whose rewriting rules have a minimal circumscribing rectangle [15] no greater than 2 x 2, (i.e. 1 x 1, 1 x 2, 2 x 1, 2 x 2). If it is 2 x 2 we can select the so called J_1 neighborhood [58] for the corresponding cellular automata. For instance, the rewriting rule $\begin{array}{c}A\\BC\end{array} \to \begin{array}{c}X\\YZ\end{array}$ corresponding to the following local transition function δ with J_1 neighborhood:

$$\begin{array}{c}***\\ *A*\\ *BC\end{array} \xrightarrow{\delta} X\ , \qquad \begin{array}{c}*A*\\ *BC\\ ***\end{array} \xrightarrow{\delta} Y\ , \qquad \begin{array}{c}A**\\ BC*\\ ***\end{array} \xrightarrow{\delta} Z$$

where *'s are redundancies (which means don't care).

If it is 2 x 1 or 1 x 2, the H_1 neighborhood (§ 3.2) is enough. For instance : the rewriting rule $\begin{matrix} B \\ C \end{matrix} \rightarrow \begin{matrix} Y \\ Z \end{matrix}$ corresponds to the following local transition functions δ with H_1 neighborhood :

$$\begin{matrix} & * & \\ * & B & * \\ & C & \end{matrix} \longrightarrow Y, \qquad \begin{matrix} & B & \\ * & C & * \\ & * & \end{matrix} \longrightarrow Z$$

From [57] we know that for an arbitrary cellular space Z with a J_1 neighborhood, there is a cellular space Z' with an H_1 neighborhood which simulates Z in two times real time. Thus we can propose a so called Pre-Theorem as described in § 3.3.

Since it has been shown (§3.4) that cellular automata are inherently faster than iterative acceptors and cellular automata can do both generation as well as recognition with high speed [43], and recognition can be treated as a reverse process of pattern generation, we are particularly interested in the following open problems : (1) Can every finite pattern be generated from $\bar{0}1\bar{0}$ in 1-dimensional, binary, scope-3 tessellation automata ? This is an incompleteness problem proposed by Yamada and Amoroso [71]. It has already been solved partially by Amoroso and Patt [6]. (2) Are the context-free languages a subset of the real-time DBCS languages ? We have shown only partial answers to this question [59]. For example, the linear context-free languages are real-time DBCS languages. (3) Are the real-time DBCS languages closed under concatenation and reversal ? The answer to the general question is probably No. The real-time DBCS languages have been shown closed under union, intersection, complementation and set difference. (4) Do there exist non-linear DBCS predicates, i.e. DBCS languages which require non-linear recognition time ? It seems that the answer to this very interesting problem should be Yes, but attempts to use the diagnoalization proof technique [59] have all failed so far. The major difficulty appears to be the real memory requirement. This is essentially the problem first posed by Beyer [8].

To summarize, syntactic pattern analysis and recognition have been found to offer an approach to dealing with pattern data which cannot be conveniently described numerically or otherwise so complex as to defy analysis by conventional techniques. Some formidable hurdles remain to be cleared before syntactic pattern recognition can be widely applied. Some problems and areas of promise include :

(1) A syntactic approach to the determination of appropriate syntactic elements. Some sort of interaction between the primitives selection procedure and the evaluation of recognition performance is needed, both to optimize the performance and to minimize the analyzer complexity.

(2) Grammatical Inference (grammar synthesis based on samples of pattern data) techniques are needed by which analyzers could learn grammars from sets of training patterns [19,22,28,47].

(3) Generalizations of concatenation to multiple dimensions and more complex syntactic relationships, e.g. Shaw's PDL and web grammars due to Pfaltz and Rosenfeld appear promising in this respect [1,37,46,51,52,53].

(4) Presently attainable processing speed need to be improved, i.e. efficient parsing of development of special pattern languages and grammars. Parallel processing and cellular automata are possible solutions [43,51,57, 58,59,60].

(5) Detailed formation of a stochastic syntactic pattern analysis model capable of processing pattern with distortion and noise. A future imaginary syntax-controlled pattern scheme : A stochastic grammar is found for each individual pattern class; for each pattern to be classified, a parse and its associated probability are obtained according to each class grammar; based on the parse probability, the classification is then made accorging to a criterion such as minimum risk. Advantage: apply syntactic analysis directly to the raw data rather than going through an initial noise cleaning stage which makes little or no use of the wealth of prior information stored in the pattern grammar [21-23].

(6) Further contributions of automata theory to the syntactic analysis problems. Work is continuing on characterization of the classes of languages recognizable by various types of automata. Stochastic automata [23] theory may eventually provide both a theoretical foundation for stochastic syntactic analysis and an alternative approach to the realization of analytical mechanism based on stochastic grammars [21,22,23,40,44,45, 63,65].

The range of important problems to which syntactic pattern analysis could be applied and which otherwise appears to be beyond the scope of presently known techniques will continue to stimulate research in this new direction.

Chapter 6 Goals, Future Research and Methodology

As stated in the beginning of this paper our goals of the intended research is to develop a theory of syntactic pattern recognition through techniques from (1) grammatical analysis and (2) cellular automata analysis. We intend to establish the following as tentative subgoals (1) to give a formal definition of "syntactic pattern recognition" (2) to minimize the recognition (parsing) time with minimum errors (3) to solve miscellansous open problems cited in Chapter 5 and (4) from syntactic pattern recognition to semantic pattern recognition.

Based on our discussions and the general goals, our future research in this field can be described as follows :

(1) Given a class of sample array languages as input data find an array grammar that generates them. This includes many subresearch such as (a) Find the required minimum size of the sample language, the grammar from which can generate the whole set of language. For instance, in Example 4.1 if the grammar derived from any six strings out of the 7-tuples can not generate the whole set $\left\{ca^nb,\ b^2a^nb \mid n \geqslant 0\right\}$ then the 7-tuple is the required minimum set. (b) Put the grammar in the programmed form as discussed in §2.2 and if the associated probabilistic distribution set is given (as in Chapter 4) put it in the stochastic programmed form so that the grammar may be in the most simplified compact form. (c) Since the results derived from a grammar are always quite different depending upon whether it is applied parallely or sequentially or even parsingly (as discussed in §2.1 and §2.4) different grammars (for generation or

parsing, parallel or sequential) can be designed and they should be compared from speed point of view (number of steps) (d) Classify the language as discussed in §2.2 or find a hierachical structure for array grammars similarly as in §2.1[13].

(2) From the array grammar obtain its corresponding array automata or Turing acceptors (§3.3) that can recognize the array language. In this step we may design a stochastic experiment (§2.4) to test the confidence level and the recognition time, trying to enhance the confidence level (by adjusting λ [23]) and minimize the recognition time (by adjusting the grammar).

(3) Either from step (2) (see §3.3 [36]) or directly from step (1) (via Pre-Theorem of §3.3 and a result from Smith [60] which says that domains $\{(0,0)\}$, $\{(0,0).(0,1)\}$, and $\{(0,0),(0,-1)\}$ suffice for the productions of the class of array grammars.) we can find its corresponding cellular automata (§3.3, §3.4) that recognize the array languages. We can also compare the results from both ways and find whether there is any difference. Here miscellaneous open problems may be encountered as discussed in Chapter 5.

(4) Finally we should compare the results of step (2) with step (3) and see in which one the recognition speed is faster. It is expected that cellular automata are faster than array automata (§3.4), but for the stochastic and deterministic cases [58] we are still not quite sure. Until then, a more complete model for syntactic pattern recognition may then be established and a new trial effort from syntactic pattern recognition to semantic pattern recognition will be pioneered if proper "meanings" are

assigned to each syntactic structure [5].

The future research described above can roughly be sketched as in the diagram shown in next page.

From the general survey, discussions and future research described above we have introduced some formalism in our research for the concept of syntactic pattern recognition by "stochastic", "programmed", "array", "grammars", "grammatical inference" and "cellular automata". This will be the basic spirit of our methodology which will include techniques from probability theory (statistics), programming languages (FORTRAN V, SNOBOL etc), mathematical logic (formal theory, 1st order predicate calculus), modern algebra (group theory, lattice, homomorphism etc) and automata theory (with their relations to formal languages) and so on.

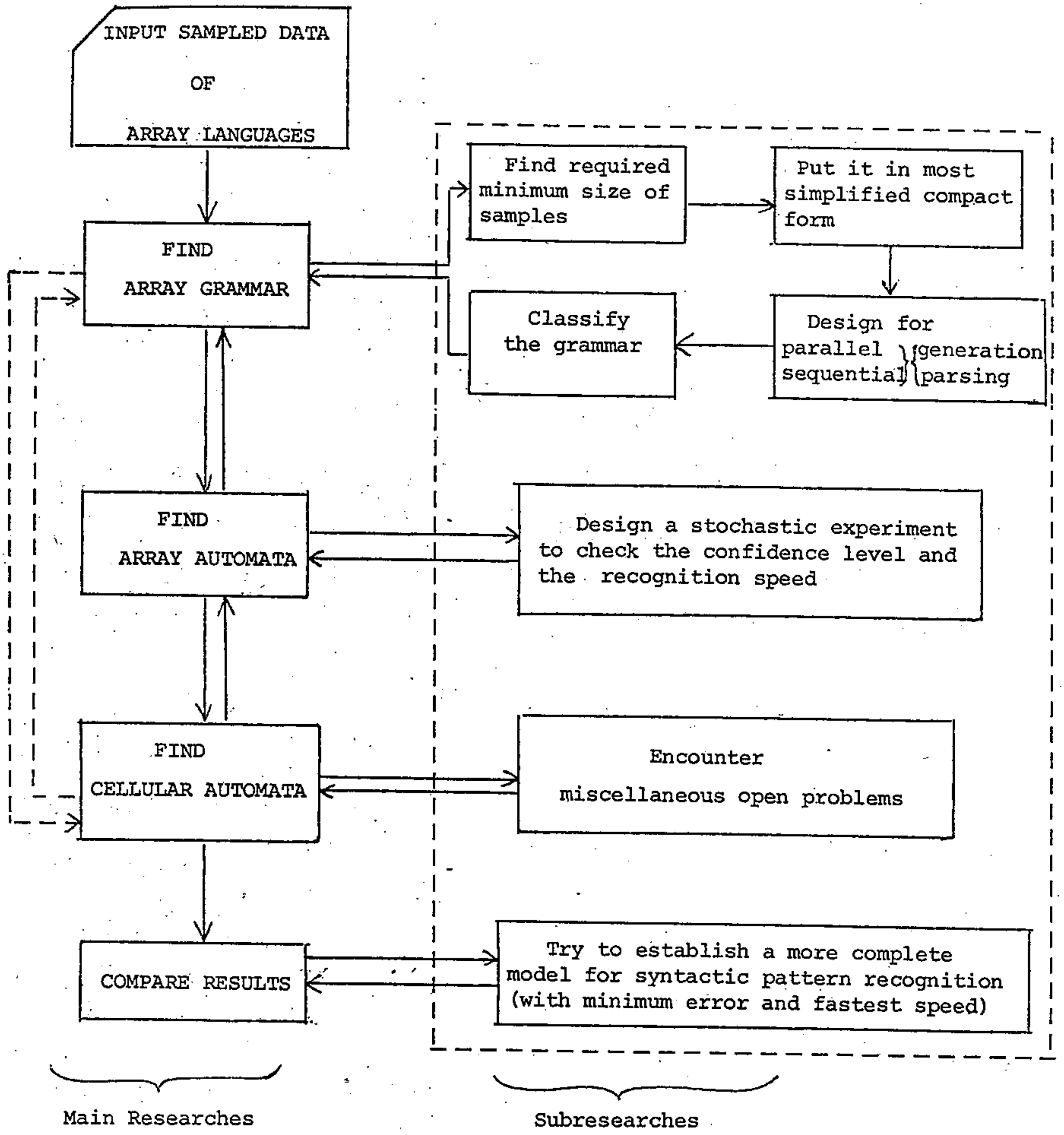


Fig 6.1 Sketch Diagram of the Future Research (note that the upward directed arrows mean that a reverse process will also be very interesting research topic.)

ANNOTATED BIBLIOGRAPHY

[1] Abe, N. et al "Web Grammars and Several Graphs" J. Computer and System Sciences 7 (1973), 37-65
Concerned with the class of web grammars introduced by Phaltz, Rosenfeld and Montanari.

[2] Abraham, S. "Compound and Serial Grammars" Information and Control 20 (1972), 432-438
Two new classes of generative grammars are defined, compound grammars and serial grammars. The generative power of them consisting of finite-state grammar is investigated.

[3] ________"Some Questions of Phrase Structure Grammars I" Comput. Linguistics 4 (1965), 61-70
Special interest was shown towards generative grammars and a new type of generative grammar called "matrix grammar", which can be considered as a programmed grammar with only success branch field.

[4] Aho, A.V. "Indexed Grammars - An Extension of Context-Free Grammars" IEEE pub. no. 16-C-56 (1967)
Attempting to define a class of languages in the region of unconditional transfer context-free programmed grammars (utcfpg's) with identical success and failure fields.

[5] Aho, A.V. and Ullman, J.D. The Theory of Parsing, Translation and Compiling Prentice-Hall ed. (1973), vol 1 and 2
Developed the relevant parts of mathematics and language theory and developed the principal methods of fast syntactic analysis.

[6] Amoroso, S. and Patt, Y.N. "Decision Procedures for Surjectivity and Injectivity of Parallel Maps for Tessellation Structures" J. Computer and System Sciences 6 (1972), 448-464
Demonstrate the existence of nontrivial injective parallel maps, which appears to be quite rare.

[7] Arbib, M.A. "Simple Self-Reproducing Universal Automata "Information and Control 9 (1966), 177-189
Present a self-reproducing universal array with simple programming. This is made possible by using as basic unit a finite automaton which can excute an internal program of up to 20 instructions.

[8] Beyer, T. "Recognition of Topological Invariants by Iterative Arrays" Ph.D dissertation, MIT (1970)
Propose iterative acceptors that can recognize topological patterns with real memory requirements.

[9] Birmann, A. W. and Feldman, J.A. "A Survey of Results in Grammatical Inference" Frontiers of Pattern Recognition, Watanbe ed. Hawaii (1972), 31-54.
Describe some of the major developing results on grammatical inference and indicate where the interested reader can look further.

[10] Booth, R.L. "Probabilistic Representation of Formal Languages" IEEE Symposium on Switching and Automata Theory 10 (1969), 74-81
A necessary condition for "consistency" of stochastic production rules is derived.

[11] Burks, A.W. Essays on Cellular Automata U. of Illinois Press (1970)
A collection of 15 essays of cellular automata edited by Burks in memory of Von Neumann. Most of them are concerned about the theory of self-reproducing automata.

[12] Chang, S.K. "A Method for the Structural Analysis of 2-Dimensional Mathematical Expressions" Information Sciences 2 (1970), 253-272
A structure specification scheme is described which can be used to specify the structures of certain 2-dimensional patterns. Algorithms are developed to test whether a pattern has a well formed structure with respect to a given structure specification scheme.

[13] __________ "Picture Processing Grammars and its Applications" Information Sciences 3 (1971), 121-148
A method for the description of the hierarchical structure of 2-dimensional pictures is proposed. The model is called picture-processing grammar.

[14] Chomsky, N. "Formal Properties of Grammars" Handbook of Mathematical Psychology John Wiley and Sons, Inc., New York, 2 (1963), 323-418
The closure properties of languages are discussed.

[15] Codd, E.F. Cellular Automata Academic Press, N.Y. (1968)
Helpful to computer designers and programmers who want a better understanding of the principles of homogeneous cellular systems, to automata theoreticians, and to biochemists interested in the possibility of biochemical computers with self-reproducing capability.

[16] Cook, C.R. and Wang, P.S.P. "A Chomsky Hierarchy of Isotonic Array Grammars and Languages" Computer Graphics and Image Processing 8, (1978) pp. 144-152.
Complete the Chomsky hierarchy of isotonic array grammars. Obtain Chomsky and Greibach normal forms for context-free array grammars. Surprisingly, the famous context-free languages $\{a^n b^n | n>1\}$ is not isotonic context-free, but isotonic context-sensitive.

[17] Dacey, M.F. "A 2-Dimensional Language for a Class of Polygons" Pattern Recognition Pergamon Press 3 (1971), 197-208
Poly is a 2-dimensional language the produces line pictures of polygons that may be decomposed into 45° right triangles and rectangles. The elements of the language are described and an example illustrates the use of the languag to produce a picture.

[18] Eickel, J. and Loeckx, J. "The Relation between Derivations and Syntactical Structures in Phrase-Structure Grammars" J. Computer and System Sciences 6 (1972), 267-282
Conditions for a phrase-structure grammar are established which warrant that any of its derivations univocally defines a syntactical structure of the sentence.

[19] Evans, T.G. "Grammatical Inference Techniques in Pattern Analysis" Tou ed. Software Engineering 2 Academic Press N.Y (1971), 183-262
Describe some recent studies of methods for the automatic inference of pattern grammars.

[20] Fischer, M.J. "Two Characterizations of the Contect-Sensitive Languages" IEEE Symposium on Switching and Automata Theory 10 (1969), 149-156
Two dimensional bugs define precisely the context-sensitive languages. Consider also finite state acceptors with n two-way non-writing input tapes.

[21] Fu, F.S. "On Syntactic Pattern Recognition and Stochastic Languages" Frontiers of Pattern Recognition Watanabe ed. (1971), 113-137
Stochastic languages and their applications to syntactic pattern recognition are described. An algorithm in grammatical inference for discovering such languages is also studied.

[22] ________ and Huang, T. "On Stochastic Context-Free Language" Information Sciences 3 (1971), 201-224
Properties of normalized stochastic languages are discussed and alternative procedures for constructing the Chomsky and Greibach normal forms for normalized stochastic context-free grammar (nscfg) are presented.

[23] _____ and Li, T.J. "On Stochastic Automata and Languages" Information Sciences 1 (1969), 403-419
For the -stochastic languages, a flexible stochastic experimental procedure based on Chebyshev's inequality is proposed. Then, the class of maximum-likelihood stochastic language is defined on the basis of the maximum-likelihood final-state distribution.

[24] _____ and Bharat, K. "Tree Systems for Syntactic Pattern Recognition" IEEETC, vol c-22 No. 12 (1973), pp 1087-1099
An approach of representing patterns by trees is described. Tree grammars are used for pattern description, and tree automata are used for classification. Illustrative examples are given.

[25] _____ and Lu, S. "A Clustering Procedure for Syntactic Patterns" IEEETSMC, vol SMC-7, No. 10 (1977), pp 734-742
A distance between two syntactic patterns is defined in terms of error transformations. A clustering procedure for syntactic patterns is also described together with a character recognition experiment illustration

[26] Hartmanis, J. and Stearns, R.E. "On the Computational Complexity of Algorithm" Trans. Amer. Math. Soc. 117 (1965), 285-306
Present a diagonalization proof techniques trying to solve some non-linear recognition time problems.

[27] Hopcroft, J.E. and Ullman, J.D. Formal Languages and Their Relations to Automata Addison-Wesley Press (1969)
Presents the theory of the formal languages as a coherent theory and makes explicit its relationship to automata.

[28] Horning, J.J. "A Procedure for Grammatical Inference" Information Processing (1971), 519-523

Exhibit a procedure which is possibly an optimum solution to the grammatical inference problems, and discuss a number of variations, including the learning of probabilities as well as grammars, and learning in the presence of noise.

[29] Ibara, O.H. "Simple Matrix Grammars" Information and Control 17 (1970) 354-394
Simple matrix languages and right-linear simple matrix languages defined. The closure properties of the bounded languages are also discussed.

[30] Kain, R.Y. Automata Theory: Machines and Languages Mc-Graw Hill Press (1972)
An extinguished textbook containing mathematical linguistic, turing machines, miscellaneous automata and formal language theories etc.

[31] Kanal, L. and Chandesekaran, B. "On Linguistic, Statistic, and mixed Models for Pattern Recognition" Frontiers of Pattern Recognition, Watanbe ed. (1972), 163-192
Present a selective discussion of some aspects of linguistics, statistical and mixed approaches of pattern recognition.

[32] Kasvand, T. "Experiments with an On-Line Picture Language" ditto
A preliminary version of the on-line picture language has been completed.

[33] Kirsch, R.A. "Computer Interpretation of English Text and Picture Patterns" IEEE Tran. 13 (1964), 363-376
Consider a class of information sources consisting of English text and pictures with an illustration of al algorithm for matching the sentences given by a simple grammar against the class of simple pictures which these sentences support to describe.

[34] Lee, H.C. and Fu, K.S. "A Stochastic Syntax Analysis Procedure and Its Application to Pattern Classification" IEEE Tran. on Computers c-21 7 (1972), 660-666
A Procedure is described for stochastic syntax analysis of context-free languages. The parsing algorithm utilizes a bottom-up technique used in procedure languages. All decision encountered in the algorithm are determined statistically. Also developed is a stochastic context-free language for use in pattern classification of chromosome images.

[35] Meyer, P.L. Introductory Probability and Statistical Applications Addison-Wesley Press (1965)
A textbook: Precise mathematical language is used and some concepts, definitions as well as theorems are discussed.

[36] Milgram, D.L. and Rosenfeld, A. "Array Automata and Array Grammars" Information Processing (1971), 69-74
It is shown that grammars that rewrite arrays are equivalent to Turing machines having array "tapes" and that "monotonic" array grammars (in which arrays never shrink in the course of a derivation) are equivalent to "array-bounded" machines.

[37] Miller, W.F. and Shaw, A.C. "Linguistic Methods in Picture Processing" Proc. AFIPS, FJCC 33 (1968), 279-290
Survey research in linguistic methods for describing and processing pictures. Also extract some common features and difficulties and indicate directions for future research.

[38] Mondelson, E. Introduction to Mathematical Logic D. van Nostrand, N.Y. (1966)
A very nice textbook containing propositional calculus, quantitative theory etc.

[39] Narasimhan, R. "Syntax-Directed Interpretation of Classes of Pictures" C. ACM 9 March (1966), 166-173
Describe the structure of syntactic descriptive models by considering their specific application to bubble chamber pictures.

[40] Nasu, and Honda, "Fuzzy Events Realized by Finite Probabilistic Automata" Information and Control 12 (1968), 284-303
Concept of probabilistic events is introduced and their closure properties under the operations on the set of all fuzzy events is studied.

[41] Nilsson, J.N. Learning Machines McGraw-Hill Press (1965)
Present some of the results of research in the new and exciting field of the learning machines. The basic approach adopted in this book involves the concept of discriminant functions that define the behavior of the pattern-classifying machine.

[42] Nishio, H. "Heuristic Use of Image Processing Technique for Theoretical Studies of Automata" Frontiers of Pattern Recognition, Watanabe ed. Hawaii (1972), 373-388
Problems of information transmission in cellular automata and pattern recognition are studied.

[43] Ohmon, K. et al "An Application of Cellular Logic for High Speed Decoding of Minimum-Redundancy Codes" FJCC (1972), 345-351
Present a new high speed decoding system consisting of cellular logic which have such merits as high decoding speed, design simplicity and case of machine fault detection.

[44] Page, C.B. "Strong Stability Problems for Probabilistic Sequential Machines" Information and Control 15 (1969), 487-509
The strong stability problem is studied for the behavioral equivalences indistinguishability and n-moment equivalence.

[45] Paz, A. Introduction to Probabilistic Automata Academic Press (1971)
Serve both as a monograph and as a textbook with a large collection of exercises distributed among the various sections including: stochastic sequential machines, Markov chains, events, languages and acceptors, etc.

[46] Pfaltz, J.L. "Web Grammars and Picture Description" TR-70-138, Computer Science Center, U. of Maryland, Sept. (1970)
A method of analyzing pictures of "neurons" is described. Creates a data structure to represent the given picture and uses this structure to direct array-processing operations which provide the information model for the analysis.

[47] Reghizzi, S.C. "An Effective Model for Grammar Inference" Information Processing (1971), 524-529
A grammar inference model is a black box which receives primary linguistic data, and produces a grammar. The effectiveness of a computer program based on the algorithm opens the way to several potential applications and confirms some assumptions concerning natural language acquisition.

[48] Richardson, D. "Tessellation with Local Transformations" J. Computer and System Sciences 6 (1972), 373-388
Contributions to the characterization of continuity and translation invariance and processes which can occur in a tessellation universe in which the "law of the nature" is a local transformation.

[49] Rosenberg, G. "Direction Controlled Programmed Grammars" Acta Informatica 1 (1972), 242-252
A generalization of the notion of a context-free grammar is presented. It is based on the notion of a programmed grammar.

[50] Rosenfeld, A. "A Grammars for Maps" Software Engineering 2 Tou ed. (1971) 227-239
Describe a grammar that has the set of "adjacency multigraphs" as its language.

[51] Rosenfeld, A. "Isotonic Grammars, Parallel Grammars, and Picture Grammars" Machine Intelligence 4 (1971), 281-294
Isotonic context-sensitive array rewriting rules are essentially the same as local digital picture processing operations. Since the latter are often applied to pictures in parallel, it is of interest to study grammars which operate in parallel.

[52] Rosenfeld, A. and Mercer, A. "An Array Grammar Programming System" C.ACM 16,1 (1973), 299-305
A package of Fortran programs has been developed that permits a user to interactively design and test array grammars. Examples are given involving array languages consisting of simple geometrical patterns, as well as a language of "neuron pictures".

[53] __________ and Pfaltz, J.L. "Sequential Operations an Digital Picture Processing" J.ACM 13(1966), 471-494
The relative merits of performing local operations on a digitized picture in parallel or sequentially are discussed. Some applications of the connected component and distance functions are also presented.

[54] Rosenkrantz, D.J. "Programmed Grammars and Classes of Formal Languages" J.ACM 16 No. 1 (1969), 107-131
The proporties of programmed grammars, a generalization of phrase structure grammars, with various types of production cores are investigated. The new classes of grammars are defined which lie between the context-free and context-sensitive grammars in their generative power.

[55] Solomaa, A. "Matrix Grammars with a Leftmost Restriction" Information and Control 20 (1972), 143-149
The family of languages generated by matrix grammars with context-free (λ free) core productions and with a leftmost restriction on derivations equals the family of recursively (context-sensitive) languages.

[56] Siromoney, G. et al "Picture Languages with Array Rewriting Rules" Information and Control 22 (1973), 447-470
Generative models of picture languages with array rewriting rules are presented. A distinct hierarchy is chosen to exit between the different classes introduced.

[57] Smith, R.A. "Cellular Automata and Formal Languages" IEEE Symo. on Switching and Automata Theory 11 (1970), 216-224
Cellular automata are shown to be inherently faster than iterative acceptors. Many positive results are presented to indicate that the context-free languages can, perhaps, be accepted in time n and space n by cellular automata.

[58] ______ "Cellular Automata Complexity Trade-Offs" Information and Control 18 (1971), 466-482
The general theory of cellular automata is investigated with special attention to structural complexity.

[59] ______ "Real-Time Language Recognition by One-Dimensional Cellular Automata" J. Computer and System Sciences 6 (1972), 233-253
Pattern Recognition by parallel devices is investigated by studying the formal language recognition capabilities of 1-d cellular automata. Closure properties and cellular automata transformation lemmas are also presented.

[60] ______ "Two-Dimensional Formal Languages and Pattern Recognition by Cellular Automata" IEEE Symp. on Switching and Automata Theory 12 (1971), 144-152
A formal study of pattern recognition capabilities of cellular automata is undertaken based on a class of recently introduced grammars for 2-D array grammars. Also solve an open problem of Beyer.

[61] Stanat, D.F. "A Homomorphism Theorem for weighted Context-Free Grammars" J. Computer and System Sciences 6 (1972), 217-232
Exhibit a form of the theorem using a Greibach normal form and allowing weighted productions.

[62] Swain, P.H. and Fu, K.S. "On Syntactic Pattern Recognition" Software Engineering 2 Tou ed. (1971), 155-182
A review and evaluation of the state of the art of syntactic pattern recognition.

[63] ______ "Stochastic Programmed Grammars for Syntactic Pattern Recognition" Pattern Recognition Pergamon Press 4 (1972) 83-100
A stochastic version of the programmed grammars is proposed as a powerful and convenient formalism for syntactic pattern recognition. An algorithm for parsing strings generated by stochastic context-free programmed grammars is described and an example is presented which generates noisy squares.

[64] Thatcher, J.W. "Self-Describing Turing Machines and Self-Reproducing Cellular Automata" from Essays on Cellular Automata, Burks ed. (1969), 103-131
Provide an elementary introduction to some fascinating, as well as important aspects of automata theory. Describe a basic note of universality in the von Neumann cellular automata.

[65] Turakainan, P. "On Stochastic Languages" Information and Control 12 (1968), 303-313
The notion of a probabilistic automaton is generalized, and it is shown that this does not affect the family of representable languages.

[66] von Neumann, J. Theory of Self-Reproducing Automata, Burks ed. U. of Illinois Press, Urbana (1966)
An algorithm for determining a configuration of self-reproducing cellular model is designed.

[67] Wang, P. "Sequential/Parallel Matrix Array Languages," J. of Cybernetics, vol. 5.4 (1975) pp. 19-36
Sequential/parallel technique for two-dimensional pattern generation is introduced. Several properties are also investigated.

[68] __________ "Recognition of Two-Dimensional Patterns" Proc. ACM '77, Seattle Wa (1977), pp. 484-489
A method for recognizing two-dimensional sequential/parallel matrix languages is presented.

[69] Watanabe, S. Methodologies of Pattern Recognition Academic Press (1969)
Written by 29 outstanding authorities presently active in the field, place emphasis on the "philosophy" of his approach rather than math. deri.

[70] Yamada, H. "Tesselation Automata" Information and Control 14 (1969), 299-317
Certain mathematical studies of pattern recognition, evolution theories, and self-reproducing automata motivated the definition of the tessellation automaton which is a mathematical model of an infinite array of uniformly interconnected identical finite-state machines.

[71] __________ and Amoroso, S. "A Completeness Problem for Pattern Generation in Tessellation Automata" J. Computer and System Sciences 4 (1970), 137-176
Deals with the question of whether or not, for a given tessellation automaton, there exists a finite pattern that cannot evolve from a given primitive pattern no matter what sequence of environmental input transformations are applied.

[72] Zadeh, L.A. "Note on Fuzzy Languages" Information Sciences 1 (1969), 421-434
It is shown that any context- sensitive fuzzy grammar is recursive. For fuzzy context-free grammars, procedures for constructing the Chomsky and Greibach normal forms are outlined and illustrated by examples.

[73] __________ "Quantitative Fuzzy Semantics" Information Sciences 3 (1971), 159-176
Given a particular x in T, the membership function $\mu_L(x,y)$ defines a fuzzy set, M(x), in U whose membership function is given by $\mu_{M(x)}(y) = \mu_L(x,y)$ The fuzzy set M(x) is defined to be the meaning of the term x, with which playing the role of a name for M(x).

[74] __________ "Similarity Relations and Fuzzy odering" Information Sciences 3 (1971), 177-200
Various properties of similarity relations and fuzzy oderings are investigated and as an illustration, an extended version of Szpilrajn's theorem is proved.