PATTERN RECOGNITION USING SOME PRINCIPLES OF THE ORGANISM-ENVIRONMENT INTERACTION

J.F. O'Callaghan, B.E.(Hons.) Tas.

Submitted in partial fulfilment of the requirements for the Degree of Doctor of Philosophy in The Australian National University.

September, 1968

Department of Engineering Physics,
THE AUSTRALIAN NATIONAL UNIVERSITY,
CANBERRA, A.C.T.
DECLARATION

I declare that this Thesis reports my own original work, that no part of the Thesis has been accepted or presented for the award of any degree or diploma by any University, and that to the best of my knowledge the Thesis contains no material previously published or written by another person except where due reference is given to that author by direct credit in the text or in the bibliography.

John F. O'Callaghan,
CANBERRA
September, 1968.
PREFACE

This Thesis is primarily concerned with problems in the machine recognition of patterns, with special emphasis on the recognition of hand-drawn line figures. Because of the nature of the field and the approach taken by the author, the project has encompassed several disciplines including Computer Science, Cybernetics and Psychology.

The presented work was commenced at the University of Tasmania in March 1965. In March 1966, the author transferred to the Australian National University, where the project has been completed. Throughout this entire period, the work has been under the guidance of Dr. S. Kaneff, of The Australian National University.

As to the contents of the Thesis, Chapter 1 reviews overall trends in pattern recognition, outlining the reasons for the approach taken by the author. The approach is developed in Chapter 2, where various principles of the organism-environment interaction are mentioned in relation to how they might be incorporated into a recognition machine.

Chapter 3 describes the environment (for the programs), which is sequences of direction codes representing hand-drawn line figures.

Two series of computer programs have been developed from the general framework of the approach provided. The first series is presented in Chapters 4, 5 and 6 - the first of these contains the design of the programs, the second, the operation of the programs and their results, and the third, a discussion of the scheme and results, with suggested extensions and applications for the programs.
The second series is discussed in Chapters 7, 8 and 9 in the same manner as for the first series.

A general discussion, relating to the important features of the work, and the extension of the approach to incorporate other patterns, is presented in Chapter 10.

Appendix A contains a suggestion for a method to recognize handwriting, arising from the first series of programs. Appendices B and C each present details of computer programs, one from each series, respectively. Appendix D describes a method for 'on-line' input of the direction codes.

In comparison to other research published, the author's work is suggested to be novel in the following respects:-

(a) the outline of the principles, which as far as the author is aware, have not been discussed with the same bias, although various principles have been identified in a number of disciplines;

(b) the incorporation of some of these principles into (two series of) programs;

(c) the approach taken to character recognition in the first series of programs - the following features are important:-

   (i) the form of model,

   (ii) generation of the input description,

   (iii) the similarity measure, defined in terms of factors of the description,

   (iv) supervised learning to average irregularities from the model;

(d) the approach taken in the second series of programs - this includes:-
(i) application of a coding method, which codes regularities in an input and then measures the reduction in coding as a basis for decision-making,

(ii) the generation (and 'averaging') of patterns in the stored model.

(e) preliminary investigation into a method for the recognition of handwriting.

A paper written by the author, has been published in the International Journal of Control, Vol. 5, No. 4, 1967, pp. 297-305, entitled "A General Approach to the Machine Recognition of Patterns" - a copy is contained in the flap of the back cover of the Thesis. The ideas reported in the paper have been subsequently revised in Chapters 2 and 4.
ACKNOWLEDGEMENTS

The author wishes to acknowledge and express thanks to his supervisor, Dr. S. Kaneff, of the Department of Engineering Physics, The Australian National University, for valuable discussions and guidance.

Appreciation is expressed to Mr. J. Beresford, of the Department of Electrical Engineering, University of Tasmania, and Professor G. Newstead of the Department of Engineering Physics, The Australian National University, for allowing the author to work in their Departments.

The author is indebted to the Postmaster-General's Department for the granting of leave in order to perform the research, and to the Commonwealth Scholarship Board for financial assistance.
## CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Summary</td>
<td>xi</td>
</tr>
<tr>
<td>1. INTRODUCTION</td>
<td>1</td>
</tr>
<tr>
<td>2. THE OVERALL APPROACH TO PATTERN RECOGNITION</td>
<td>10</td>
</tr>
<tr>
<td>2.1. Outline of the Approach</td>
<td>10</td>
</tr>
<tr>
<td>2.1.1. The Environment</td>
<td>11</td>
</tr>
<tr>
<td>2.1.2. The Goal</td>
<td>12</td>
</tr>
<tr>
<td>2.1.3. Processing the Input</td>
<td>13</td>
</tr>
<tr>
<td>2.1.4. The Model</td>
<td>16</td>
</tr>
<tr>
<td>2.1.5. Use of the Model</td>
<td>20</td>
</tr>
<tr>
<td>2.1.6. Feedback</td>
<td>23</td>
</tr>
<tr>
<td>2.1.7. Summary of the Principles</td>
<td>23</td>
</tr>
<tr>
<td>2.2. The Nature of the Approach</td>
<td>23</td>
</tr>
<tr>
<td>2.2.1. Comparison with Similar Research</td>
<td>23</td>
</tr>
<tr>
<td>2.2.2. Relation of the Present Approach to Other Fields</td>
<td>31</td>
</tr>
<tr>
<td>2.3. The Attitude to Program Design</td>
<td>32</td>
</tr>
<tr>
<td>3. THE INPUT SEQUENCES</td>
<td>35</td>
</tr>
<tr>
<td>3.1. Construction of the Sequences</td>
<td>35</td>
</tr>
<tr>
<td>3.2. Comments on the Tests and Programs</td>
<td>38</td>
</tr>
<tr>
<td>4. THE APPROACH TO THE FIRST SERIES OF PROGRAMS</td>
<td>43</td>
</tr>
<tr>
<td>4.1. The Model for the Programs</td>
<td>43</td>
</tr>
<tr>
<td>4.2. Processing the Input</td>
<td>49</td>
</tr>
<tr>
<td>4.3. The Decision</td>
<td>53</td>
</tr>
<tr>
<td>4.4. Learning</td>
<td>58</td>
</tr>
</tbody>
</table>
5. PROGRAMS IN THE FIRST SERIES - INPUTS TREATED AS REPRESENTING LINE DRAWINGS

5.1. Introduction to the First Program, AM
   5.1.1. Storage
   5.1.2. Prediction
   5.1.3. Formation of Parts
   5.1.4. Comparison
   5.1.5. Updating the AV

5.2. Modification to the First Program, AN
   5.2.1. Orientation Independence
      5.2.1.1. Orientation Independent Program, GA
      5.2.1.2. Orientation Independent Program, AN
   5.2.2. Segmentation
   5.2.3. Tolerance
   5.2.4. Confirmation Measure
      5.2.4.1. Size
      5.2.4.2. Orientation
      5.2.4.3. Length
   5.2.5. Averaging
   5.2.6. Curve Determination in the Current Model

5.3. Program AN6
   5.3.1. Comparative Performances of Programs
   5.3.2. Results of Numeral Recognition
5.3.3. Causes of Error
  5.3.3.1. Tolerance on the Curve Type
  5.3.3.2. Hypothesis Purging
  5.3.3.3. Similarity Measure
  5.3.4. Similarity Values
  5.3.5. Learning

5.4. Summary

6. DISCUSSION OF THE PROGRAMS OF CHAPTER 5
  6.1. General Remarks
  6.2. Recognition Applications
  6.3. Use in Interpretive Situations
  6.4. Features of the First Series of Programs

7. THE APPROACH TO THE SECOND SERIES OF PROGRAMS
  7.1. General Considerations
  7.2. The Model of the Sequences
  7.3. Model Generation
  7.4. The Decision
  7.5. Learning
  7.6. Summary

8. PROGRAMS IN THE SECOND SERIES - INPUTS TREATED AS SEQUENCES OF ABSTRACT ELEMENTS
  8.1. Operation of the Program, EC
    8.1.1. Extraction of the Model from Initial Examples
10.2. The First Series of Programs and Their Extensions

10.3. The Second Series of Programs and Their Extensions

10.4. The Approach Applied to Other Problem Areas

11. CONCLUSIONS

BIBLIOGRAPHY AND SELECTED REFERENCES

APPENDIX A A SUGGESTION FOR A PROGRAM TO RECOGNIZE HANDWRITING

A.1. Description of the Scheme

A.2. An Example of the Operation of the Scheme

A.3. Discussion

APPENDIX B DETAILS OF PROGRAM AN6

APPENDIX C DETAILS OF PROGRAM EF

APPENDIX D DETAILS OF AN ON-LINE SYSTEM FOR INPUTTING THE DIRECTION CODES

D.1. The Equipment

D.2. Operation of the System

D.3. Comments on the Operation

D.4. Conclusion
SUMMARY

An approach is presented to the machine recognition of patterns, suggested by considering the part pattern recognition plays in the life of an organism in its interaction with the environment. The operation of two series of computer programs incorporating the principles derived from this approach, and results obtained from the programs for the recognition of number sequences, representing line drawings, are given. The programs in one series are successful in recognizing user-specified figures under a variety of distortions, and are relatively independent of size and orientation. The programs in the second series recognize sequences with the restriction of focusing on only internal relations between elements of the sequences. Suggestions are made for various applications of the basic method, including the recognition of line drawings on a graphic display, the generation of a description of line drawings in a readily understood manner for a user, and the simulation of human behaviour in a certain concept learning experiment.
Chapter 1
INTRODUCTION

Human beings and lower level organisms are capable of a wide variety of behaviour, which can be termed 'intelligent'. Various animals have been trained to discriminate certain shapes and sounds; a child can learn to speak a language; adolescents learn to make valid inferences from a given body of data. The suggestion of mechanizing these operations has aroused considerable interest in recent years. Technological advances in computers and progress in various fields such as automata theory, cybernetics and information theory have undoubtedly 'set the stage' for attacking these problems. Attempts to program machines to perform some of these processes have nevertheless proven to be quite difficult.

The desire to mechanize stems from three main areas:-

(a) **Behavioural Science**: where the interest is in studying the behaviour of organisms, testing various theories by simulation on a computer.

(b) **Neurophysiology**: where an understanding of the physical operation of the brain - as a collection of neurons - is attempted by constructing or simulating the postulated nerve nets.

(c) **Artificial Intelligence**: where the main goal is to construct machines - possibly but not necessarily operating on principles similar to those suggested in (a) and (b) above - which are an efficient and economical solution to a given task.

The research reported in this Thesis concentrates on the last approach. The divisions given above are not at all well-defined; many workers may stress for example the simulation aspect of their devices, while also stating that
they have practical uses. While this Thesis does not present such an extreme, it has indeed been influenced by work in psychology and computer simulation.

The development of mechanized procedures to achieve the recognition performance of the human being has been one of the major areas for research in Artificial Intelligence. The field commonly termed 'Pattern Recognition' encompasses an ill-defined region involving many suggested techniques for a wide variety of patterns. A complete survey is beyond the scope of the Thesis; rather, this Chapter will briefly trace developments in the field, (with reference mainly to review articles), in order to outline the reasons for the approach taken in this project.

It is difficult to decide upon a time at which pattern recognition research commenced. Some of the mechanization problems in the early 1950's obviously generated interest in this area. The desire to have efficient input-output devices for computers and the need for reading machines in business and industry has led to the development of character readers and to efforts to create facilities for accepting speech, (196). Another problem has concerned the economical transmission of data, where pattern recognition procedures can be employed to remove redundant parts of the message. This problem is related to satellite communication and television bandwidth reduction, (54). Several problems of detecting signals in noise such as electroencephalograph, electrocardiograph and seismic records have been suggested, (120). There are some areas, such as medical diagnosis by computer, which have developed as an offshoot from the classification field. The emphasis on recognition, in particular, has probably been due to its important role in the behaviour of an
organism, and many have suggested that the developed techniques would be applicable to a wide variety of problems, (174).

When the difficulty of the problem became apparent in the late 1950's, there were some attempts to limit the possible variations in the pattern set. This could be adopted for some applications in character recognition by devising a special type-writer font,(easily recognizable by a machine but nevertheless readable by human beings), or applying constraints on the user. The former approach led to the magnetic ink character readers, which have been commercially available for some time, (27). In the second class are some of the optical characters readers, now available, capable of reading several fonts of typewritten material or, in some cases, well-formed handprinting. For all machines, however, there are strict tolerances on the print quality and location of the material, (27,59).

The techniques implemented in these devices involve the search for a set of well-defined properties or templates. In the magnetic ink readers, the quantized electrical signals, produced from sensing the strength of magnetization, provide the features. Most optical readers employ a type of 'mask' which is placed on the input, to determine a feature, (59). The set obtained is sequentially matched, by a 'decision tree' method, to the known standards for each character. In the more sophisticated machines, some tolerance on the match is allowed.

The general form of a recognition device has been considered to involve two parts - the receptor and the categorizer, (132). This subdivision is shown schematically in Figure 1.1. The former provides a set of 'n' features; n is a numerical value which may be small or large depending
Receptor Categorizer

Figure 1.1. The Classificatory Approach to Recognition.
on the problem. There have been many mathematical, mainly statistical, techniques applied to the design of the second part. The task of the categorizer is to partition the n-dimensional space, so formed from the receptor's measurements, into the required classes. The optimum decision function is, in general, not practically realizable and some assumptions must be made. These usually involve either,

(a) the distribution of the measurements from the receptor;

or

(b) the structure of the categorizer.

In the first situation, a given set of samples allows an approximation to be made to the underlying distribution, either by analytic or non-analytic means, (175). These latter means are used to provide an efficient procedure for the approximation as, for example, Sebestyen's adaptive sample set construction method, (175). The area of automatic classification, closely related to pattern recognition, often incorporates such techniques - an example, is cluster analysis, (143).

In the second situation, the type of surface to be employed in delineating the classes is defined. Learning samples are employed to calculate its exact form. The most common surface is the hyperplane, (89), but other non-linear forms have been implemented, (175). Methods for training units employing these functions as in the Perceptron, (167), and other networks, (154), have received a considerable amount of attention.

These statistical techniques have been applied to many problems, including character and speech recognition and, recently, recognition in control systems, (222). Other mathematical techniques such as moments, (6), auto-correlation, (40), and mathematical programming,
have also been the subject of experiments. Several workers have attempted to relate certain theories such as set theory, to the problem.

In contrast to the research on the categorizer, there has been little done on the receptor. The design of features to characterize the various classes is still a problem for the ingenuity of the designer. There are two main approaches to the problem:

(a) to reduce the dimensionality of the input measurements by a suitable performance criterion,
(b) to extract specific features from the input.

The first approach involves the defining of a suitable criterion; some criteria have been suggested in terms of information-theoretic measures. The second approach is often applied in two ways. One involves the combination of matrix cells (or n-tuples) in the input, while the other involves analogue features, such as straight lines and cusps. The decision as to which is better depends on the nature of the problem. In some programs, the machine is capable of generating and selecting its own features from a prescribed form. There are other techniques for defining the input - often ad hoc procedures.

Pattern recognition is a problem which pervades many areas; some of the more complex ones are fields of research in their own right. These include:
- pictorial data analysis,
- language processing and information retrieval,
- automated teaching,
- problem solving (game theory, theorem proving).

In these cases the problem of pattern recognition is not simply "identifying and assigning patterns to classes", but...
The inputs are usually so complex that a set of unrelated features is inadequate to characterize the class—consider, for example, the specification of features for the class of well-formed English sentences. As exemplified by pictorial diagrams, the input involves not only parts of figures, but also relations between these parts. These relations must be extracted in the receptor. Again a machine designed to classify single characters may not cope with an input containing a set of overlapping figures.

Many of the problems in the above areas involve some form of man-machine communication. This is an important and growing field, as evidenced by the accelerated use of graphic displays. It is desirable in such circumstances for a machine to produce efficiently a readily understood description of each input. The extent to which it can do this depends mainly on the machine's ability to extract and organize relevant information from the input. For example, one may want the machine to describe,

(a) an object, which may occur in any orientation in a photograph;
(b) the structure of a sentence, from which the meaning of the sentence can be extracted;
(c) a student, from the set of answers he gives to the machine's questions.

As Narasimhan points out in his field of bubble chamber photograph analysis, "the aim of any adequate recognition procedure should be not merely to arrive at a 'yes', 'no', 'don't know' decision but to produce a structured description of the input", (147). This ability to give a description does not necessarily imply that the machine can classify the input; for example an input of a line-drawn figure may not be able to be placed in any known
figure category, but nevertheless it can be described in terms of similarities to known figures or in terms of line segments.

It was with this background that the project reported here was commenced. The 'broader look' at pattern recognition, which could almost be deemed a separate problem, meant that the nature of the receptor needed to be considered in more detail. With little evidence of any major 'break-through' in the field, the author took the approach of reverting to a study of the organism's recognition capabilities in an effort to define overall principles to incorporate into a recognition device. These principles and a comparison of the approach to others evident in various papers, will be discussed in Chapter 2. The nature of the approach has proved to be logical rather than statistical.

The particular problem which has been considered is the recognition of number sequences, representing a direction code for hand-drawn lines - explained in Chapter 3. This was chosen as a suitable vehicle for the study because of its simplicity and ease of generation. Furthermore, any machine designed to recognize these sequences has obvious extensions to the recognition of hand-drawn figures.

Two series of programs have been developed. One series is discussed in Chapters 4, 5 and 6; the first Chapter, 4, is a detailed discussion, the second explains the actual programs, modifications and results, and the third discusses possible extensions to the programs. The second series is presented in Chapters 7, 8 and 9 in the same manner as for the first. Chapter 10 gives a summary of the overall approach. Appendix A describes an application to handwriting recognition, devised during the project, and
Appendices B and C show the layout (in PL/1 language) of two programs (one from each series). Appendix D describes an on-line experiment, using one of the programs in the first series, to learn to recognize hand-drawn characters.
Chapter 2

THE OVERALL APPROACH TO PATTERN RECOGNITION

2.1. Outline of the Approach

The approach taken by the author is derived from consideration of the part pattern recognition plays in the life of the organism. The term 'organism' is (and has been) used here to denote any organized body possessing life; its use is similar to the way it is employed in psychology and cybernetics. For obvious reasons, the main organism with which this discussion is concerned is the human being.

An attempt will be made to highlight the important features of the organism (and its environment) possessing implication for the design of a machine to recognize patterns. The principles will not provide the details of such a machine; but they will suggest a framework in which one might proceed to devise them.

Most features have been mentioned often in other terms, in various disciplines. The following discussion will refer to these ideas, with a bias toward the possible incorporation of the features into a recognition machine. This approach has received little attention in the literature, although workers have commented on the need for additional requirements for a recognition device, (e.g. 24, 48, 141).

The discussion considers the organism as a system, following principles which have been emphasized in cybernetics, (219), and general systems theory, (213). However, the interest here is not directed towards physiology with which both disciplines have tended to concern themselves, (72). Information theory, (176), plays
an important role in the aforementioned disciplines. The term 'information', as selective information content, and the related terms of 'redundancy' and 'coding,' have been found useful in this Thesis, for discussing the organism as a system. Because the approach considers the activities of an organism, there are obvious connections with psychology. There has been no attempt to follow any theory rigorously, but alliances with certain ones can be identified.

The first part of this Chapter outlines the main features, explaining how they might be incorporated into a recognition machine. The second part compares this approach with other attempts at recognition and with other related areas in Artificial Intelligence.

2.1.1. The Environment

In order to avoid controversies concerning the nature of the environment, the term 'environment' shall refer here to the 'sensed data' received by the organism. The sensing occurs continually and in a variety of ways, such as by sight, touch and sound. Most organisms have different capabilities in their sensory apparatus, and are therefore limited to a certain degree in the information which can be extracted from the surroundings. For example, the human being cannot sense cosmic rays.

The design of a mechanical system can be such that its input, which is considered here to be continuous, (see Chapter 3), contains all the information required by the machine. That is, if a machine is required to recognize a house, then the input is designed to contain the necessary structure for a house to be extracted. The programs should include the necessary 'information' or abilities by which
the organization (e.g. the house) can be discovered in the input. It is further desired that the machine should be given as little 'innate' ability as possible, requiring the extraction of as much information as possible from the input data. The extent to which one can do this is a function of the environment.

The environment is assumed to be a sequence of discrete elements. This assumption allows the problem to be formulated and studied in simple terms. Provided the sampling occurs frequently enough for the necessary information to be 'transmitted', there is no generality lost by the simplification. The elements of the stream are, as for any recognition or perception device, symbols representing some facet of the real world.

That there is some redundancy (or order) in the input stream can be explained by considering the extreme conditions in which the environment is:

(a) Completely random - then one (as an organism) could never predict any event with success better than that of chance; (but one can in fact predict events).

(b) Completely redundant - then every event would be determined, and one would 'know' the future; (but this of course is not the situation).

Consequently, the environment must be (overall) partly redundant; obviously, there will exist 'pockets' which are more (or less) redundant than other parts.

This nature of the environment can equally well be explained in terms of constraints on the events in the system. Given a certain set of events, (e.g. black clouds), then there is a bias or constraint which tends to produce certain consequences, (e.g. rain).
The elements of the input sequence to the machine actually have numerical values, and form a partly redundant environment, (see Chapter 3).

2.1.2. The Goal

The organism is a system, characterized by its goal-directed behaviour, (128). Its main goal is that of survival, and many activities performed by the organism can be seen to be directly controlled by the seeking of a stable existence in a world which could destroy it. There are secondary goals such as attainment of pleasure, which a particular organism may be trying to achieve.

In the same way, the machine will have a basic drive, incorporated in the execution of the program, to reach its ultimate goal of correctly recognizing patterns in the input. There will be no measure of 'health' incorporated; although the degree of success in its recognition could be regarded as one. The machine is therefore basically a decision-making device, outputting the 'name' of any object (character) which is recognized in the input. Secondary goals exist in recognizing (and describing) the parts and relations which constitute the named object.

2.1.3. Processing the Input

Because the sensed data of the organism is partly redundant, certain groups of elements will be ordered, and hence will recur in the sequence. These groups are termed patterns, (170). The ordering is such that given some of the elements, two sets (of elements) can be formed for which:

(a) one contains elements which are more probable continuations of the given ones, than the other.
(b) as more elements are received, the size of the more
probable set decreases while the size of the other increases.

An important point about a pattern is the inherent notion of prediction (shown in (a)). The discovery of part of a pattern is useful to an organism for predicting future events in the environment. Events which are linked by a cause-effect relation exhibit this notion well. Detecting the presence of a cause, (e.g. black clouds), suggests the occurrence of the effect, (e.g. rain). The organism knowing this particular relation can take preventive action against the rain.

It is therefore possible in such an environment for an organism to 'come to know' the organization of its surroundings. In other words, it can discover the form of the constraints on the environmental system. In this situation, the process of learning is necessary for an organism to adapt its behaviour continually to cope with the uncertainty in the sensed data. It is through this process that the organism can, in an everchanging manner, come to know the environment, thereby increasing its chances of survival. The level of redundancy which exists (in the environment) at any stage can be thought of as the amount of knowledge the organism can attain, or as McKay puts it, "the scope for learning", (130).

The extraction of this order from the sensed data has an analogy in the ability of a person wishing to transmit a message of symbols economically. His aim is to remove the redundancy in the message by coding it into another form having less symbols, with the consequence of increasing the average information content per symbol. The efficiency of transmission depends on the capacity of the communication channel. The important point,
however, is that the organism can be regarded as performing a 'coding operation' on the input, allowing itself to store the message in a more compact manner. The conscious thought processes, in these terms, deal with a coded form of the sensed data.

There would appear to be an anomaly here, for the coded sequence, because of its decreased information capacity, is more random. However, the organism is coding the input with 'known' patterns, gained from its experience. In a subjective sense, (rather than the objective one of the communication analogy), it is the input which is random, because of its complexity to the perceiver, and it is the coded version which is ordered, because the symbols constituting this version are 'known'.

The removal of redundancy as an activity of the brain has been mentioned by workers in several fields. Barlow (14, 15) has presented evidence that the brain physically performs successive codings of the sensed data. In particular, he suggests that the economy is not in terms of the number of channels used, (as expected from Shannon's theory), but in the number of impulses transmitted along the nerve fibres. Attneave (10) explains how visual perception can be considered psychologically as a coding. The desire to search for homogeneity or order, can explain some of the Gestalt laws of perceptual organization. For example, the choice of a point to extrapolate a line ('good continuation') is usually given to that one which provides the simplest or most ordered description.

The organism behaves in other ways which can also be described in terms of coding. Miller (137) has explained the importance of coding in the remembering of certain information, such as long strings of numbers. Experienced
events in the environment are particularly useful codes; thus a certain building in a picture can be described verbally as 'my house'- a compact description of some of the pictorial information. As another example, methods of inductive reasoning are employed to suggest the form of a pattern. An hypothesis is an effective code explaining a set of circumstances in the real world.

The machine will in the same way, be considered as a coding device - continually searching for patterns. The input will therefore be described more economically (in storage) in terms of 'known' elements. Just as certain events can be anticipated by the organizer, so should the machine be capable of expecting certain elements in the input.

2.1.4. The Model

The organism possesses an inherent desire to discover patterns, in order to comprehend the environment. This discovery allows the organism even with a limited memory to remember some parts and aspects of the input stream, which may be useful in its search for a stable existence.

The ability to code and organize the information extracted from the sensed data, leads to the construction of classes of patterns. Two patterns may be considered equivalent or similar because the rules (or constraints) which order their elements can be effectively predicted in the same manner. Thus the class of circles can be formed in an environment of triangles and squares for the following reason: - given part of the circle, its continuation can be predicted better by a known circle (independent of size) than with the triangle or square. There are other ways in which a class may be formed -
one may simply focus on a certain property of an object and specify a class of all objects having this property, (e.g. all figures containing four right angles). However, it is the former method which the machine shall possess.

The extension of the class depends partly on the immediate goal of the organism and partly on the reward from the consequences of actions based on the classification. Obviously, the primary desire to distinguish mushrooms from poisonous toadstools is on the basis of their possible disastrous effects to someone who does not produce the appropriate classification. The description of an object may in other circumstances depend on the immediate goal (and to a lesser degree on the consequences of classification). For example, an object may be a 'tree' to a layman, a 'Pinus Radiata' to a botanist and 'timber' to a woodcutter. Interesting differences arising from physical and social needs, occur in the various classifications given to an object in different languages.

Classes are employed as the basic 'elements' in describing the environment. Experience is organized into a set of interrelated categories or a 'model'. This model is a representative of the relevant aspects of the environment - what is 'relevant' will depend on the factors discussed above.

Craik (45) has described a model as "any physical or chemical system which has a similar relation-structure to that of the process it imitates". He conceives it as something similar in function to the designer's model of an aeroplane, a ship or a dam. The important point is that it can be manipulated to provide a means for
anticipating the consequences of the real world.

Emphasis should also be placed on the coding aspect of the model. Craik has suggested that the only reasons for building it are on the basis of "cheapness, speed and convenience". Other people have coined expressions which incorporate this aspect - Attneave (10) has used the term, an "economical description"; Galanter and Halstenhaber (69), a "collection of regularities". It has been often pointed out that the ability to form, organize and use a model of the environment is the basis of thinking in an organism, (45, 69).

To overcome the problem of defining 'relevant' features for a model, an alternative definition proposed by Minsky, can be considered :-

"To an observer B, an object A* is a model of an object A to the extent that B can use A* to answer questions that interest him about A," (142).

This notion can be applied to the machine - the observer B could be the machine itself which requires to ask questions concerning the nature of the input which will efficiently determine its class. However, the observer could also be the user of the machine, and in the situation, the machine must be capable of providing a detailed description of the environment, which can be manipulated to the wishes of the user. The extent to which a machine is designed to cope with the above two factors determines the detail and flexibility required of its model. As Sayre (170) suggests, a recognition machine should be capable of producing a description or identification of the input. The output of this description should be in a form (such as a subset of English), which is readily understood by the user. This may involve an 'interpretation', (106), of the symbolic information stored in the machine.
The structure of a model is defined as the set of connections between the constituent classes. Our (human) environment can be represented in an hierarchical structure. For example, the class of 'trees' consists of lower level elements, such as 'trunk' and 'branches', which in turn are composed from other elements such as 'bark' and 'wood'. This type of structure is also embodied in a successive coding approach to class formation. Thus by reducing the redundancy in a message in stages, higher level elements can be formed from sets of lower level elements. On each stage, the discovered pattern can be given a 'name' of the class to which it belongs. This method of class formation, through synthesis, is incorporated in the machine.

In effect, what the machine embodies is a 'schema' for organizing its experience, in the same manner as Bartlett's suggestion for human memory, (16). Others, such as Hebb(86) in explaining the perceptual process on a neurophysiological basis, and Piaget (158) in his studies on child development, have used corresponding terms. Oldfield (155) has shown how an hierarchical coding on a binary number sequence, entails a schema. This is indeed similar to the process to be incorporated in the machine.

The model which the machine constructs is then an hierarchical set of classes. The highest level of classes corresponds to each figure or character discovered in the input. The term 'class model' shall be used to refer to the set of patterns in each figure class. In this sense, the class model is a skeleton figure constructed from various components which have been found in members of that class. The input to the machine is also described
in terms of this hierarchical coding and sometimes the word 'model' shall be applied to the model of the input. As a further point of terminology, the word 'description' is usually employed (in this report) to refer to the hierarchical set of patterns found in an input. Obviously, a description may entail only one level of patterns, (i.e. a description in terms of line segments).

2.1.5. Use of the Model

In order to handle the complexity of the environment, the organism extracts patterns from the sensed data. This information is coded into a model, which represents the important events in the organism's past experience.

The main purpose for constructing this model is that it can be employed to anticipate future events. In this way, decisions can be made concerning the future activity which the organism is to pursue.

This use of experience can also be an important heuristic governing the activity of a machine, (141). In the particular case of the recognition machine, new inputs can be described in terms of the previously derived patterns in the environment. The prediction of future events from the detection of part of a pattern, enables the determination of a difference between the input and the model. Thus, given part of a line drawing, one of a number of known classes, the class to which the entire drawing belongs, can be found by comparing the prediction from each of the classes. At the level of figures, this difference - a similarity value or a 'correction' to the schema - provides the basis for a decision.

The organism has a number of means for adapting the environment to its own needs. For example, many animals
construct homes in which to live. The programs to be described here are limited in their control of the environment, but extensions of this nature are discussed in Chapter 6.

Possession of a model reduces the necessity for constant learning of the perceived elements. An input sequence which has not previously been perceived can be categorized (and hence cognized) according to the coding scheme which has been developed. Thus once a child knows what a 'cat' is, then a variety of cats can be classified simply on the basis that each 'new' cat can be coded (described) in the same way - that is, it possesses similar features.

In the above discussion, it has been inferred that the machine continually learns. However, for practical purposes (i.e. as a device for recognizing characters) a test of the effectiveness of the model is performed in a working phase. In effect, the machine is tested for its ability to generalize upon its experience, (gained during the learning phase), to categorize new inputs. Naturally, it can classify only in relation to the known categories. However, it is possible for the machine to describe a figure by relating it to known parts or patterns of other figures. The machine's ability to learn and to generalize to new cases in the working phase are important aspects of a recognition device and will receive special treatment in this report.

A scheme for categorizing also allows the possibility of ordering and relating classes of events, rather than single instances. Thus, knowing that a radiator element will cause burns, one can expect that all red-hot metals, (of which the element is a member), will also cause burns.
As Bruner has stated, "the moment an object is placed in a category, we have opened up a whole vista of possibilities for 'going beyond' the category by virtue of its superordinate and causal relationship linking this category to others", (30). Processing of this type is limited in the machine, but the capabilities for doing so, form an important extension to the device, (see Chapter 6). The relationships are inherently present in the model; processing to exhibit the various relationships or links is necessary when certain 'questions' are asked about the model. The extent to which this processing is included in the machine depends on the problem area.

2.1.6. Feedback

The organism constantly receives notification about the stability of its existence. This feedback can be given by:-

(a) the environment; for example, when a child is told that the animal is a 'dog' and not a 'cat',
(b) an internal criterion; such as the discovery that acts (e.g. eating the toadstool) based on a certain decision had a dangerous consequence.

It is important to note that the organism must make some decision (or entertain some hypothesis) and act on this decision, in order that feedback be utilized. That is, the organism must be "concerned about error", (36).

Feedback is the major concept in Cybernetics, and has been employed in the servo-mechanism. In a model-building machine, feedback allows discovered patterns to be included in (or delineated from) a certain class. For example, the class of 'matches on fire' can be included with 'red-hot metals' into the causes of 'burning'. 
For the particular machine discussed here, the correct classification of the figures will be given by the 'environment' - by what is termed here an 'outside teacher'. This is necessary because the classification of line drawn characters is an external convention. Pattern recognition terminology notes the above difference as that of 'supervised learning' and 'unsupervised learning'. It is regarded a simple matter to operate the machine in an unsupervised mode, by incorporating a suitable criterion of success.

2.1.7. Summary of the Principles

In order to summarize the various abilities of the machine, reference will be made to the block diagram in Figure 2.1. The machine incorporates a discovery procedure for patterns in the input stream, (A), to be used when experience is not available, or in certain cases when it is not relevant for the current input. These patterns are organized, (B), to form a 'model' of the environment, (C). Decisions concerning the appearance of certain types of pattern in the input are made, (D), using the information contained in the model. The validation of this decision by an outside teacher, (O.T.) allows the model to be refined. The following discussion will attempt to relate this view to the more usual approaches to pattern recognition.

2.2. The Nature of the Approach

2.2.1. Comparison with Similar Research

The process to achieve recognition of an input is then conceived as having two stages:-

(a) the generation of an hierarchical model of the input.

This is performed by describing the input in terms of known patterns in the environment; if no experience
is available, then a given procedure generates patterns from the current input.

Operations on the model to determine whether or not the input belongs to a known set of categories may occur, of course, if the input cannot be assigned to any category. This is a refinement of the top-down categorizer notion mentioned in Chapter 1. The elements extracted in the first stage are not necessarily retained from a "battery of tests," (133). However, they are an important part of the categorizer's operation, as they provide a starting point for the categorizer's learning process. The set of elements found in the second stage are used by the categorizer to define an input in terms of other inputs. As the categorizer learns, the set of elements used in the second stage are refined and those on the right may be replaced by those on the left, that is, the elements may be discovered previously in the input sequence.

The generation procedure can also operate on a wide variety of inputs - for example, the program can (and describe) any sequence of directions on a line drawing. Obviously, there are to be greater than certain size.

The categorizer forms a comparison between given models, particularly, those of the input and certain average core of a class discovered in the learning phase. This comparison is done by determining the category of the input. The comparison in these programs involves a numerical statement in the degree of similarity between the two models. This particular environment because a certain input (e.g. 7.) may be similar to a number of known figures.

Figure 2.1. Outline of Functions in the Basic Approach.
is available, then a given procedure generates patterns from the current input.

(b) operations on the model to determine to which of a known set of categories the input belongs. It may occur, of course, that the input cannot be assigned to any category.

This is a refinement of the receptor-categorizer notion mentioned in Chapter 1. The elements extracted in the first stage are not numerical values obtained from a "battery of tests", (133). Rather, they are an hierarchical set of interrelated patterns. They form a detailed symbolic description of the input on several levels. For example, one level of the model in the first series of programs concerns curves - allowing the program to describe an input in terms of its constituent curves. As explained, these curves are related to those which have been discovered previously in the input sequence.

The generation procedure can also operate on a wide variety of inputs - for example, the programs can accept (and describe) any sequence of direction codes representing line drawings. Obviously, there are system limitations, such as the figure must be greater than a certain size.

The categorizer forms a comparison between given models - in particular, those of the input and a certain average figure of a class discovered in the learning phase. Thus relations between the various parts, as well as the parts themselves, are included in determining the category of the input. The comparison in these programs involves a numerical assignment to the degree of similarity between the models. This operation is considered necessary in this particular environment because a certain input (e.g. 7) may be similar to a number of known figures,
(e.g. '2', '7'). Hence, the description of the input can be made in terms of each class. A similarity measure is incorporated to decide which class has the greatest similarity. Further operations, perhaps using the similarity measure, are also envisaged in the second stage - for example to relate various figures which may be present in the input.

Resemblance to the 'syntactic approach' as proposed by Ledley (121, 122) can be seen. The word 'description' as employed by Ledley, corresponds to the definition of an input in terms of the primitive elements known by the machine, and the relations between these primitives. His first stage involves the obtaining of this description. The second and final stage is a "definitional reduction of the parts and their relationships to the finally desired recognition statement". Both of these stages would appear to form what has been described in the approach of this Thesis, as the first stage of the process. The reasons for incorporating the second stage of the approach (used herein) have been given above. Other differences can be noticed in the nature of the reduction rules. In the approach of the author, these rules are implicit, and are dependent on the machine's experience; in Ledley's work, they are explicit and specified by the designer.

Because the machine is continually living in the environment, each element of the stream is processed as it is received. In this approach, the process of model generation of the input is a synthesis or building-up from lower level elements. The processing required for recognition is performed 'on-line'. (The advantages and disadvantages of this synthesizing approach are discussed in Chapter 6). This is not quite true of the second series
of programs, as those programs receive and process 'chunks' of the input at certain intervals.

Pattern recognition is also viewed as the result of a discovery-induction process, mentioned by Uhr and Vossler, (209, 215). Patterns which are useful in describing the various class members, are discovered in the input. These patterns are used as a sample, to provide the basis for further action (and decisions) in the environment. This aspect is a learning by induction from experience.

Uhr and Vossler (208) have written a character recognition program based on such an approach. An input, a 20 x 20 matrix, is searched for features or operators which are 5 x 5 matrix templates, (or logical combinations of these), in which each cell has a '0', '1' or 'don't care' value. Each operator contains four characteristics (expressed numerically) relating to the 'average position' in which an operator has appeared in the (input) members of a known class. These operators are generated either from their occurrence in a particular input, randomly or by user specification. During the learning phase, various weights for the characteristics are optimized and poor operators may be purged.

When an operator match is found in a new input, the difference between its known characteristics and those for the input, is evaluated. After a series of weightings of the characteristics taken in turn for each operator, for all operators, and for all classes, a final difference value is obtained. That class with the smallest value is given the decision. The program achieved recognition rates comparable to those obtained by other methods, and was applicable to a wide variety of inputs. The results would suggest that a learning approach is not necessarily
inferior (in respect to effectiveness of recognition achievement) to one which has its features (and weights) pre-programmed, (216).

Several variants of the program have been applied to other types of patterns. One of these discussed by Prather and Uhr (160) also accepts characters, but the input values are distance-angle pairs for line segments along the character's boundary. The templates, in this case, are distance-angle pairs, which are generated as in the above program. The match of the pairs in a new input is not required to be exact; various tolerances are specified. A similarity measure, again with weights for each template, is assigned to the input from each class. Few results are given in this paper, but the generality of the operators and associated tolerances would appear to limit the program's performance.

Another program, "BOGART", discovers 'good' and 'bad' board positions in a game-playing environment, (153). These positions are used to suggest moves in the subsequent games; the good positions provide sub-goals while the poor ones are avoided. This program appears to be inefficient in its learning, requiring many trials to achieve satisfactory performance. The insertion of 'heuristics' to generalize upon the experience greatly improves the results.

The close relationship between pattern recognition and language translation is shown in a program which can, by discovering 'patterns' of letters, build the correspondences between two sequences of letters, representing sentences in different languages, (206). The corresponding patterns, formed from information gained by the feedback given by a 'teacher', are organized by various
rules into a model. While the program does learn, it relies heavily on the gradation of pattern differences in the training sequences. The model-building rules, while effective, do not appear to have the generality to form an adequate model for a practical language translation program.

The above discussed approach represented by these programs differs in a number of ways from the one presented by the author, who considers the model to be a detailed description of the whole input, rather than a set of features. (Uhr and Vossler do suggest neurophysiological evidence for their choice of 5 x 5 matrix operators).

The model mentioned herein is constructed by a general rule (and certain transformations) rather than by a 'random' selection from the input as in the Uhr and Vossler approach. Furthermore, the model is averaged over the patterns appearing in the input, whereas there is no attempt to modify purged templates in any of the character recognition programs described by other writers. Another difference occurs in the nature of the similarity measure, which is specified in the approach of the author, rather than being dependent on weighting factors. These differences are further discussed, in general, in Chapter 4.

The emphasis in this approach has been on the model constructing power of the machine. There is an attempt to show in the program how recognition and discrimination can be achieved by incorporating a process for building a detailed model. The work of Fogel (61) and Fogel et al. (62) appears to have resulted from similar objectives.

In the paper by Fogel, an account is given of a machine whose aim is to describe a time-varying signal by a set of descriptor-symbols. Each symbol represents an amplitude
range and is formed under the conditions of:-
(a) each symbol being equi-probable,
(b) the value which determines the amplitude range converging to the calculated value (according to a second-order differential equation) between the received input samples of the signal.

When the machine is waiting for the next sample, it predicts an expected environment and the symbols are modified in readiness for this change. The paper is an interesting exposition of the model building approach.

A series of programs written by Fogel et al. (62) is more concerned with the simulation of evolutionary learning. The model is a finite-state machine which predicts an environment of a sequence of numbers, containing repeated patterns in noise. At certain times in the interaction, (e.g. when the success of prediction falls below a certain value), a new machine can be generated from the current machine by such modification as increasing the number of states or changing the transition properties. By accepting only those generated machines which have better success over recent experience, a hill-climbing procedure is incorporated, thereby continually improving the overall performance.

Several modifications, such as including multiple mutations and using a variety of environments (with varying amounts of noise), were made in the experiments. The programs were more pattern discoverers than recognizers, because there was only one pattern sought. Thus class relationships did not exist. An attempt at the discrimination of a set of signals using prediction as an indicator of success gave rather poor results.

The above evolutionary programming is an inductive
procedure and is compared by Fogel et.al. to the scientific method. It is felt by the author that this is a much too grandiose an analogy for describing the programs presented herein, because of the simple model and weak generation rules incorporated.

Foulkes (63) has discussed a program, operating in an 'inductive' manner, to generate an 'n-gram' machine to predict ergodic sequences. The error in prediction is used to change the value of 'n' by either combining or splitting states. The machine continually adapts to the generating model of the environment. Different modes of reaching this goal are exhibited when the error criterion for changing the model is altered.

2.2.2. Relation of the Present Approach to Other Fields

Concept learning, as defined in psychological studies, is similar to pattern recognition as used in the present work. The corresponding process to concept learning is actually supervised learning; concept formation corresponds to unsupervised learning (mentioned in Section 2.1.6). The distinction between concept learning and pattern recognition in general, is that the former is thought of as a more conceptual act, whereas the latter is more perceptual. As Hunt points out, (97), patterns are often given the same name for different stimuli in recognition. For example, a coin is always 'perceived' circular, in various orientations in which the actual form is elliptical. The concepts for a concept learning problem are a function of the perceived stimuli. While this distinction is important, there is perhaps a matter of degree between the two types of acts, (10), and a 'coding process' could be attributed to both types. In fact, the ability of a machine to treat a perceptual and a conceptual act in
terms of codings is shown by the programs in this Thesis. The difference between the fields is more marked by the types of concept learning studies which have been performed to date. These have involved well-defined stimuli and classification rules.

Because of its discovery-induction nature, the machine can be considered as a problem solver. The approach to recognizing patterns is tackled in much the same way as a machine might attack any 'problem'. The similarity of the approach to "BOGART" has already been briefly mentioned. This approach can be contrasted to the more inductive type of programs (e.g. Samuel's checker program, (168)), and the deduction ones (e.g. Newell, Shaw and Simon's Logic Theorist, (152)). However, the correspondence to problem solving will not be stressed, because the programs are limited in their discovery procedures (see Chapter 10).

The importance for an intelligent machine to possess a model has been stressed in various reviews, (141, 156). Some programs have also stressed the need for the existence of an adequate model to provide:-

(a) generality for various problems e.g. Ernst and Newell (53).

(b) semantic interpretation to 'go beyond' the given information e.g. Lindsay (126).

(c) the ability to be creative, e.g. Amarel (8).

An adequate representation or model of the input for a machine to achieve pattern recognition is considered to be of prime importance in the present work.

2.3. The Attitude to Program Design

The approach outlined in this Chapter has presented a general basis for the design of a recognition device.
The generality of the approach has been shown by the similarity it bears to other related areas. It is now intended to discuss further the details of the machine.

The construction of a model depends on three factors:
(a) the nature of the environment, (the information must be present in the input),
(b) the goal, and
(c) the required capabilities of the device.

The goal of the machine has been given as that of recognizing figures in the input. Two series of programs have been developed from different attitudes towards the abilities which the recognition machine might have.

The first series concerns a device which attempts to describe the figures in a meaningful way. To achieve this, the programs have a wide range of abilities, not only to code and transform the input but also to extract a variety of factors in the similarity measure. This approach is an attempt to produce an efficient pattern recognition device. The programs will, therefore, be contrasted with the current approaches in character recognition.

In the second series, the abilities of the machine are restricted. In particular the patterns which are generated depend on simple relations (identity and succession) detectable in the input stream. The problem has been framed in terms of a typical concept learning experiment. However, due to the nature of the input, the programs bear little resemblance to concept learners. They do, however, more closely relate in part to many programs which have been suggested for generating regularities from data. Programs to generate regularities
from data and to learn concepts will be compared with the second series of programs developed here.

The difference between the two series can be thought of as this:— the former operate on the pictorial form of the characters, in that they can cope with the expected distortion in a picture; the latter operate on the sequences of numbers, not knowing what the elements represent.

The input to a recognition device is a representation of the real world. This representation is usually presented to a digital computer as a sequence of discrete elements or symbols. Actually the input to the programs is considered here to be a set of sequences, each sequence representing a figure or character. Because these sequences are a discrete representation of real world objects, they are guaranteed to be partly redundant — as required for the device (see Section 2.1). That this is the case is obvious from an examination of the sequences themselves. This input has the advantage of being readily acceptable and meaningful by a machine and can be easily generated. (The author was responsible for all of the generations).

The input elements represent direction codings of hand-drawn line figures. These codings are assigned the numbers 1 to 6 as shown in Figure 3.1. Six values of quantisation are used because equi-length segments are obtained. (not true for eight values), and enough information is present in the sequences for the size of drawing considered here.

An example of the coding process will now be given. The lines are drawn on triangular coordinate graph paper as shown in Figure 3.2 (a). Coding commences at some specified position — the outlined hexagonal cell is chosen.
Chapter 3

THE INPUT SEQUENCES

This Chapter describes the environment for the programs developed. Also discussed are the types of tests for recognition, and the nature of the programs.

3.1. Construction of the Sequences

The input to a recognition device is a representation of the real world. This representation is usually presented to a digital computer as a sequence of discrete elements or symbols. Actually the input to the programs is considered here to be a set of sequences; each sequence represents a figure or character. Because these sequences are a detailed representation of real world objects, they are guaranteed to be partly redundant - as required for the device, (see Section 2.1 ). That this is the case is obvious from an examination of the sequences themselves. This input has the advantage of being readily acceptable and manipulable by a machine and can be easily generated. (The author was responsible for all of the generations).

The input elements represent direction codings of hand-drawn line figures. These codings are assigned the numbers 1 to 6 as shown in Figure 3.1. Six values of quantization are used because equi-length segments are so obtained, (not true for eight values), and enough information is present in the sequences for the size of drawing considered here.

An example of the coding process will now be given. The lines are drawn on triangular coordinate graph paper, as shown in Figure 3.2 (a). Coding commences at some specified position - the outlined hexagonal cell is chosen
Figure 3.1. The Direction Codes

Figure 3.2. An Example of the Coding Operation

445434433322222121111121111343434343434

Figure 3.2. An Example of the Coding Operation
in Figure 3.2 (a). The line drawing is traced and the crossover points with lines on the grid are noted. Each of these points is approximated by the node on the graph paper to which it is closest. If this node is not the same as the previous one found, then the direction from the previous node to this new node is given a direction number, (Figure 3.1). If the node is the same, then the search for another crossover point is continued. The process proceeds until the end of the line is reached.

In some cases, the crossover point will be 'midway' between two nodes, in which case, an arbitrary allocation is made. It should be pointed out that small discrepancies formed by an incorrect choice in the case above, represent 'noise' in the sequences, with which the machine should cope. For this reason, there was no serious attempt, when the author formed the sequences, to make certain that every coding was correct.

The number sequence for the figure '2' in Figure 3.2 (a) is shown underneath the drawing. Each figure is delineated in the input, making the recognition problem much simpler. If the figures are not delineated (as in Test 16), then the problem becomes one of handwriting recognition. Following the delineation symbol, ('/'), the name of the figure is given, to allow the program to check its decision.

The drawings considered were about 1" to 2" in height on a grid with 1/10" divisions, (see Figure 3.2). The length of the sequences therefore ranged from about 20, (e.g. for '1', '7'), to 60 elements, (e.g. for '8', '9').

The characters were always coded in the same way, starting at the end, (topmost in normal orientation), from which the writing commenced. The pen was assumed never to leave the paper. The multiple stroke character '4' was
It is possible to reconstruct an approximation to the original drawing by simply tracing between the nodes which are related by the direction codes in the sequence. These nodes have been marked in Figure 3.2 (b), and this form of presentation is used in some Figures.

Freeman (65, 66, 67) has discussed this kind of coding operation and has presented means for extracting information such as area of closed figures, from such sequences. Some of the ways in which a sequence can vary for similar figures are shown in Table 3.1.

3.2. Comments on the Tests and Programs

The tests are designed to examine various recognition properties of the programs. Several types are to be found (as shown in Table 3.2):

(a) Discrimination, (DISC) - where (usually two) figures are similar, having little difference in their sequences. (Note the orientation similarity in Test 6). The differences are such, however, that a human subject can usually distinguish the classes in the original drawings.

(b) Recognition, (REC) - where many classes are present. These drawings test the generality of the constructed model.

(c) Learning, (LEARN) - many samples of the same figure are given to examine the generation of the model.

(d) In addition, there were tests designed for word recognition experiments described in Appendix A, (WORD).

Also presented in Table 3.2, are the figures, the number
<table>
<thead>
<tr>
<th>Change</th>
<th>Description</th>
<th>Given 4343333221</th>
</tr>
</thead>
<tbody>
<tr>
<td>LOCAL CHANGES</td>
<td>Stretches and squeezes in local parts of the drawing - changes the local proportion of numbers</td>
<td></td>
</tr>
<tr>
<td></td>
<td>e.g. 4344332221</td>
<td></td>
</tr>
<tr>
<td></td>
<td>e.g. 4433323221</td>
<td></td>
</tr>
<tr>
<td>ORIENTATION</td>
<td>The addition of ( +n \pmod{6} ) to each element rotates the drawing ( n \times 60^\circ ) (anti-clockwise)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>( + 60^\circ ) 5454444332</td>
<td></td>
</tr>
<tr>
<td></td>
<td>( - 60^\circ ) 3232222116</td>
<td></td>
</tr>
<tr>
<td>EXTRANEOUS SEGMENTS</td>
<td>Extra length of line at beginning and/or end of the drawings</td>
<td></td>
</tr>
<tr>
<td></td>
<td>e.g. 54444343333221111</td>
<td></td>
</tr>
<tr>
<td>SIZE</td>
<td>The repetition of each element ( n ) times increases the size of the drawing ( n ) times</td>
<td></td>
</tr>
<tr>
<td></td>
<td>e.g. 44334433333333222211 double size</td>
<td></td>
</tr>
<tr>
<td>DIFFERENT CODING</td>
<td>The coding starting from the opposite end of the line.</td>
<td></td>
</tr>
<tr>
<td>DIRECTION</td>
<td>45566666161</td>
<td></td>
</tr>
</tbody>
</table>

Table 3.1. Some Variations in the Sequences.
of sequences and the number of learning samples usually given to the program for each test.

Table 3.2 presents the tests employed in this project to determine the learning speed of the program written in the PL/I language and the number of errors it produced. The tests were designed to cover a wide range of sequences and to be as similar as possible to those used in the computer experiments of Appendix D. The tests are presented in Table 3.2 to provide an overview of the performance of the program.

<table>
<thead>
<tr>
<th>TEST</th>
<th>FIGURES</th>
<th>TEST TYPE</th>
<th>NO. OF SEQUENCES</th>
<th>USUAL NO. OF LEARNING SEQ</th>
<th>USE IN SERIES</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>U,V</td>
<td>DISC</td>
<td>12</td>
<td>12</td>
<td>A,E</td>
</tr>
<tr>
<td>2</td>
<td>5,S</td>
<td>DISC</td>
<td>12</td>
<td>12</td>
<td>A,E</td>
</tr>
<tr>
<td>3</td>
<td>2,Z</td>
<td>DISC</td>
<td>10</td>
<td>10</td>
<td>A,E</td>
</tr>
<tr>
<td>4.6</td>
<td>2,2,2</td>
<td>DISC</td>
<td>25</td>
<td>3</td>
<td>A</td>
</tr>
<tr>
<td>5</td>
<td>2,2,2</td>
<td>DISC</td>
<td>25</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>&gt;,V,&lt;</td>
<td>DISC</td>
<td>16</td>
<td>6</td>
<td>A</td>
</tr>
<tr>
<td>7</td>
<td>(,),/</td>
<td>DISC</td>
<td>20</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>6,G</td>
<td>DISC</td>
<td>25</td>
<td>6</td>
<td>A,E</td>
</tr>
<tr>
<td>9</td>
<td>C,E</td>
<td>DISC</td>
<td>25</td>
<td>6</td>
<td>A,E</td>
</tr>
<tr>
<td>10</td>
<td>6,0,U,D</td>
<td>RECOG</td>
<td>36</td>
<td>16</td>
<td>A</td>
</tr>
<tr>
<td>11</td>
<td>0-9</td>
<td>RECOG</td>
<td>81</td>
<td>40</td>
<td>A</td>
</tr>
<tr>
<td>12</td>
<td>E</td>
<td>LEARN</td>
<td>25</td>
<td>25</td>
<td>A</td>
</tr>
<tr>
<td>13</td>
<td>2,2,2</td>
<td>LEARN</td>
<td>26</td>
<td>26</td>
<td>A</td>
</tr>
<tr>
<td>14</td>
<td>6,0</td>
<td>RECOG</td>
<td>40</td>
<td>6</td>
<td>A</td>
</tr>
<tr>
<td>14.2</td>
<td>6,0</td>
<td>DISC</td>
<td>14</td>
<td>14</td>
<td>A,E</td>
</tr>
<tr>
<td>15</td>
<td>L,E,F,B,A,D</td>
<td>WORD</td>
<td>24</td>
<td>-</td>
<td>D</td>
</tr>
<tr>
<td>16</td>
<td>(WORDS FROM ABOVE LETTERS)</td>
<td></td>
<td>-</td>
<td>-</td>
<td>D</td>
</tr>
<tr>
<td>17</td>
<td>0-9</td>
<td>RECOG</td>
<td>58</td>
<td>-</td>
<td>A</td>
</tr>
<tr>
<td>18</td>
<td>0,0,∞</td>
<td>RECOG</td>
<td>24</td>
<td>12</td>
<td>A</td>
</tr>
<tr>
<td>19</td>
<td>0-9</td>
<td>RECOG</td>
<td>51</td>
<td>-</td>
<td>A</td>
</tr>
<tr>
<td>20</td>
<td>0-9</td>
<td>RECOG</td>
<td>33</td>
<td>-</td>
<td>A</td>
</tr>
</tbody>
</table>

**TABLE 3.2. THE TESTS EMPLOYED IN THIS PROJECT.**
of sequences and the number of learning samples usually given to the program for each test.

Two main series of programs (A series, E series) have been developed in this project. Each program is designated by the series letter followed by another letter, (e.g. AM), and sometimes by a further number, (e.g. AN6). The tests in Table 3.2 refer either to these series or to the programs which recognize handwriting, (D Series). A table of the main programs to be mentioned is given in Table 3.3, together with a brief description of their capabilities, and their position in this Thesis.

The programs are written in the PL/l language for an IBM 360/50 computer. Speed of operation was not a crucial factor in determining the success of the programs; however, they were written to be as concise as possible to ensure quick turnaround time from the computer.

Appendices B and C present two programs (one from each series) developed in this project in order to exhibit the kind of details employed - it is impracticable because of space limitations, to present full details of all programs developed, but these are to be published as an internal report of the Department of Engineering Physics.

Appendix D describes an on-line recognition system, using a modified A series (AN3) program. Various experiments have been and are currently being conducted with this system.
<table>
<thead>
<tr>
<th>PROGRAM NAME</th>
<th>COMMENT</th>
<th>WHERE DISCUSSED</th>
</tr>
</thead>
<tbody>
<tr>
<td>AM</td>
<td>First successful program written for recognition of sequences; 1st Series.</td>
<td>5.1</td>
</tr>
<tr>
<td>AN</td>
<td>Orientation independent program; modification of AM.</td>
<td>5.2.1.2</td>
</tr>
<tr>
<td>AN3</td>
<td>Entails 3-element line segments; modification of AN.</td>
<td>5.2.2</td>
</tr>
<tr>
<td>AN5</td>
<td>Revised confirmation values of AN; no learning.</td>
<td>5.3</td>
</tr>
<tr>
<td>AN5</td>
<td>Same as AN5 with learning.</td>
<td>5.3</td>
</tr>
<tr>
<td>DB</td>
<td>Total curvature stored instead of orientation of segments; simplification of AM.</td>
<td>5.2.6</td>
</tr>
<tr>
<td>DD</td>
<td>Program for recognizing hand-writing; modification of DB.</td>
<td>App.A</td>
</tr>
<tr>
<td>FA</td>
<td>No curves used; modification of AM.</td>
<td>5.2.4</td>
</tr>
<tr>
<td>GA</td>
<td>Orientation independent program; simplification of AM (and DB).</td>
<td>5.2.1.1</td>
</tr>
<tr>
<td>GB</td>
<td>No prediction of line segment division; modification of GA.</td>
<td>5.2.2</td>
</tr>
<tr>
<td>EC</td>
<td>First program for recognition of sequences; 2nd Series.</td>
<td>8.1</td>
</tr>
<tr>
<td>ED</td>
<td>Modification of EC for general recognition.</td>
<td>8.3</td>
</tr>
<tr>
<td>EE</td>
<td>Revised model in EC.</td>
<td>8.4</td>
</tr>
<tr>
<td>EF</td>
<td>Allows l-element patterns (to EE).</td>
<td>8.5</td>
</tr>
<tr>
<td>OM</td>
<td>Modification of AN3; for on-line experiment.</td>
<td>App.D</td>
</tr>
</tbody>
</table>

**TABLE 3.3. The Programs Mentioned in this Thesis**
The interest in the following series of programs is to produce a machine which can recognize the sequences in a meaningful way. That is, the programs are required to be capable of extracting from the input those parts and relations which might be extracted by a human observer. It is intended that the user of the programs should be capable of readily interpreting the description provided by the system. The machine naturally contains the description in symbolic form; extension of the programs to output in a simplified language is briefly discussed in Chapter 6.

The model, which represents a particular input or the experience of the machine, is an hierarchical set of interrelated classes. There are four levels, namely,

Level 1: FIGURE
Level 2: CURVES
Level 3: LINE SEGMENTS
Level 4: DIRECTION ELEMENTS

The direction elements form the input stream, and hence correspond to the smallest discernible part of a figure. Each higher level consists of a set of lower level elements. Relations exist between elements which form higher level classes; line segments are sets of similar direction elements. Curves are defined in terms of a set of line segments with similar slope difference - there are basically five types, but there can be many different sets within these types. Finally, a set of curve types constitutes a figure. The relation of order exists between the parts on any
level. Various parameters are used to store the properties of a known model. As an example, a figure '2' may consist of three curve types - '>', 'v' and '—', ordered in this way.

It should be noted that the above hierarchy presents a class model - a set of these for the discovered classes (to be recognized) corresponds to the model of the environment.

The model provides a detailed description of the input - at least when compared with that provided by a set of 'n' unrelated features. However, the processing time for its generation is naturally considerable. Most attempts at character recognition try to define a set of easily extractable but potentially powerful features, in order to keep the time to process the inputs to a minimum. While processing time is not a crucial factor to this project, it is important if the device is to be considered for practical use, and there has, in fact, been an attempt to reduce the amount of processing.

It is also expected that with a detailed model, the possibility of improving the recognition rate will exist. Uhr's suggests in connection with this aspect that the "retention of all, or even too much, geometric detail will lead to over-many false differentiations", (210). The point that is apparently being made is that the retention of too much detail from the learning samples may lead to a 'biased' model, (4). There must be a compromise in design between the amount of detail and the number and type of learning samples. However, given a model, then surely the more information obtained from the input the better will be the decision.
The present suggestion for a model can be thought of as lying in a certain continuum representing the structural properties of the receptor's features. At one end is the template which represents a fixed structure for a class. The abilities to generalize over class members depends on suitable pre-normalization techniques which are difficult to specify and implement for a wide variety of members, as found in handprinting. At the other end, is the random net, which initially has no structure imposed upon it. Through interaction with the given samples, the device generates combinations of properties (usually hierarchically) to describe the defined classes. Again, this model possesses a lack of efficiency to generalize to new members of the class. Furthermore, in both of the above cases, the size of the device becomes prohibitive when the number of classes (and variations within the class) increases. For templates, more normalization techniques and templates are required; for the random net, more levels and associations between the levels are necessary. The result leads to a complex system requiring in the case of random nets, extremely long training sequences. Neither of these schemes has produced results useful for recognizing handprinting.

Characteristic features, such as analogue parts or sets of matrix cells, seem to fit between these extremes - results from programs incorporating these features show considerable improvement over the methods in the previous paragraph. The choice of which kind of receptor is used, usually depends on the 'noise' in the input; for instance, curve following techniques for analogue parts fail when there are breaks in lines. Another factor is that matrix cell or 'n-tuple' operations are easier to implement, and
are more suited to statistical techniques often employed in the categorizer. The advantage with both types of feature is that a wide variety of inputs can be specified in terms of the presence or absence of a set of properties.

The model suggested in this Section (4.1) appears to bear resemblance to the various types of receptors. The line segments and curves are, in effect, analogue parts of the input. The set of line segments which is sought in an input can be thought of as simple templates. A relation with the results of certain neurophysiological research, especially by Hubel and Wiesel (95) working with cats, may be noticed. However, the line segments which organize the input in the author's programs are derived from the experience of the machine; this appears to contradict any suggestion of a basic innate mechanism for their development. In the respect of learning and in the hierarchical nature of the model, the proposal is similar to a random net. However, the 'net' is constrained because only a certain type of pattern can be extracted at each level, and only a given form for the patterns can be extracted on each level. Thus, on the third level, a set of curves, each being one of five types, must be extracted.

Receptors may be considered in another 'continuum', which was alluded to earlier - that of detail in the description. At one end, (rather ill-defined) are those receptors which extract a simple set of features, (23,212). At the other end (the model suggested in this Section is a case in point) are those which attempt to describe the entire input, noting the various parts and relations. Usually these schemes involve an hierarchical model. There have been few attempts at providing this 'Gestalt' or
'whole' specification; some of them will be discussed below.

One of the earliest programs, but nevertheless one of the most sophisticated, is that written by Grimsdale et al., (79). The initial step involves preprocessing the input matrix, to overcome noise and to standardize the line width. The second stage analyses the figure into a set of groups which appear to the author to be 'blocks' of straight lines. A synthesis section forms simpler groups from these blocks. Thus a cross 'X', would be described in terms of two lines rather than four blocks. Various properties, such as the types of joins between the groups, are appended to a total description containing 40 digits.

Mermelstein and Eden (136) analyse handwriting into upward and downward strokes by noting the position of zero velocity in the writing. These strokes are then placed in predefined categories, in a not necessarily unique manner. The categories depend on the nature of the writing, and various types have been presented in connection with a model for generating handwriting, (51,52). As will be mentioned later in this Chapter, the recognition of a word is performed by a synthesis of the strokes into allowable letters and then words. The stroke types appear to be more complex than the line segments, but less complex than the curve types used in this Thesis.

Ledley (121) has defined various curve types to be found in the contour of a chromosome. While these types bear some resemblance to those suggested here, there are more variations within each type in the problem of this project (handprinting ) than in the contours of chromosomes. Essentially, however, there is little difference between the models.
Bernstein (21) has suggested the formation of curves for handprinting, on the basis of the amount and direction of rotation. The curve types are not quite the same as suggested here, as Bernstein proposed the additional features of cupped elements and loops. Sayre (170) has also suggested some curve types but he incorporates a straight line and a sharp corner into one type - these are separated in the present project. It is difficult to reconcile Sayre's suggestion with his presented discussion in which he mentions the work of Attneave. The author came to the different view by noting that lines are more 'primitive' (more redundant) than sharp corners. Ideally, sharp corners should be a relation between certain curve types (e.g. straight lines). However, there is a problem in forming the description by this means, because in sloppy handprinting, a sharp corner often merges into a well-formed curve (e.g. '7' instead of '7'). For this reason, the sharp corner has been made a special curve type herein.

Some workers have simply defined the character in terms of straight line segments. Spinrad (193) has a particularly good noise rejection method for inserting the segments (in 8 orientations) into a given matrix-form input. His model consists of the length and orientation of the segments and the vector between the centre of gravity of each segment and the centre of gravity of the figure. Marill et.al. (134) synthesize line segments from a set of (X,Y) coordinates lying on the input drawing. Properties of these segments are used to classify the input.

Topological properties have been often used to define various 'curves' in input characters, and to limit the search of possible input membership, (Kuhl (117), Minneman (140), Sherman (179), Spilerman (192)). Geometric information such
as the slope and length of each curve is then extracted to classify the input uniquely. Knoke and Wiley (109), who, like Kuhl, operate on a Freeman code for the lines, also appear to describe the input in terms of topological properties. Their approach is syntactic, and two phrase structure languages are provided—one to describe the shapes between the graph's vertices and the other to reduce this description to known statements. Unfortunately, the conciseness of their paper makes a comparison difficult, but one would expect similar constraints on class formation to exist in their programs to those reported here.

The above discussion has mentioned some of the approaches to 'analogue' description of character. In the situation of reading the character, the lines are usually thick, and slope information is difficult to extract. The curve types have been identified more easily by intersections and ends of lines. Some programs have concentrated on the use of straight line segments. Slope information is readily obtained when the input is presented 'on-line'. However, most programs in this situation have concentrated on the fast processing of the characters (49, 80, 200) and have tended to define a limited set of features. The choice of curve types depends on various factors such as the nature of the input and the aim of the device. In this Thesis, the model has been governed by the notion of regularities or patterns. The following Section will explain how a model of the input is generated and how a model of the environment is constructed by the programs.

4.2. Processing the Input

The programs must incorporate some means for generating the figural model discussed in the previous Section from an input sequence. How the patterns in the sequence can be
coded to produce the hierarchical description is now discussed.

The scheme groups elements, as they are received in the input, into line segments, the general forms of which are prespecified. These forms are related to, but not generated from, the similarity between the constituent elements. The set '3333' is an example of a line segment. It is considered that the learning of a general form is possible by the machine, but a procedure is not incorporated because of the desire for the device to be an efficient learner.

The segmentation is not unique for any given sequence of numbers, and a general rule, RO, determines the 'best' (segmentation) when no experience is available:

RO: 'The next line segment is the same as that found previously'. (The first segment assumes the previous one is '1111' (equivalent to 0°).

Each line segment which is formed can be coded into its particular orientation. In some programs, the length of the segment, which is variable, is also recorded. The segment '1111' is taken as the origin at 0°; hence '3333' is equivalent to 120°.

Successive differences in slope values form another sequence of elements which can be grouped into curves. Again, the form of the curves is known by each program and is dependent on the similarity between the successive slope difference values. Thus a string of values

\[40 \, 40 \, 40 \, -40 \, -40 \, -40\]

would give rise to two curves each traced in opposite directions, and each having three elements.

When one slope difference is obtained, RO can be replaced by a higher level rule, R1:
Rl: 'The next line segment is that which gives the closest value of slope difference to that previously obtained'.

This rule projects the expectation of the present curve to continue. In turn, curves can be coded into a set of certain types to represent figures. Given the experience of one figure, another input can be searched for similar parts that were discovered in that figure. Thus, at the segment level, another more general rule, R2, can replace Rl:–

R2: 'The next line segment is that which is closest in slope to that expected in the stored figure'.

If there is a number of stored figures, then a number of R2 rules will be ordering the input. On the curve level, the form of the rule is:–

R2 (curve):– 'The next curve is that which has been found in the discovered class'.

These rules can generate an hierarchical structure of patterns from the input sequence. The programs have the ability to:–

(a) form parts (line segments, curves, figures) from an appropriate sequence of elements, and

(b) code these parts into another sequence, in which the higher level elements are found.

The process groups and transforms parts of the input.

The rules R0, Rl and R2, in effect, limit the generation of possible description, by recourse to the experience of the machine. That is, they describe the input in terms of regularities found in previous sequences. This aspect is discussed further in Chapter 7, in connection with similar rules for the second series of programs.

The generation of the description is partly a synthesis–
the elements of the sequence are successively combined into higher level categories. This process can be described by analogy with a 'bug' (c.f. Ledley's bug (123) which defines the edges of objects) which travels along the line drawing. Its aim is to remember its travelling experience as economically as possible. It is suggested here that if the bug can sense direction and changes in slope, a description of curve types (as similar slope difference line segments) is a natural coding for the bug to use.

The synthesis of Marill et.al. (134) has been mentioned in the previous Section. Groups of \((X,Y)\) coordinate positions are combined in a manner independent of the sequence of these positions. The program incorporates a rule, basically a continuation of regularity in direction, to determine the groups. The advantage of the grouping, as pointed out, is that the device has the ability to sort out figures from 'structured noise' (such as adjacent or overlapping figures). This advantage, present in the author's scheme, is not present in a feature extracting kind of program.

By contrast to the synthesis, a common procedure in character recognition programs, after noise cleaning routines, is one of delineating the various parts in the figure. For example, Sherman (179) delineates the line figures by 'quasi-topological' properties, and Mermelstein and Eden (136) segment the word into stroke categories. The input has been so designed here that there is no need for such a process.

There are two further uses for the rules. Firstly, a measure can be given to the degree of similarity between the generated parts in the input and those expected. A
basis for comparison might be, for example, the match of the orientation of line segments. This match can give rise to a decision for further action, such as the requirement of another input segment for better agreement with the expected value. On the figure level, the similarity which is a function of the similarities on the lower levels, is used as a basis for the decision of input class membership. Secondly, feedback from the outside teacher validating the decision allows the generated patterns to modify or average the corresponding stored values for members of the same class. In this way, a refined model of the environment is obtained. These aspects are discussed in the following Sections.

4.3. The Decision

The experience of the machine is stored in a model, which contains various patterns on the different levels of the hierarchy. The generated model will therefore contain a set of average members (AVS) which consists of the various patterns (e.g. curves, line segments) found and expected to be found in members of a particular class.

It is possible that more than one AV* for a given class can be generated. These situations arise when differently shaped (hence differently patterned) figures with the same name are found, (e.g. \( \mathfrak{a}, \mathfrak{b} \)). The generation of these AVS is discussed in the next Section.

The main aim of the program is to make a decision concerning the class membership of the input. This may be thought of in terms of answering the question: - 'To which 'average' set of patterns are those in the input most similar?' A similarity measure is specified in the

* AV is the singular of AVS.
programs to provide the answer. It will be noticed that the decision function depends on an average or typical member of the class and a measure of similarity. This approach can be contrasted with the discriminant techniques which incorporate the specific delineation of class boundaries, - the philosophy of the linear decision function, for example. The method considered here is an approximation to the decision theory approach, (175), and has followed naturally from the outlined theory, (Chapter 2).

The process leading to the decision-making bears some resemblance to the inductive procedure of hypothesis generation and testing. It is of interest to discuss the process in these terms. The hypothesis can be thought of as an implicit statement, relating to the uncertainty of class membership of the input, as, for example, 'The input is most similar to the AV, type X', where X is a class name already discovered in the environment.

The generation of the hypotheses in the programs simply involves the postulation, at the beginning of the input sequence, of the set of above statements for each AV. These hypotheses are then tested by comparing the consequences of the statement (the patterns in the AV) to the input (the generated patterns). Thus, the determination of similarity is essentially a comparison of models.

As previously mentioned, the input description is not necessarily unique, and different hypotheses will partition the input stream in different ways.

The degree of agreement between the models (a particular AV and the input) is found by a similarity or confirmation measure, which is given a numerical value. This measure is determined empirically and depends on the agreement at each level. Thus the confirmation value
given to an hypothesis depends on the number of curves, and the similar shape between the expected and generated curves, which in turn depends on the agreement between the line segments. The measure is employed to determine whether one hypothesis is more confirmed than another. That is, it is used in a comparative manner and should not be construed as giving a quantitative value to the similarity.

Because each class is considered equi-probable _a priori_, the measure is also equivalent to one for the credibility or relative preference of each hypothesis, (217). The nature of such measures is a contentious issue, especially in philosophy. In some cases, credibility can be given a numerical value calculated from the relative frequency of events (e.g. coin tossing) or it may follow 'logically' from the supporting evidence, (32, 87). In other cases, it is doubtful whether numerical values can be assigned at all, (161). The confirmation measure as employed here can be considered as being subjective (to the machine), thereby partly avoiding this controversy. In these terms, the problem for the designer is to provide a 'rational' measure for the confirmation, or one that would be given by the 'average' human observer. Studies to determine such values given by human subjects for various inputs, by methods outlined by Carnap (32), would form an interesting experiment.

The value for confirmation is continually updated as more of the input is received. It is expected, of course, that as more information is gained, the values will tend to become concentrated on the more probable hypothesis. Finally, at the end of the sequence, a decision is given to that class whose hypothesis is most confirmed.
In the sense of having one hypothesis eventually becoming the 'law', this process differs from that in natural science, where no objective hypothesis exists. In such a situation, the evidence is 'infinite' and hypotheses must be continually modified in the light of new evidence; sometimes, new hypotheses must be generated. This aspect is not incorporated in the programs, although the possibility for generating more hypotheses, is of interest, (see Chapter 6).

It is possible for the generated patterns to be quite dissimilar from those which are expected. In the programs, this might correspond to a curve of positive slope difference being found when one of negative slope difference was expected. In such cases, the hypothesis can be purged - a process of 'logical refutation'.

Some programs have been specifically designed to study the generation and testing of hypotheses. Watanabe (217) has developed a mathematical model based on Bayes' theorem, to provide measures for credibility and confirmation. Results are presented for the 'urn problem' - deciding the proportion of white and black balls in an urn by sampling - and Kochen's (110) concept learning problem. The essential features for the inductive scheme, as outlined above, are exhibited by the model; in particular, the continual concentration of the credibility on the 'correct hypothesis'.

Kochen's (110, 111, 112) programs incorporate an added complexity in being able to modify current hypotheses which are logically refuted. Several programs allowing various logical combinations of features in the conceptual rule have been developed. His weight for an hypothesis, based on the cardinality of the set defined by the hypothesis,
appears to correspond to a credibility measure.

Jeans (100) has discussed hypothesis testing in the problem of coin tossing and the determination of randomness (by testing recurrence of patterns) in a binary sequence. In these cases a function derived from the probability of 'success' for the alternatives, is presented for the credibility.

Marill et al. (134) have viewed their device as generating and testing hypotheses. While the generation is similar to that suggested for the programs in this series, Marill et al. proceed to confirm their hypothesis by a set of pre-determined tests, (such as 'a straightness coefficient for the input must be less than a certain value for the input to be a figure 'f' '). A similar proposal is employed by Munson (49). These differ from the author's confirmation measures basically in their nature and form – herein, one final value is obtained from the set of values formed for different factors (shape, orientation). The similarity measure of Grimsdale et al. (79) appears to extract essentially the same information from the generated description as proposed here, although this is difficult to verify from Grimsdale's limited explanation. Given the same number of segments in the input as in the stored figure, a score is calculated from the agreement to a suggested orientation between the number of ends, joins, crosses and length of each segment. Different segments are matched to determine the best orientation, which gives the maximum score. The orientation is taken into account, if necessary, to distinguish, for example, 'M' and 'W'.

Other similarity or distance measures (e.g. Uhr and Vossler (208), Sebestyen (175)) concentrate on the various
features or characteristic parts in each figure, rather than extracting information about factors (e.g. size, orientation) in the whole figure. It is conspicuous that many of the programs with the more sophisticated descriptions of the input (e.g. Kuhl (117), Sherman (179)) rely on a 'decision tree' approach in the categorizer. An advantage of the measure adopted in the present work is the meaningfulness of the information extracted, which can be understood by a user of the machine.

4.4. Learning

As explained in the previous Section, each class is typified by an AV, which contains parts commonly found in members of that class. These AVs are generated from the experience of the machine in the learning phase, when the decisions are validated by an outside teacher.

The machine cannot give any decision to the very first input sequence, because it is unaware that any figure categories exist. The second input sequence is always given (by the machine) the name of the first and only category it knows. When more than one class is known, the decision will depend on the similarity measure for each class.

If the decision is correct, then the AV for the correct class can be averaged, or in a sense, generalized (from the viewpoint of the extension of the class). The patterns are actually averaged in the programs by incorporating the input line segments into the total set of line segments which form the AV. If the decision is incorrect, then one of two cases may arise:

(a) the class is not known. The input is then stored as the typical member. (The first sequence is a
special case of this situation). In this way, the names of all the classes are found.

(b) The class of the input is known - but the calculated similarity for the correct class is not the maximum. Some modification procedure such as forming a new subset, is taken.

The purpose of the learning phase is therefore to construct suitable AVS for each class. Noise and irregularities present in particular members are eliminated because they are not typical of the members in the class.

The learning phase may be terminated when the AVS are well-averaged and stabilized. To assess when a particular AV is in such a state is a subjective decision for the user. One, of course, assumes that the samples are typical of the classes, and of those to be given in the working phase, but this again is difficult to define for a given set. The usual practice is to choose a set of well-formed samples and keep presenting them to the machine, until the recognition of these learning samples appears satisfactory. If sometimes occurs, however, without the user's knowledge of the AVS at the end of the learning phase, that some AVS may not be well-averaged. Often the phrase, 'poor AVS', will be suggested as the reason for an incorrect decision relating to this problem.

The success of the learning phase depends also on the gradation of similarity in the order of presentation of the samples. It is possible for a certain ordering to produce a number of subsets, whereas a different arrangement of the same samples may not cause any subdivision. The 'best' order of presentation is not at all obvious to the user.

It has been assumed that the learning scheme does,
of course, produce averaged members. There exists the possibility that the scheme may form a bias in the averaging, with a tendency for an AV to retain certain irregularities which may appear in the samples. Again, the problem of deciding whether the scheme is adequate, is a subjective decision. For the above reasons, there have been special investigations into the learning properties of the programs, (Chapter 5).

The basic rule for learning presented herein, has been termed by Sebestyen (175) (in connection with the 'n-dimensional' hyperspace concept), a "proximity algorithm". The decision discussed by Sebestyen for class membership is based on a distance measure between the input and all the sampled members of the class. That class whose member has the minimum distance is given the decision. The rule used herein does not consider each member individually, but is modified to measure the distance to only the AV. (Actually the inverse of this distance is calculated). Sebestyen has modified the "proximity algorithm" to one which computes the distance to a limited number of members - these members are found in the learning phase and have the property that no other sample is closer than a specified distance to them. In this way, the original distribution is approximated and the number of members stored is limited. The problem with incorporating such a procedure into the author's scheme is that the storage would increase considerably with each new AV - the results given by the method actually used, however, do appear satisfactory, although application of the method might not be possible to other areas such as speech recognition, because of the nature of the distribution of members in a class in the latter.
It is noticeable that few of the programs cited in the literature possessing detailed models, can learn from their experience. Grimsdale's program, perhaps the most sophisticated, allowed description of unknown classes to be stored, but 'similar' members of the same class were not averaged.

4.5. Summary

The block diagram of Figure 4.1 shows the basic processing routines outlined in the present theory. There are four levels in the Model hierarchy (Figure, Curves, Line Segments (Line Segs.), Direction Elements, (Dirn. Els.)), from each of which predictions (P) are made. These expectations are compared (CF) to the corresponding input elements. Confirmation values (CN) are determined from the degree of the match. The input elements are then coded to form higher level parts (F), and the coded parts are passed as input to the next level.

At the end of the sequence, a decision (DEC) is made from the confirmation measure, which is a function of the measures determined at each level. This decision is validated by an outside teacher, (O.T.). If the machine is in the learning phase (l.p.), then the model is refined (AVE), and the current class model determined, (CVE). Otherwise in the working phase, the machine awaits another input sequence.

The following Chapter 5 presents the operation and results of programs, which incorporate the basic functions outlined in the above-mentioned block diagram.
The diagram describes a series of programs which were developed to overcome certain problems or to determine detailed results (originally reported in Chapter 2, and further discussed in Section 7.3). This final program is the best that has been developed, as it does produce better results than those on other processors.

The basic functions shown in Figure 4.1 have been shaded. The amendments to this diagram have been made, in order to produce an efficient recognizer for the sequence of functions.

(a) no attempt is made to predict direction elements and subsequently input those predicted. Any comparison would therefore be a confirmation of relatively insensitive measurements.

Further discussed in Section 7.3.

Figure 4.1. Outline of Functions in Programs of First Series.
This Chapter describes a series of programs which is based on the theory outlined in Chapter 2, and developed in Chapter 4. A description is given of the first program written (designated AM), of modifications which were made either to overcome certain problems or to determine an effect, and of detailed results from an improved version of AM (designated AN6). This final program is not considered to be the best possible - it does produce better results than from the other programs developed, but would require further modification for specific applications. This aspect is discussed in Chapter 6.

The basic functions for all programs have been shown in Figure 4.1. Two amendments to this diagram have been made, in order to produce an efficient recognizer for the given sequences: -

(a) no attempt is made to predict direction elements and subsequently to compare input elements with those predicted. A measure obtained from the comparison would provide a confirmation to orientation of a figure, but another (more efficient) measure for the factor is used. This amendment arises in the second series of programs and is further discussed in Section 7.3.

(b) on the segment level, the confirmation values are not updated after each input segment.

The other functions shown in Figure 4.1 are represented in the programs.
5.1. Introduction to the First Program, AM

The operation of the program AM, presented by giving an example of processing an input sequence, is now described:

5.1.1. Storage

The various values defining an average member (AV) in the model, are kept in two PL/1 structures - the first contains the curve numbers for each figure, and the second contains the particulars of each curve. An example of a figure '5' and 'S' are shown in Table 5.1. The curve type may be one of five values; Table 5.2 shows these possibilities, which are determined by the CVE routine, (see Section 5.2.6).

5.1.2. Prediction

Each AV successively predicts the expected curve types which in turn predict the orientation of the expected segments. The rules RO, R1 and R2 (Section 4.3) are used to form a description (similar to a given AV), of the input.

Naturally few input segments have the same value as the predicted orientation (PD), but some will have a similar value. Associated with each PD, there is a range (PAS-FUT) within which an input orientation is said to have an 'exact match' to PD. The ranges for the predictions from the figure '5' (of Table 5.1) are presented in Table 5.3. The PAS and FUT values lie midway between successive predictions (e.g. Nos.7, 8) but must have a minimum tolerance to PD of 20°,(Nos. 1, 2, 3). In cases where there is no previous or future prediction, (Nos.1,11), or where two curves of opposite slope difference sign are consecutive, (Nos.6,7), one tolerance will not be defined by the above rules and it is made equal (but opposite in
TABLE 5.1. CURRENT CLASS AVS STORED BY AM (TEST 2).
The curve types are independent of orientation but dependent on the direction of drawing.

Table 5.2. Curve types formed by the CV routine.

<table>
<thead>
<tr>
<th>Curve No</th>
<th>Pictorial Repn</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td><img src="image" alt="Sharp Corner; (+)VE Slope Difference" /></td>
<td>Sharp Corner; (+)VE Slope Difference</td>
</tr>
<tr>
<td>2</td>
<td><img src="image" alt="Sharp Corner; (-)VE Slope Difference" /></td>
<td>Sharp Corner; (-)VE Slope Difference</td>
</tr>
<tr>
<td>3</td>
<td><img src="image" alt="Curve; (+)VE Slope Difference" /></td>
<td>Curve; (+)VE Slope Difference</td>
</tr>
<tr>
<td>4</td>
<td><img src="image" alt="Curve; (-)VE Slope Difference" /></td>
<td>Curve; (-)VE Slope Difference</td>
</tr>
<tr>
<td>5</td>
<td><img src="image" alt="Straight Line" /></td>
<td>Straight Line</td>
</tr>
</tbody>
</table>

In certain cases, a segment cannot be found from an input segment list. In such cases, a segment is included in the sequence if it has not been previously included and if it is closest to the predictions. In other cases, the number of segments in the sequence of each element is greater than the number of additions, while that for '5' includes '282'.
sign) to the other one.

### 5.1.3. Formation of Parts

As elements are received by the program, they are grouped into line segments. Allowable forms for the segments have been specified as:

\[
\begin{align*}
xy, & \quad yx, \\
xxy, & \quad yxx, \\
xxxxy, & \quad yxxx, \quad xxxx, \\
xxxxxy, & \quad yxxxx, \quad yxxxx,
\end{align*}
\]

where \( y = x \pm 1 \pmod{6} \) and \( x = 1, 2, \ldots, 6 \).

For any given sequence, the segmentation is usually not unique. The segmentation obtained by the program is that most 'similar' to the predicting AV, and examples for the '5' and 'S' (of Table 5.1) for an input are shown in Table 5.4. The total number of ways of segmenting this particular input is greater than 60.

In certain cases, a segment cannot be found from an input either because:

(a) the input element has \( y = x \pm 2, 3 \) or 4, or
(b) some elements are left over at the end of a sequence, (e.g. '___22').

In either case, additional elements are concatenated into the sequence, so that the resulting segment is the closest to the predictions. It is for this reason that the number of segmentations in the sequence of Table 5.4 can be greater than 60. In the segmentations obtained, an element '1' is added into the 4th line segment (___ 62 ____ connection) for both AVS; the 6th line segment of 'S' does not require additions, while that for '5' includes '222'.

The production of curves depends on the AV. Table 5.4 also shows in the CV column, the different curves which have been formed as 'similar' to those in the
The end of the figure is known by the program, so that a response is expected.

**INPUT, TEST 2, NO. 7:**

616161621124333332322211616166

**PREDICTION FROM AV '5':**

<table>
<thead>
<tr>
<th>NO</th>
<th>SEGMENT</th>
<th>SLOPE</th>
<th>DIF</th>
<th>CV</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>61</td>
<td>-30</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>61</td>
<td>-30</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>61</td>
<td>-30</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>61</td>
<td>-30</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>211</td>
<td>19</td>
<td>59</td>
<td>2</td>
</tr>
<tr>
<td>6</td>
<td>2222</td>
<td>60</td>
<td>41</td>
<td>2</td>
</tr>
<tr>
<td>7</td>
<td>433</td>
<td>139</td>
<td>79</td>
<td>2</td>
</tr>
<tr>
<td>8</td>
<td>3332</td>
<td>106</td>
<td>-23</td>
<td>3</td>
</tr>
<tr>
<td>9</td>
<td>3222</td>
<td>74</td>
<td>-32</td>
<td>3</td>
</tr>
<tr>
<td>10</td>
<td>116</td>
<td>-19</td>
<td>-93</td>
<td>3</td>
</tr>
<tr>
<td>11</td>
<td>16</td>
<td>-30</td>
<td>-11</td>
<td>3</td>
</tr>
<tr>
<td>12</td>
<td>166</td>
<td>-41</td>
<td>-11</td>
<td>3</td>
</tr>
</tbody>
</table>

**PREDICTION FROM AV 'S':**

<table>
<thead>
<tr>
<th>NO</th>
<th>SEGMENT</th>
<th>SLOPE</th>
<th>DIF</th>
<th>CV</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>61</td>
<td>-30</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>61</td>
<td>-30</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>61</td>
<td>-30</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>61</td>
<td>-30</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>211</td>
<td>131</td>
<td>101</td>
<td>2</td>
</tr>
<tr>
<td>6</td>
<td>2222</td>
<td>90</td>
<td>-41</td>
<td>2</td>
</tr>
<tr>
<td>7</td>
<td>4333</td>
<td>90</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>8</td>
<td>3332</td>
<td>90</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>9</td>
<td>3222</td>
<td>90</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>10</td>
<td>116</td>
<td>41</td>
<td>-49</td>
<td>3</td>
</tr>
<tr>
<td>11</td>
<td>16</td>
<td>-30</td>
<td>-71</td>
<td>3</td>
</tr>
<tr>
<td>12</td>
<td>16</td>
<td>-30</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>13</td>
<td>166</td>
<td>-41</td>
<td>-11</td>
<td>3</td>
</tr>
</tbody>
</table>

**TABLE 5.4. SEGMENTATION OF INPUT, BY AM (TEST 2).**
generating (stored) AV.

The end of the figure is known by the program, so that only one figure per sequence is expected.

5.1.4. Comparison

Figure 4.1 shows the formation phase following the comparison (CF) and confirmation (CN) phases on each level. On the 'element stage' in the program, the CF and CN phases are omitted so that the formation into line segments is the first operation on the input elements. In the actual program, the CF and formation phases are joined, (Appendix B).

The orientation of each line segment generated, (AP), is compared to the PAS-FUT range of the predicted line segment from a given AV. There are four possible outcomes of this line segment comparison:

P(a) AP lies within the range,
P(b) AP is on the PAS side of the range,
P(c) AP is on the FUT side of the range,
P(d) the sign of the slope difference between successive AP values is opposite to that expected.

In each case, there is a specified response by the program.

On the segment level, the response could be one of the following:

L2 (a) the input is confirmed, (e.g. from P(a)).
L2 (b) the input is 'stretched' - i.e. pictorially, there is an additional line segment, (e.g. from P(b)).
L2 (c) the input is 'squeezed' - i.e. pictorially, a segment is missing, (e.g. from P (c)).

On the curve level, the response may specify that either:

L3 (a) the input segment is part of the last curve,
L3 (b) the input segment corresponds to the succeeding
curve type in the AV.

In each of these five cases, subsequent action is taken to obtain either another input segment (as in L2 (b), L3 (a)), another prediction (as in L2 (c), L3 (b)), or both (as in L2 (a)).

In certain situations, following the outcome of P (d), the AV hypothesis may be purged (on the figure level). The specification of when purging can occur is a problem for the designer.

In Table 5.5, the various outcomes are shown in the variables 'C' and 'U'. Examples of outcomes, L2 (b), (C=3), and L3(a), (C=9), can be seen. Notice that for figure '5', in No. 10 there is a 'stretch' followed by a 'squeeze'. Examples of a pure 'squeeze', (U >1), and extra input segments, (C=9), do not occur. If extra input segments occurred, these segments would be considered as 'stretches', (provided P (d) did not result).

5.1.5. Confirmation

The confirmation measure (CF (X)) for each curve (X), is calculated by:

\[ CF (X) = \frac{T \times T}{K \times J} \]

where T is the number of matches (C=1), K is the number of segments, and J is the number of predictions.

The confirmation (CP(W)) for each hypothesis (W) is the sum of that for each curve weighted over the number of predicted values:

\[ CP (W) = \sum_{X} \frac{CF (X)}{XM} \]

That class with the maximum confirmation at the end of the input, receives the decision. Table 5.6 gives the
GENERATION FROM AV '5':-

<table>
<thead>
<tr>
<th>NO</th>
<th>SEGMENT</th>
<th>C</th>
<th>U</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>61</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>61</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>61</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>61</td>
<td>9</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>211</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>2222</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>7</td>
<td>433</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>8</td>
<td>3332</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>9</td>
<td>3222</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>10</td>
<td>116</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>11</td>
<td>16</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>12</td>
<td>166</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

GENERATION FROM AV 'S':-

<table>
<thead>
<tr>
<th>NO</th>
<th>SEGMENT</th>
<th>C</th>
<th>U</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>61</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>61</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>61</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>61</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>21</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>12</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>7</td>
<td>4333</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>8</td>
<td>32</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>9</td>
<td>32</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>10</td>
<td>221</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>11</td>
<td>16</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>12</td>
<td>16</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>13</td>
<td>166</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

C: VARIABLE INDICATING A 'STRETCH'
(C=3: SEGMENT STRETCH; C=9: CURVE STRETCH).

U: VARIABLE GIVING THE NO. OF PREDICTIONS
(U>1: SQUEEZE).

TABLE 5.5. THE VARIABLES 'C' AND 'U' IN AM (TEST 2).

<table>
<thead>
<tr>
<th>FIGURE</th>
<th>X</th>
<th>CF(X)</th>
<th>NO. OF SEG. (PER CURVE)</th>
</tr>
</thead>
<tbody>
<tr>
<td>'5'</td>
<td>1</td>
<td>1.00</td>
<td>3</td>
</tr>
<tr>
<td>'5'</td>
<td>2</td>
<td>0.75</td>
<td>4</td>
</tr>
<tr>
<td>'5'</td>
<td>2</td>
<td>0.38</td>
<td>6</td>
</tr>
<tr>
<td>'5'</td>
<td>2</td>
<td>0.57</td>
<td>7</td>
</tr>
<tr>
<td>'5'</td>
<td>2</td>
<td>1.00</td>
<td>3</td>
</tr>
<tr>
<td>'5'</td>
<td>3</td>
<td>0.00</td>
<td>2</td>
</tr>
<tr>
<td>'5'</td>
<td>3</td>
<td>0.50</td>
<td>2</td>
</tr>
<tr>
<td>'5'</td>
<td>3</td>
<td>1.00</td>
<td>2</td>
</tr>
<tr>
<td>'5'</td>
<td>3</td>
<td>0.64</td>
<td>5</td>
</tr>
</tbody>
</table>

(TABLE GIVES VALUES AS FOUND SUCCESSIVELY).

\[
CP('5') = 0.768 \\
CP('5') = 0.797
\]

FIGURE '5' RECEIVES THE DECISION, (TEST 2, NO.7).

TABLE 5.6. CONFIRMATION VALUES FOR EACH CURVE, FROM AM.
successive listing of CF values and the final CP value for the '5' and the 'S'.

5.1.5. Updating the AV

Each decision is validated by the outside teacher. If correct, or in certain cases when the decision is incorrect but the input is similar, (e.g. CP > 0.8), the orientations of corresponding line segments in the AV and the processed input are averaged. The mean of the corresponding values for the current number of learning samples for the AV is taken. In the other cases of incorrect decisions, a new AV (which may be a subset) is formed.

Associated with each AV is a store of orientation values for all of the line segments found in the samples. This store includes 'stretched' elements and extraneous segments. Associated with each value is the frequency of occurrence of the segment. If a segment has appeared in at least half of the samples, then it is included in the current description of the AV. All such segments are given to the routine CVE which forms the curve types shown in Table 5.2.

Table 5.7 shows the stored orientations for the figure '5', (before and after the given learning sample), and the curve types in the new AV. The CVE routine did not produce a 'good' curve description in this example - curves 3 and 4 should have been joined. Note that the 6th segment in the store is included into the current AV because the 'stretch' in the input with which it was averaged increases the frequency of occurrence of that segment above 0.5.
### BEFORE SAMPLE

<table>
<thead>
<tr>
<th>AP</th>
<th>FO</th>
<th>AP</th>
<th>FO</th>
</tr>
</thead>
<tbody>
<tr>
<td>-30</td>
<td>1.00</td>
<td>-30</td>
<td>1.00</td>
</tr>
<tr>
<td>-30</td>
<td>0.66</td>
<td>-30</td>
<td>0.75</td>
</tr>
<tr>
<td>20</td>
<td>0.66</td>
<td>-30</td>
<td>0.25</td>
</tr>
<tr>
<td>140</td>
<td>1.00</td>
<td>19</td>
<td>0.75</td>
</tr>
<tr>
<td>120</td>
<td>0.33</td>
<td>55</td>
<td>0.75</td>
</tr>
<tr>
<td>102</td>
<td>1.00</td>
<td>139</td>
<td>1.00</td>
</tr>
<tr>
<td>45</td>
<td>0.66</td>
<td>116</td>
<td>0.50</td>
</tr>
<tr>
<td>-22</td>
<td>1.00</td>
<td>95</td>
<td>1.00</td>
</tr>
<tr>
<td>-30</td>
<td>0.66</td>
<td>45</td>
<td>0.75</td>
</tr>
<tr>
<td>-35</td>
<td>0.66</td>
<td>-21</td>
<td>1.00</td>
</tr>
<tr>
<td>-41</td>
<td>0.66</td>
<td>-30</td>
<td>0.75</td>
</tr>
</tbody>
</table>

### AFTER SAMPLE

<table>
<thead>
<tr>
<th>AP</th>
<th>FO</th>
<th>AP</th>
<th>FO</th>
</tr>
</thead>
<tbody>
<tr>
<td>-30</td>
<td>1.00</td>
<td>-30</td>
<td>1.00</td>
</tr>
<tr>
<td>-30</td>
<td>0.75</td>
<td>-30</td>
<td>0.25</td>
</tr>
<tr>
<td>20</td>
<td>0.25</td>
<td>19</td>
<td>0.75</td>
</tr>
<tr>
<td>140</td>
<td>1.00</td>
<td>55</td>
<td>0.75</td>
</tr>
<tr>
<td>120</td>
<td>0.33</td>
<td>139</td>
<td>1.00</td>
</tr>
<tr>
<td>102</td>
<td>1.00</td>
<td>116</td>
<td>0.50</td>
</tr>
<tr>
<td>45</td>
<td>0.66</td>
<td>95</td>
<td>1.00</td>
</tr>
<tr>
<td>-22</td>
<td>1.00</td>
<td>45</td>
<td>0.75</td>
</tr>
<tr>
<td>-30</td>
<td>0.66</td>
<td>-21</td>
<td>1.00</td>
</tr>
<tr>
<td>-35</td>
<td>0.66</td>
<td>-30</td>
<td>0.75</td>
</tr>
<tr>
<td>-41</td>
<td>0.66</td>
<td>-36</td>
<td>0.75</td>
</tr>
</tbody>
</table>

### UPDATED AV:

- **XM=4**
- **C1**
- **C2**
- **C3**
- **C4**

- **VC**
- **0**
- **49**
- **-23**
- **-50**

- **AT**
- **139**
- **95**

### TABLE 5.7. UPDATING THE AV IN AM (TEST 2).

**XM:** NO. OF CURVES (C1, C2, ...).

**VC:** SLOPE DIFFERENCE VALUES FOR EACH CURVE.

**AT:** ORIENTATION OF FIRST LINE SEGMENT IN THE CURVE.

**TP:** THE TYPE OF CURVE.

**FQ:** FREQUENCY OF OCCURRENCE.
5.2. Modifications to the First Program, AM

3.2.1. Orientation Independence

It was desired that the final program should be capable of recognizing characters in any orientation. A major problem in achieving this, caused by the 'bug-on-a-line' approach (Section 4.2) is the determination of when a character is rotated.

Consider the AV shown (pictorially), below left and the first part of an input, shown below right.

\[ \hat{2} \]

There are two possibilities* - one that the input is a rotated '2' (below left) and the other that the input has an extraneous line (below right).

\[ \n \hat{2} \]

Obviously, more of the input is required to make a satisfactory decision and both hypotheses should be generated, but these generations are not made in the program. Rather, the character is assumed to be rotated, by virtue of the orientation of the first line segment. The programs form an 'average' orientation for the input from the difference between the orientations of first line segments in all curves. Because of this procedure, the programs are not very independent of extraneous segments. The processing of extraneous segments is considered a separate problem (see Section 6.1).

* For the input to confirm the '2' AV.
Two programs have been developed for orientation independence and both are briefly outlined below.

5.2.1.1. Orientation Independent Program, GA

The following changes to AM are incorporated:

Storage: An attempt is made to reduce the storage of the AV. Instead of retaining each slope difference value, a total curvature value (AO) is kept. The prediction of the orientation of line segments assumes equal slope changes between each line segment in the curve.

Comparison: A modification to the PAS-FUT range method of comparison is made, (the variables PAS and FUT are not used); the outcomes of the comparison however are the same as for AM.

Confirmation: There are two measures for each curve - length, in terms of the number of line segments, (ZM), and shape, the CF measure of AM. There are two measures included in the total confirmation for the input figure - orientation, in terms of the average deviation compared to 180°, and the number of curves.

Averaging: If the decision is correct, then the values of orientation, (AT), total curvature, (AO), and number of segments, (ZM), are averaged (by taking the mean) over all given samples of the AV. If incorrect, then another AV is formed.

The lack of the model to discriminate between variants of a curve having the same curvature (e.g. a well-formed curve and a loop) causes many errors. The AVS for a '6' and '0' in one test are very nearly the same, (Test 14),...
and are unsatisfactory for discrimination:

| '6' :  | AT $0^\circ$ | AO $346^\circ$ | ZM 8 |
| '0' :  | $-11^\circ$  | $341^\circ$    | 9    |

It would appear that another set of curve types would provide better results.

Although the comparison rules for AM and GA are similar, GA shows a tendency for 'premature jumping' - that is, matching input segments to those of a following curve rather than the present (correct) one. A limit is placed on the ZM value at which a jump is possible, but some errors still occur. The results indicate that the PAS-FUT range is a better means of comparison.

A major problem is the lack of an effective way to average over the curve, caused by not maintaining a segment store as in AM. The curve types used in the AV are those found in the first sample. Taking the smallest number of curve types found in any number appears a possible approach, but does not hold for all classes. The idea of inserting transformational rules (see Appendix A) would also appear to be dependent on the actual input figures.

Because of the above problems, the approach of GA was abandoned and a program (designated AN), more similar to AM, was developed.

5.2.1.2. Orientation Independent Program, AN

The following changes to AM are incorporated:

**Storage:** The same values in the AV as in AM are stored. The prediction, as in GA, is made independent of the expected orientation of the figure.

**Comparison:** The comparisons are performed on the slope difference values (rather than the orientation
values, as in AM).

**Confirmation:** The measure includes that for shape, (CF), orientation and number of curves, used in GA.

**Averaging:** The averaging is performed over the slope difference values between line segments. The first element of the AV array (see Section 5.1.5) contains the orientation of the first line segment.

The initial results from AN appeared promising, and the features of this program were used in further improvements, described in the following Sections 5.2.2 to 5.2.6.

### 5.2.2. Segmentation

In accordance with the theory of Chapter 4, the specified segment forms are controlled by two factors:

(a) the segments are composed of similar elements, i.e. when the segments are repeated, the result is a 'good' approximation to a straight line.

(b) the complexity of the input stream is reduced by the extraction of the segments, i.e. the groupings consist of more than one element, (but less than a certain number).

The set for AM has been presented in Section 5.1.3.

There is an inherent length asymmetry associated with the AM segments. Consider a line at 30° and another at 0°, shown below:

(a) (b)
Segments of type 'xxxx' are found in (a) while those of type 'xy' are found in (b). In AM, the only information passed to the curve level is the orientation of the segments. Thus, lines at intervals of 60° from 0° appear longer than those at intervals of 60° from 30°. The result is poor discrimination between figures which depend on the differences in the length of the segments, (e.g. '6', '0').

Two improvements to overcome this problem have been attempted. In the first approach, a fixed length segment is used. A length of 3 elements is chosen because,

(i) the maximum tolerance (PAS-PD) required is 20°, (compared to the 2-element case of 30° and the 1-element case of 60°).

(ii) the segments correspond to approximately equiangular intervals of 20°, which is not true for segments with more elements.

Thus a segment may have the following forms: - xxy, xyx and yxx where y=x±1 and x=1, 2...6.

Because of the problem with concatenating elements into the sequence, single and double element segments (e.g. '2' '22') are allowed when y=x±2, 3 or 4.

With such a procedure, no prediction is required and hence a suitable 'input' would be sets of 3-element segments.

The program AN3 incorporated this segmentation, together with the revised confirmation measures, (see Section 5.2.4). Considerable improvement over AM resulted as shown in Table 5.11 (p.94). A modification of AN3, designated OM, which accepts the equivalent of 3-element segments, has been used in an on-line recognition study, described in Appendix D. The performance
of this system has substantiated the above results for AN3, and has shown that the program OM is an effective character recognizer for an on-line user.

It was desired, however, to follow the theory by incorporating the larger set of segments into the program. The second alternative accordingly involves the use of the length of the segments as a factor in the confirmation measure. Thus, discrimination depends also on a function of the length of the constituent segments. The beginning-end vector of each curve has proved to be a suitable measure, (see Section 5.2.4).

As previously inferred in this Section (5.2.2), the discrimination ability of AM is degraded by the inclusion of extra elements in certain segments. The reason for incorporating this procedure is for the factor (a) mentioned above. An example of the decrease in performance of AM is given in the results of Test 4.6, in which different forms of '2' are to be recognized. The difference between types A and B (shown below) occurs in a small number (2 to 4) of elements (in the asterisked parts marked in the figure below).

\[ \begin{array}{c}
\ast \ 2 \\
A \\
\end{array} \hspace{2cm} \begin{array}{c}
\ast \ 2 \\
B \\
\end{array} \]

Concatenation of elements into a Type B sequence makes that figure very similar to a Type A.

The above problem has led to the formation of a larger set of segment types, namely:

(a) those included in the AM set, plus

(b) \( x y x, \)
\( x y x x, \)
\( x y x x x, \)
\( x \times y \times x, \)
\( x y x x x, \)
\( x y x x x, \)
\( x x y x x, \) and
(c) in cases where the above segment types cannot be formed, x, xx and xxx. The length of each segment formed is also noted. This set is used in program AN6.

While this set does overcome problems in segmentation, the partitioning sometimes formed does not correspond to that expected from a search for patterns. For example, the 'G' AV of Test 8, does not have a straight line (curve) at the end; the '5616161' ending, is (in a given example) partitioned as '56/161/61/', and the 'G' is described by one curve, (c.f. --5/61/61/61).

To determine the effect of AV prediction on the description given to an input, and hence the results of the program, the rule R1 (Section 4.2) has been used to generate the segments - that is, the input segments are dependent only on the slope difference of the two previous segments. The program GB incorporates this feature, and in all other respects GB is identical to program GA. Examination of the results given in Table 5.11 (p.94) shows a marked decrease in recognition performance. It would seem that an expectation of changes in curves, especially the appearance of sharp corners, is important for the programs in segment formation and, hence, shape discrimination.

5.2.3. Tolerance

The value of the PAS-FUT range is an important factor in the recognition ability of the programs. If the tolerance is increased, then discrimination between similar figures is more difficult. The minimum value for the PAS-PD (equal to the PD-FUT) range depends on the approximation used to represent the sequence by the segment types. For example, the input,
2 2 2 2 1 2 2 2 2 2 1 represents a straight line but must be approximated by the following segments, from the AM set,
2 2 2 2 / 2 1 / 2 2 2 2 / 2 1 which have (+ or -) 30° difference between successive slopes. Hence the minimum tolerance for the PAS or FUT values must be, at least, 30°; otherwise, the outcome of the segment comparison could be a 'premature jump' or even a purging of the (correct) hypothesis.

Another problem is that some sequences which represent curves of large radius, can be partitioned to have a local change of sign in the slope differences. Consider, for example, an expected set of segments, 4 3 3 3 / 4 3 / 4 4 4 4 and the input 4 3 3 3 3 4 3 3 4 4 4 4. Segmentation according to the 'closest' segments gives 4 3 3 3 / 3 4 / 3 3 4 / 4 4 4 4 which has a change of sign in the slope differences. (Other segmentations are possible without this effect). Some amendments to the comparison rules can be made to overcome such problems. However, these amendments do not produce satisfactory results in general, and some errors do occur because of 'poor segmentation'. It is to be noted, however, that the extended (AN6) set of segment types and a minimum PAS (FUT) tolerance of 20° has produced no (incorrect) purgings in any of the tests given to AN6.

5.2.4. Confirmation Measure

The confirmation measure is used to determine the comparative similarity between a given input description and an AV, and must extract the essential factors by
which members of the classes can differ. The factors studied here include the shape (angle variation and length) and orientation of the figures. Extensions to the CF measure of AM (Section 5.1.5) are discussed below.

5.2.4.1. Size

It was desired that the recognition of the figures be size independent, and accordingly, CF was modified by a SIZE value, viz.

\[
\text{CF} = \begin{cases} 
\text{CF}/\text{SIZE} & \text{if } \text{SIZE} < 1 \\
\text{CF} \times \text{SIZE} & \text{if } \text{SIZE} \geq 1 
\end{cases}
\]

\text{and } \text{SIZE} = K/KM,

where K is the number of line segments in the input, and KM is the number of line segments in the AV. Segments, rather than direction elements, are used to determine the size, because of the dependence of CF on the segments. However, their use leads to a bias in the size value for straight lines, due to the types of segments extracted, (see Section 5.2.2). Thus for the same number of elements, a straight line at different orientations can have a different 'size'. This effect is noticed in Tests 1 and 6, with AN6.

It would appear that, for some inputs, a size for each curve (rather than for the whole figure) should be calculated. Examples suggesting this are figures which possess an exaggerated proportion, e.g.

\[\text{compared to} \quad 2 \]
and for which the CF and the beginning-end measure (Section 5.2.4.3) decrease out of proportion.

A more difficult determination arises when the exaggerated proportion is not simply an increase in size, but a distortion, e.g.

\[ \begin{array} {c}
\text{D} \\
\text{compared to} \\
\text{D} 
\end{array} \]

In this case the size as calculated above is incorrect and, as in the problem of the previous paragraph, errors are sometimes made, (e.g. in Test 10). To overcome these problems is a matter for future research.

5.2.4.2. Orientation

An orientation is calculated for each input curve, by comparing the orientation of the first line segments in each input curve with that expected. A measure is formed by comparing the above difference with \(180^\circ\) - its value is unity when there is \(0^\circ\) difference, and zero, when there is \(180^\circ\) difference.

As mentioned in Section 5.2.1, the value of the measure depends on extraneous segments in the figure. Naturally, some extraneous segments are allowed, but the number depends on the actual input and the similarity of the AVS.

If the orientation of the characters is unimportant or meaningless (e.g. if the same figures are to be drawn from different beginnings (e.g. Test 16)) then the measure must be removed from the total confirmation. This option (for the user) can be easily incorporated in the programs.

The 'shape' measure, CF, does take some account of the
length of the curve by virtue of the restrictions on the form of the line segments. However, the amount considered is insufficient because of the necessarily wide tolerances in the segment comparisons.

The first attempt to account for length, incorporated the distance along the curve. However, this is a poor measure as can be seen from the figures '6' and '0'. Assuming one curve for each figure, the line length for each figure is approximately the same, (see Table 8.14, p.181) giving little extra information for the decision of class membership.

The consideration of the beginning-end relationship of each curve is more fruitful. A measure for this vector has to be context-dependent, as shown in cases (a) and (b) below for a '0' and a '6'.

\[ \begin{align*}
(a) & \quad (b)
\end{align*} \]

In (a), the important factor is the distance, while in (b) the direction also becomes relevant. A minimum value of 4 units (direction code lengths) is prescribed for the expected beginning-end distance, below which the beginning-end angle is not considered. If the distance is greater than 4 units, then the difference in beginning-end angle (between that expected and obtained) is incorporated into the measure by a 'COS' function. When the angular difference is greater than or equal to 90°, the value of the measure is zero.

The total confirmation used in AN6 for an input is determined by the sum of the values for the above three measures (shape, orientation and length) for each input curve,
divided by the total number of curves expected.

There are some relationships which exist in certain figures and which are not extracted by the measures. The 'true' value for size is a factor which, as explained in Section 5.2.4.1, is not obtained. Another is outlined below:

Assume that all lines in the figures 'X' and 'V' (shown below) are straight, and that when the pen is lifted from the page, straight lines are formed, (shown dotted). The drawings proceed from B to E.

The important factor in discriminating these figures is the relation of slope of the full lines, compared with the length of the dotted line. This relation cannot be extracted from the measures suggested above. Note, however, that other ways of tracing the figures, e.g.

\[ \begin{align*}
X & \quad V \\
 B & \quad E & E & \quad B \\
\end{align*} \]

do not present the same problem.

5.2.5. Averaging

The averaging of line segments in an AV is performed on the slope differences between the segments and the orientation of the first segment, in the orientation independent programs. Obviously for these programs, all learning samples are required to have approximately the same orientation. Differences in the first segment of the inputs averaged in the same AV, (e.g. caused by extraneous segments) appear
at the end of the first curve in the stored slope values. This occurs because the first input segment is always matched to the first segment in the AV. Unmatched segments in the first curve, therefore, appear at the end of that curve.

In the programs, the slope and slope difference values can be any integer from -180 to 180. Experiments using a 10° quantization on the values in the AVS (for Test 14) show no decrease in the program performance. However, a 20° quantization causes some errors, an additional 4 out of 40).

The method of AV formation leads to poor results in some cases. A new AV is formed from the slope values for the input line segments which have been found as 'similar' to a known AV. If this generating AV is in fact similar to the input figure (e.g. a '5' generating an 'S') then a minimum 'difference' exists between the new AV and the AV of the generating class, thus making discrimination more difficult. If the generating AV is not similar (e.g. a '2' generates an 'S'), then the slope difference values formed from the input will not be conducive to 'good' curve description. This effect can cause errors in future inputs. For example, if an AV has a number of sharp corners, (when a good description does not have them), then because of the flexible nature of the comparison rules, premature jumping and even purging of the(correct) hypothesis can occur.

In both of the above cases of generation, the effects of poor AV description, can continue for a number of samples, before being reduced to a negligible degree by averaging. A method for avoiding the problems (but not incorporated in the programs) is to have each new AV
generated in the same manner as the very first input sequence (i.e. independent of the stored AVS).

5.2.6. Curve Determination for the Current Model

The curve types are formed on the basis of the magnitude and sign of the slope difference values. Three routines have been developed:

(a) SIMP, in which the rules for curve formation are dependent on actual values, e.g. a positive sharp corner is defined in terms of a slope difference of greater than 80°.

(b) COMP, in which all rules are dependent on relations between values, e.g. a positive sharp corner is defined by a value being at least twice as great as the previous and following values.

(c) CVE which is a combination of (a) and (b).

The SIMP routine generally produces few types, while COMP is sensitive to changes in the slope difference values. CVE gives results usually similar to one of the above, ((a) or (b)), and is used in all of the described programs. A comparison of curve types from each routine for a set of figures is given in Table 5.8.

Table 5.9 shows further examples of the kind of curves generated by CVE, for the numerals 0 to 9. It will be noticed that the curve types have rather general shapes. All the numerals have less than 6 curves per figure; the figure '4' has 5, while the '1', '6' and '0' have one.

The following comments apply to the CVE routine: The requirement for a straight line to be formed is that at least two consecutive slope difference values, each less than 24°, occur. Because of the nature of the
FIGURE COMP CVE SIMP

\begin{tabular}{ccc}
12 & 4 & 1515 \\
15 & 5 & 134 \\
16 & 3 & 515 \\
10 & 5 & 533 \\
11 & 3 & 515 \\
12 & 3 & 515 \\
13 & 3 & 515 \\
14 & 3 & 515 \\
15 & 3 & 515 \\
16 & 3 & 515 \\
17 & 3 & 515 \\
18 & 3 & 515 \\
19 & 3 & 515 \\
20 & 3 & 515 \\
21 & 3 & 515 \\
22 & 3 & 515 \\
23 & 3 & 515 \\
24 & 3 & 515 \\
25 & 3 & 515 \\
26 & 3 & 515 \\
27 & 3 & 515 \\
28 & 3 & 515 \\
29 & 3 & 515 \\
30 & 3 & 515 \\
31 & 3 & 515 \\
32 & 3 & 515 \\
33 & 3 & 515 \\
34 & 3 & 515 \\
35 & 3 & 515 \\
36 & 3 & 515 \\
37 & 3 & 515 \\
38 & 3 & 515 \\
39 & 3 & 515 \\
40 & 3 & 515 \\
41 & 3 & 515 \\
42 & 3 & 515 \\
43 & 3 & 515 \\
44 & 3 & 515 \\
45 & 3 & 515 \\
46 & 3 & 515 \\
47 & 3 & 515 \\
48 & 3 & 515 \\
49 & 3 & 515 \\
50 & 3 & 515 \\
\end{tabular}

(NUMBERS REFER TO CURVE TYPES)

TABLE 5.8. CURVE GENERATION FROM 3 ROUTINES.

THESE FIGURES CORRESPOND TO THOSE IN FIGURE 5.1.

TABLE 5.9. CURVE TYPES FOR NUMERAL AVS.

segments (see Section 5.2.2), the actual number of elements which can form a straight line is dependent on its orientation. Thus lines at 60° intervals from 0° are formed with less complexity than those at 60° intervals from 30°. This anomaly could be overcome by taking length into considera-
tion. The purpose of this research was to determine comput-
ability requirements and whether or not it was necessary. It
was found that the tolerance of 24° was sufficient because the
function is kept small enough to allow 'wobbly' lines to be considered straight. In some cases, as shown in the
theoretical generalization to any size of input in the working
phase.

The inclusion of the beginning-end vector as a factor in the computation, the value for distance and
type of vector was determined. This is performed
in the Curve routine, based on a knowledge of the orientation
and length (known from the segment types) of each segment
in the normal curve sequence. In addition, the vector was calculated.

The possibility of changing the curve level as the
program designated DB. The sequence of curves is an
input (generated independently of the store) is matched to
the stored set for known members. If a match occurs,
then the decision is given to the class of that known
member. If the same (input) curve sequence belongs
to more than one class, factors such as total curvature
and orientation for each curve are used as further
segments (see Section 5.2.2), the actual number of elements which can form a straight line is dependent on its orientation. Thus lines at 60° intervals from 0° are formed with less elements than those at 60° intervals from 30°. This anomaly could be overcome by taking length into consideration, but this was not done, because the additional complexity required for the program was deemed unnecessary.

The tolerance of 24° is a specification which depends on the nature of the figures. It is kept 'high' to allow 'wobbly' lines to be considered straight. In some cases, as shown in the example of AM (Section 5.1.6), this factor leads to 'poor' curve formation. Because of the 24° specification, it means that the learning samples must be of a certain (approximate) size, although the program can theoretically generalize to any size of input in the working phase.

The inclusion of the beginning-end vector as a factor in the confirmation, requires the value for distance and angle of the vector to be determined. This is performed in the CVE routine. From a knowledge of the orientation and length (known from the segment types) of each segment in the formed curve, the distance and angle can be calculated.

The possibility of using the curve level as the lowest level for a basis for recognition, was investigated in a program designated DB. The sequence of curves in an input (generated independently of the store) is matched to the stored set for known members. If a match occurs, then the decision is given to the class of that known member. If the same (input) curve sequence belongs to more than one class, factors such as total curvature and orientation for each curve are used as further
discriminants.

The results are rather poor because,

(a) of the variation in the sequence of curve types within a given class - many members must be stored.

(b) the secondary factors are weak discriminators, (see GA; Section 5.2.1).

Examples of the stored members for two tests are shown in Table 5.10. Note that there is little difference between B(3) and C(1) in Test 4.6. The samples of 'V' in Test 2 are the only ones to show any cohesion in the sequence of curve types.

An idea that the variations could be eliminated by some reduction rules, suggested further experimentation. Another series of programs has been developed as a start to the recognition of connected sequences of letters. This part of the project is explained in Appendix A.

The AM program was modified (program designated FA) to recognize figures without the curve level in the model. In this situation, the AV is stored as a set of orientation values for the constituent line segments. The results of FA show that:

(a) the comparison rules can be made effectively the same. However, more programming is required to incorporate 'look ahead' and 'look back' procedures which are conveniently handled with curves which provide the general trends.

(b) the confirmation measure (using CF) is less effective in FA, because more weight can be given to curve types rather than individual line segments. For example, the absence of the straight line segments in the input for the first part of a '5' can be weighted more heavily if it is treated as one of three
EXPERIMENT 1 (TEST 4.6).

(3 SAMPLES OF EACH CLASS, \( (A, B, C) \)).

<table>
<thead>
<tr>
<th>CURVE TYPE</th>
<th>ORIENT. (EACH CURVE)</th>
<th>CURVATURE (EACH CURVE)</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>1. 4 1 5</td>
<td></td>
</tr>
<tr>
<td></td>
<td>2. 4 5 1 5</td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>1. 5 4 2 5</td>
<td></td>
</tr>
<tr>
<td></td>
<td>2. 5 4 3 5</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3. 4 3 5</td>
<td>161 -30 150</td>
</tr>
<tr>
<td></td>
<td></td>
<td>-191 180 0</td>
</tr>
<tr>
<td>C</td>
<td>1. 4 3 5</td>
<td>163 -15 150</td>
</tr>
<tr>
<td></td>
<td>2. 5 4 3 1 5</td>
<td></td>
</tr>
</tbody>
</table>

EXPERIMENT 2 (TEST 1).

(6 SAMPLES OF EACH CLASS, \( (U, V) \)).

<table>
<thead>
<tr>
<th>U</th>
<th>1. 5 3</th>
<th>(3 MEMBERS)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>2. 3</td>
<td>(2 &quot;&quot;)</td>
</tr>
<tr>
<td></td>
<td>3. 1 5 3</td>
<td>(1 &quot;&quot;)</td>
</tr>
<tr>
<td>V</td>
<td>1. 5 1 5</td>
<td>(6 &quot;&quot;)</td>
</tr>
</tbody>
</table>

TABLE 5.10. SUBSETS FORMED BY PROGRAM DB.
curves, than as three of fourteen line segments. However, the results in general for the tests given are similar to those of AM, (see Table 5.11, p.94).

5.3. Program AN6

Program AN6 contains several improved features over AM, viz:-

(a) orientation independence, (see Section 5.2.1),
(b) the extended line segment set, (see Section 5.2.2),
(c) a minimum tolerance of $20^\circ$ for PAS and FUT, (see Section 5.2.3),
(d) confirmation measure with orientation, shape (modified CF) and beginning-end relation factors, (see Section 5.2.4),
(e) a total confirmation measure of 0.75 for an incorrect decision by a known AV, above which the input can be averaged into the AV,
(f) the modified CVE routine, (see Section 5.2.6).

The actual program, with comments, is presented in Appendix B, wherein its specific operation can be referenced.

The program AN5 is a version of AN6 without the learning procedures. The initial input to AN5 therefore includes the values for all the AVS. AN5 is about half the size of AN6 and its turn-around time is consequently decreased.

AN6 has been given the following Tests:- 1, 2, 3, 6, 10, 11, 12, 13, 14, 2, 18. The AN and AN6 programs have been used to generate AVS for AN5; in other cases the AVS given have been specified by the author. AN5 has been tested on the following:- 4, 6, 8, 9, 11, 14, 17, 19, 20. The final program has accordingly been given approximately 170 learning samples and 240 test samples.

Details of the results of the program (i.e. of either AN6 or AN5) are presented in four ways:-
(i) comparative performances between the various programs in this series,
(ii) some examples of the numerals, 0 to 9, and the results of their recognition,
(iii) a discussion of the typical errors and examples of the confirmation measures,
(iv) results of some tests on learning.
These results are discussed respectively in Section 5.3.1, 5.3.2, 5.3.3 and 5.3.4.

5.3.1. Comparative Performances of Programs

The performance of AN6 (and AN5) compared with other programs developed in the project, is shown in Tables 5.11 and 8.13 (pp. 177, 178). In Table 5.11 results for the learning samples (the same set in each Test, unless otherwise stated) are presented separately from those for the working samples. It should be remembered that errors are always made on the first member for each class in the learning phase. AN2 and AN5 are programs which require AVS to be given to them as input, but embody the methods of AN and AN6, respectively.

The results indicate the superiority of AN6 (and AN5) over the other programs. The poor performance of AN6 in Test 8 is due to the ill-formed AVS, for which the '6' AV has a straight line ending, while the 'G' AV does not, due to a problem in segmentation (Section 5.2.2). The importance of the beginning-end relationship in the confirmation measure for AN6 can be seen in the results of Test 14. It is interesting to note that the total curvature (a feature in GA) is a useful property to discriminate between the 'e' and 'c', in Test 9.

5.3.2. Results of Numeral Recognition

The most extensive results of AN6 (and AN5) have been
those concerning the numerals 0-6, in Tests 11, 17, 19 and 20. The particular learning sequences given to AN6 (and studied here) consisted of 5 samples of each of the 10

These results have not been dressed in Figure 5.

There were also 2 other numerals 0-6, which did not appear in the sequences. It is impossible to assess the performance of these numerals on the basis of the panels in Figure 5.

As a better indication of the performance of AN6, the samples of Test 20, (similar to Test 19), are shown in Figure 5.4, and the results (shown in Figure 5.4) are shown below. Reference to the AVS (in Figure 5.4) shows that 17.2

It should also be pointed out that the programme always gives a decision, so that every input presented will be classified. (The examples (3.1.2).

(1ST LINE GIVES THE LEARNING PHASE RESULTS;
2ND LINE GIVES THE WORKING PHASE RESULTS;
EXAMPLE: 3/6 = 3 ERRORS IN 6 SAMPLES).

TABLE 5.11. COMPARATIVE PERFORMANCES OF PROGRAMS.
those concerning the numerals, 0-9, in Tests 11, 17, 19 and 20. The particular learning sequences given to AN6 (and studied here) consisted of 4 samples of each of the 10 characters, from Test 11. There were only 10 errors (the first one in each class) and most characters were averaged over the four samples. The AVS have been drawn in Figure 5.1.

These AVS were then tested on other numerals in Test 11.2, and Tests 17.1, 17.2, 19 and 20, (see Table 5.1). It is difficult to assess the performance of a program from such tables - for example, the appearance of poor results 11.2, 17.1 and 17.2 is offset by noting that 11.2 and 17.1 contain scribbled characters and that 17.2 contains the writing of another person.

As a better indication of the performance of AN5, the 33 samples of Test 20, (similar to Test 19), are shown in Figure 5.2, and the results from this Test are discussed below. Reference to the AVS (Figure 5.1) should be made when assessing the performance of AN5 on these samples. It should also be pointed out that the program always gives a decision, so that every input presented will be assigned to a character class, (see Section 5.3.3.2).

The following results have been produced by AN5:- Nos. 11, 13, 16, 20 and 28 (of Figure 5.2) are given incorrect decisions - the decisions are '8', '9', '5', '9' and '8' respectively. (Note that the correct decisions for Nos.13 and 28 are assumed to be '6' and '0' respectively - the other assignments should be obvious). When the figures are rotated 180°, Nos. 8 and 9 are both classified as a '5', while the remaining receive the same classification as when not rotated.

The following features of the program can be noted from
Figure 5.1. The Average Members Constructed by AN6, in Test 11
Figure 5.2. The Input Figures of Test 20
Figure 5.2 (Ctd.). The Input Figures of Test 20
Figure 5.2 (Ctd.). The Input Figures of Test 20
these results:-

(a) the ability to confirm straight lines and sharp corners where smooth curves are expected, (Nos. 6, 10, 14, 22, 26, 29).

(b) the ability to confirm smooth curves where straight lines and sharp corners are expected, (Nos. 1, 2, 18, 31; although No. 11 is in error).

(c) the insensitiveness to some forms of 'noise' in the slope difference values, (Nos. 4, 5, 24, 30; although Nos. 13 and 28 are in error).

(d) the ability to cope with size variations. The following presents the increases in size for some of the figures, as determined by the SIZE variable, (see Section 5.2.4.1)

<table>
<thead>
<tr>
<th>No.</th>
<th>Increase in Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>1.44</td>
</tr>
<tr>
<td>19</td>
<td>1.78</td>
</tr>
<tr>
<td>31</td>
<td>1.50</td>
</tr>
<tr>
<td>32</td>
<td>1.13</td>
</tr>
<tr>
<td>33</td>
<td>2.33</td>
</tr>
</tbody>
</table>

(e) the ability to deal with some types of extraneous segments, (Nos. 12, 15, 25; although notice that extraneous parts on Nos. 11, 20 and 28 cause errors).

(f) the confirmation of only small parts of a curve, Nos. 3, 7, 27), or exaggerated proportions, (Nos. 21, 23, 32; but notice the error in No. 16).

(g) the allowable tolerance on orientation of lines, (Nos. 7, 8, 9, 17, 24).

5.3.3. Causes of Errors

This Section contains a brief discussion of the causes of errors, found in the testing of AN5 and AN6. Reference is made to the samples in Test 20.

5.3.3.1. Tolerance on the Curve Types

The tolerance on allowable slope differences for each
curve type, embodied in the comparison rules, does not appear to match the flexibility allowed by a human observer. For example, a curve of positive slope difference (Type 3) will usually not be confirmed by a slope difference of -20°. If such a value occurs, an attempt will be made by the program to match it in the next curve, and if this cannot be made, the hypothesis will be purged. In some cases, this -20° change may represent a 'local change' in the drawing - one which the average human observer might ignore, (although the performance of this 'average' person is ill-defined). The program will therefore be in 'error' on such inputs (e.g. Nos.13,28), but nevertheless it does allow some tolerance.

Some errors have been caused by the lack of confirmation of a curve (Type 3 or 4) to a sharp corner (Type 1 or 2) of the same sign. The beginning-end and shape values for the confirmation decrease rapidly when the number of segments increases in the match to a corner, (e.g. No.11). No.28 is not classified as a '9' because the small curvature of the last line, does not confirm the curvatures of the corner expected for a '9'. Instances of curves (instead of corners) are often found (in the author's writing) when the speed of drawing increases. This problem of confirmation is also found when poorly formed AVS with many corners process more well-formed numbers. Because of the nature of the comparison rules, the matching of the input segments to the correct parts of the AVS becomes more prone to error. The design of these rules and the curve types is extremely important in this recognition scheme.

5.3.3.2. Hypothesis Purging

If all hypotheses are purged, (actually the last
one to be purged is always kept to predict the input), the
decision is given to the one which 'stayed' with the input
for the maximum number of input elements. (e.g. Nos.4, 20).
In some cases (e.g. No.20) this criterion may not give the
correct decision because other hypotheses, purged at nearly
the same time as that given the decision, may have had a
greater value of confirmation.

There is a need for a program to be capable of purging
hypotheses which, although the signs of their slope
differences are similar, the overall figure bears little
resemblance to the members of a class. Thus, in No.20,
the hypothesis for the figure '9' should have been purged
when the total curvature for the 'sharp corner' passed
(say) 180°. It is known that many inputs would be assigned
classes by the program, for which a human subject would
be inclined to give a 'not similar to any class' category.
Further programming, possibly decreasing the flexibility
of the comparison rules, is required to overcome the above
problem.

5.3.3.3. Similarity Measure

The approximate method used to determine the size,
gives poor values for distorted figures. Size values
have been given in Section 5.3.2 for some figures of Test 20.
Note that No.32 is only 1.13 times as great as the AV, based
on line segments, but that the actual increase is confined
to the initial downward strbke.

The program can recognize figures which are rotated,
provided that the rotation does not make the figure more
similar to another class. There is, however, some bias
in the decision, because whereas the '4' will be recognized
as a '9' or '5' in different orientations, the '9' or '5'
(in all tests attempted) is still recognized as such when
rotated, (e.g. Nos. 8,9).

The angular dependence in the 'beginning-end vector' measure seems to decrease too rapidly with increasing difference, (Θ) - as evidenced by the error of No.16. A suggestion here is to allow Θ to range between 0° and 180° (magnitude) and to make the measure dependent on \( \cos(\theta/2) \).

The combination of measures is not yet optimized and some work remains in this regard. It appears that the beginning-end measure is too important in relation to the other measures. Some anomalies have been mentioned above; another, is the relatively small decrease given to the confirmation value for the absence of a curve in the input. It is felt that the decision of Nos.13 and 16 are poor, (see Table 5.12). Further work is necessary to increase the ability of the measures to extract more relationships and properties of the figures (see Section 5.2.4).

5.3.4. Similarity Values

Examples of the values for the various similarity measures for the inputs Nos.3,9 (rotated 300°), 11 and 16 are given in Table 5.12. In each case, the purged values are presented first, in the order of their purging. Figures with opposite sign in their initial slope difference are usually the first to be purged. The confirmation values for orientation (CO), beginning-end vector (CL) and 'shape' (CF) measures and the total of these (TOT)(averaged over each expected curve) are presented for each 'similar' hypothesis (H).

The following points can be noted from Table 5.12: -
(a) In No.3, the effect on the CL and CF measures of not
TEST 20 NO. 3:

PURGED: '6' '8' '9' 'C' '4' '5' '1' '7'

<table>
<thead>
<tr>
<th></th>
<th>CO</th>
<th>CL</th>
<th>CF</th>
</tr>
</thead>
<tbody>
<tr>
<td>H'2': SIZE=1.11</td>
<td>0.91</td>
<td>0.47</td>
<td>0.63</td>
</tr>
<tr>
<td></td>
<td>0.99</td>
<td>1.00</td>
<td>0.90</td>
</tr>
<tr>
<td></td>
<td>0.97</td>
<td>0.40</td>
<td>0.37</td>
</tr>
</tbody>
</table>

TCT=0.739  *

<table>
<thead>
<tr>
<th></th>
<th>CO</th>
<th>CL</th>
<th>CF</th>
</tr>
</thead>
<tbody>
<tr>
<td>H'3': SIZE=1.00</td>
<td>0.89</td>
<td>0.72</td>
<td>0.22</td>
</tr>
<tr>
<td></td>
<td>0.92</td>
<td>1.00</td>
<td>0.33</td>
</tr>
<tr>
<td></td>
<td>0.13</td>
<td>0.00</td>
<td>0.00</td>
</tr>
</tbody>
</table>

TOT=0.467

TEST 20 NO. 9 (ROTATED 300°):

PURGED: '1' '3' '2' '6' '7' '0'

<table>
<thead>
<tr>
<th></th>
<th>CO</th>
<th>CL</th>
<th>CF</th>
</tr>
</thead>
<tbody>
<tr>
<td>H'4': SIZE=0.90</td>
<td>0.56</td>
<td>0.70</td>
<td>0.56</td>
</tr>
<tr>
<td></td>
<td>0.59</td>
<td>1.00</td>
<td>0.56</td>
</tr>
<tr>
<td></td>
<td>0.81</td>
<td>0.61</td>
<td>0.37</td>
</tr>
<tr>
<td></td>
<td>0.53</td>
<td>1.00</td>
<td>0.90</td>
</tr>
<tr>
<td></td>
<td>0.59</td>
<td>0.52</td>
<td>0.00</td>
</tr>
</tbody>
</table>

TCT=0.620

<table>
<thead>
<tr>
<th></th>
<th>CO</th>
<th>CL</th>
<th>CF</th>
</tr>
</thead>
<tbody>
<tr>
<td>H'5': SIZE=1.00</td>
<td>0.88</td>
<td>1.00</td>
<td>0.50</td>
</tr>
<tr>
<td></td>
<td>0.89</td>
<td>0.93</td>
<td>1.00</td>
</tr>
<tr>
<td></td>
<td>0.92</td>
<td>0.36</td>
<td>0.20</td>
</tr>
</tbody>
</table>

TCT=0.742  *

<table>
<thead>
<tr>
<th></th>
<th>CO</th>
<th>CL</th>
<th>CF</th>
</tr>
</thead>
<tbody>
<tr>
<td>H'8': SIZE=1.00</td>
<td>0.86</td>
<td>0.41</td>
<td>0.44</td>
</tr>
<tr>
<td></td>
<td>0.83</td>
<td>0.30</td>
<td>0.11</td>
</tr>
</tbody>
</table>

TCT=0.492

<table>
<thead>
<tr>
<th></th>
<th>CO</th>
<th>CL</th>
<th>CF</th>
</tr>
</thead>
<tbody>
<tr>
<td>H'9': SIZE=0.90</td>
<td>0.96</td>
<td>0.40</td>
<td>0.74</td>
</tr>
<tr>
<td></td>
<td>0.39</td>
<td>1.00</td>
<td>0.37</td>
</tr>
<tr>
<td></td>
<td>0.52</td>
<td>0.78</td>
<td>0.74</td>
</tr>
</tbody>
</table>

TCT=0.656

* : RECEIVES THE DECISION.

TABLE 5.12. VALUES FOR SIMILARITY MEASURES IN AN6.
## TEST 20 NO.11:-

**PURGED:** '2' '3' '1' '7' '6' '0' '9'

### H'4'
- **SIZE=1.10**
  - CO  CL  CF
  - 0.66  0.50  0.73
  - 0.67  1.00  0.55
  - 0.89  0.35  0.27
  - 0.89  1.00  0.22
  - 1.00  0.00  0.00
  - \(TCT=0.610\)

### H'5'
- **SIZE=1.25**
  - CO  CL  CF
  - 0.98  0.49  0.80
  - 0.97  0.52  0.63
  - 0.92  0.82  0.70
  - \(TCT=0.758\)

### H'8'
- **SIZE=1.22**
  - CO  CL  CF
  - 0.96  0.59  0.61
  - 0.98  0.57  0.65
  - \(TCT=0.794\)

## TEST 20 NO.16:-

**PURGED:** '2' '3' '1' '7' '4'

### H'5'
- **SIZE=1.50**
  - CO  CL  CF
  - 0.98  0.45  0.67
  - 0.97  0.85  0.38
  - 0.83  0.00  0.00
  - \(TCT=0.5695\)

### H'6'
- **SIZE=1.50**
  - CO  CL  CF
  - 0.76  0.19  0.77
  - \(TCT=0.5693\)

### H'8'
- **SIZE=1.33**
  - CO  CL  CF
  - 0.96  1.00  0.33
  - 1.00  0.00  0.00
  - \(TCT=0.5484\)

### H'9'
- **SIZE=1.57**
  - CO  CL  CF
  - 0.99  0.48  0.86
  - 1.00  0.00  0.00
  - \(TCT=0.3700\)

### H'0'
- **SIZE=1.10**
  - CO  CL  CF
  - 0.82  0.33  0.49
  - \(TCT=0.5463\)

* : RECEIVES THE DECISION.

### TABLE 5.12. (CTD) VALUES FOR SIMILARITY MEASURES IN AN6.

finding the last curve for H'3;' in the input can be seen; the last segment, however, is assumed to be part of the last curve, (C0>0), but must have formed a 'stretch' or 'squeeze'.

(b) In No.9, the various orientation values are shown for the similar figures.

(c) In No.11, the similarity of the first part of the input to the figure '8' can be compared to that for the '5'.

(d) In No.16, the effect of the beginning-end angular difference can be seen in the CL value. Notice that because the last curve is expected in the '5', '8' and '9', (C0>0), the TOT value is higher than if it were assumed (correctly) that the last curve has not been found.

5.3.5. Learning

The conditions on the learning sample do not appear too stringent. In most Tests, a set of well-formed characters (about 4 per class) produces AVS which are also well-formed. The formation of subsets due to poor segmentation and curve generation (Section 5.2.5) is perhaps the major problem requiring user control. This factor is made difficult to control because of the dependence of subset formation on the similarity (and dissimilarity) of the generated AVS.

The averaging of the 40 numerals (Test 11) forms quite effective AVS (Figure 5.1). It will be noted (from Nos.2 and 3) that with few samples of the figure, the averaging of sharp corners produces distortion in the AV. However, this does not imply that the AV is unsatisfactory, because the comparison rules are flexible enough for a 'sharp corner' to be found in better formed figures which will confirm the
corner expected. The main effect is that the curve following the corner in the AV has a distorted beginning-end vector, but this has not been found to cause any errors. Some 'odd' shapes in Figure 5.1, namely,

(a) the second line in '5' is sloping,
(b) the '8' is not closed, and
(c) the loop in the '9' is small,
are, in fact, typical of the samples.

Two Tests (Nos. 12 and 13) have been designed specifically to examine the stability of the AVS. The first (No. 12) involves giving the program one 'C' followed by 26 'E' figures. Note that the 'C' is necessary because if only one class is given to the program, then every input will be averaged into the one AV. As the Test is now designed, the construction of subsets may occur, and will depend on the 'similarity' of an input to each AV.

Two AVS for the 'E' class are formed from the 26 samples—one AV exhibits a horizontal line and the other an inclined line at the beginning, (shown in Figure 5.3). These features are typical of the presented samples. Table 5.13 tabulates the type of decisions given (with respect to the averaging) to the 'E' samples. The numbers recognized in each AV correspond closely to the partitioning a human subject might give. However, there is a discrepancy between the machine and human performance—viz. some of the inputs were mixed (5 in all) by the program due to the poor segmentation, mainly in the initial part of the sequence, (see Section 5.2.2). Table 5.13 also shows the full set of averaged line segment orientations formed in each subset. Notice that poor segmentation resulted, as shown by the negative and zero values, which were eventually purged from the current AV.
Figure 5.3. The Average Members Constructed by AN6, in Test 12

Figure 5.4. The Average Members Constructed by AN6, in Test 13
TABLE 5.13. DECISIONS AND AV STORE (AN6 ON TEST 12).

<table>
<thead>
<tr>
<th>NUMBER OF SAMPLES IN EACH TYPE OF DECISION</th>
</tr>
</thead>
<tbody>
<tr>
<td>C&amp;A I&amp;A C&amp;-A I&amp;-A C: CORRECT</td>
</tr>
<tr>
<td>NUMBER: 8 11 2 1 1 2 0 0</td>
</tr>
<tr>
<td>SUBSET: 1 2 1 2 1 2 1 2</td>
</tr>
</tbody>
</table>

(I&A INCLUDES AV-GENERATING MEMBERS).

FIGURE 'E'

<table>
<thead>
<tr>
<th>AV 1</th>
<th>FQ</th>
</tr>
</thead>
<tbody>
<tr>
<td>AP</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>0.7</td>
</tr>
<tr>
<td>24</td>
<td>0.8</td>
</tr>
<tr>
<td>69</td>
<td>1.0</td>
</tr>
<tr>
<td>10</td>
<td>0.1</td>
</tr>
<tr>
<td>57</td>
<td>0.8</td>
</tr>
<tr>
<td>19</td>
<td>0.2</td>
</tr>
<tr>
<td>47</td>
<td>1.0</td>
</tr>
<tr>
<td>28</td>
<td>0.9</td>
</tr>
<tr>
<td>28</td>
<td>0.9</td>
</tr>
<tr>
<td>10</td>
<td>0.1</td>
</tr>
<tr>
<td>42</td>
<td>1.0</td>
</tr>
<tr>
<td>27</td>
<td>0.9</td>
</tr>
<tr>
<td>0</td>
<td>0.1</td>
</tr>
<tr>
<td>32</td>
<td>0.9</td>
</tr>
<tr>
<td>10</td>
<td>0.2</td>
</tr>
<tr>
<td>28</td>
<td>0.7</td>
</tr>
<tr>
<td>7</td>
<td>0.4</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>AV 2</th>
<th>FQ</th>
</tr>
</thead>
<tbody>
<tr>
<td>AP</td>
<td></td>
</tr>
<tr>
<td>-20</td>
<td>0.1</td>
</tr>
<tr>
<td>16</td>
<td>0.9</td>
</tr>
<tr>
<td>28</td>
<td>0.2</td>
</tr>
<tr>
<td>76</td>
<td>0.9</td>
</tr>
<tr>
<td>19</td>
<td>1.0</td>
</tr>
<tr>
<td>36</td>
<td>0.8</td>
</tr>
<tr>
<td>24</td>
<td>1.0</td>
</tr>
<tr>
<td>-5</td>
<td>1.0</td>
</tr>
<tr>
<td>41</td>
<td>0.9</td>
</tr>
<tr>
<td>27</td>
<td>0.9</td>
</tr>
<tr>
<td>22</td>
<td>0.6</td>
</tr>
<tr>
<td>3</td>
<td>0.3</td>
</tr>
</tbody>
</table>

TABLE 5.13. DECISIONS AND AV STORE (AN6 ON TEST 12).
The second test (No. 13) involves giving a 'Z', followed by 25 '2' figures of three types A, B and C, (see Table 3.2). Three subsets are in fact formed from the 25 '2' samples, and these are shown in Figure 5.4. Table 5.14 presents the curves for each AV and the number of samples in the different kinds of decisions. No incorrect decisions were made except for those generating each AV.

No. 2 (in Figure 5.4) is generated by the first sample of a '2'; the resulting AV is a mixture between a Type A and a Type C. This is to be expected, since No. 1, which is a Type B, is generated on the seventh sample. The four 'correct but not averaged' members of AV No. 2 consist of extreme samples of Type C, and one sample of Type B (given before No. 1 was generated), which did not receive a similarity of greater than 0.75. No. 3 is generated on the 20th input sample from a Type C sample. This AV was then averaged with a Type A to produce a 'mixture' of the two types. With further samples, it is anticipated that this AV would bear more resemblance to Type C samples.

Other Tests (e.g. Nos. 1, 2, 3, 14, 2; Table 8.13, pp. 177, 178) show the ability of the programs to construct effective AVS efficiently.

5.4. Summary

This Chapter has described a series of programs which can, given adequate training, recognize certain sequences of numbers representing hand-drawn lines. Examples of the various distortions the programs can handle (including orientation, local stretches and squeezes, size) have been presented. These results indicate that the method is promising for certain applications. Modifications required for some applications are discussed in the next Chapter, 6.
### NUMBER OF SAMPLES IN EACH TYPE OF DECISION

<table>
<thead>
<tr>
<th>Subset:</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>A: AVERAGED</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number:</td>
<td>7</td>
<td>8</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>4</td>
<td>2</td>
<td>0</td>
</tr>
</tbody>
</table>

(I& A includes AV-generating members).

**AV No. 1:**

<table>
<thead>
<tr>
<th>C1</th>
<th>C2</th>
<th>C3</th>
</tr>
</thead>
<tbody>
<tr>
<td>VC</td>
<td>-59</td>
<td>153</td>
</tr>
<tr>
<td></td>
<td>-42</td>
<td></td>
</tr>
<tr>
<td></td>
<td>-53</td>
<td></td>
</tr>
<tr>
<td></td>
<td>-27</td>
<td></td>
</tr>
<tr>
<td>AT</td>
<td>171</td>
<td>-10</td>
</tr>
<tr>
<td>TP</td>
<td>4</td>
<td>1</td>
</tr>
</tbody>
</table>

**AV No. 2:**

<table>
<thead>
<tr>
<th>C1</th>
<th>C2</th>
<th>C3</th>
<th>C4</th>
</tr>
</thead>
<tbody>
<tr>
<td>VC</td>
<td>-56</td>
<td>9</td>
<td>113</td>
</tr>
<tr>
<td></td>
<td>-63</td>
<td>37</td>
<td></td>
</tr>
<tr>
<td></td>
<td>-24</td>
<td>24</td>
<td></td>
</tr>
<tr>
<td>AT</td>
<td>162</td>
<td>-31</td>
<td>15</td>
</tr>
<tr>
<td>TP</td>
<td>4</td>
<td>3</td>
<td>1</td>
</tr>
</tbody>
</table>

**AV No. 3:**

<table>
<thead>
<tr>
<th>C1</th>
<th>C2</th>
<th>C3</th>
</tr>
</thead>
<tbody>
<tr>
<td>VC</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>-50</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>-85</td>
<td>-2</td>
</tr>
<tr>
<td>AT</td>
<td>155</td>
<td>20</td>
</tr>
<tr>
<td>TP</td>
<td>4</td>
<td>5</td>
</tr>
</tbody>
</table>

C1, C2, C3, C4: CURVE NUMBERS.

**TP:** TYPE OF CURVE.

**VC:** SLOPE DIFFERENCE VALUES FOR EACH CURVE.

**AT:** ORIENTATION OF FIRST LINE SEGMENT IN THE CURVE.

---

**TABLE 5.14. DECISIONS AND SUBSETS (AN6 ON TEST 13).**
Chapter 6

DISCUSSION OF THE PROGRAMS OF CHAPTER 5

In this Chapter, the capability and limitations of the programs of the first series (in which inputs are treated as representing line drawings) are discussed. Special reference is devoted to:

(a) the possible extensions and modifications to the programs, (Section 6.1),
(b) their application to an efficient (practical) recognition device, (Section 6.2), and
(c) suggestions for presenting to a user the details of the recognition of similarity between figures, (Section 6.3).

6.1. General Remarks

The programs operate on each input element as it is received. As explained, the operations involve a synthesis - lower level elements are grouped into higher level ones. The approach used in the programs has been compared to a certain 'cognitive' bug (Section 4.2) which attempts to remember the input sequence. Some problems have been found with this approach:

(a) the input can be 'lost' in the matching to an AV, (e.g. changes in sign of the slope differences may purge an hypothesis),
(b) the input line segments can be 'matched' to incorrect parts of an AV, (e.g. small changes in slope difference can cause (incorrect) matching of an input segment to a following curve in the AV).

It is possible that more sophisticated procedures can be found to overcome these problems. For example, more hypotheses could be generated to suggest possible
variations in the input. When a single slope value suggests that a better match to the succeeding curve in the AV can be made, two hypotheses (one suggesting the change; the other suggesting some 'extra' segments) could be generated. However, if many variations occur in the input, the problem arises of testing a large number of hypotheses. Some means for reducing the growth of hypotheses is required.

Improvements in the amount of storage and processing can be incorporated by a type of 'secondary' recognition scheme which operates on the various class models. This scheme would recognize similarities and differences between the AVS; for example:

(i) the similar (first) curve of the '2' and '3' or the loop of the 'e' and 'c',

(ii) the difference between two subsets - alternatives to a curve type in a certain position (e.g. 2, 2)

In these cases (especially (ii)), the common parts of the AVS need be stored only once - each AV having a common part would refer to the same storage for that part. Such information can lead to increased speed of processing - for example, realizing that the input has an initial anti-clockwise direction, both hypotheses for '2' and '3' can be purged together. Of importance is the fact that only once does the search for line segments from both classes have to be made. This would represent a significant time saved in the present scheme, as a considerable amount of time is spent in generating a model of the input for each hypothesis. In the second example above, (ii), a further saving in processing time might result. The program could simply suggest the most likely form of the AV initially (e.g. 2). If this was not confirmed to a high degree, then other alternatives for that class could be
suggested (e.g. $2, 2$).

This type of 'associational' technique can be partly implemented in the programs by a process of first quantizing the slope values and then equating corresponding parts in the considered AVS. Given suitable quantization (at a level that does not increase the error rate significantly, (Section 5.2.5)), some similarities between curves, as mentioned above, ((i) and (ii)) could be discovered as identities.

There are alternative ways of processing the sequences (other than by a 'pure' synthesis), which may be worth considering for certain applications. The on-line processing could be sacrificed in order to perform some operations - kinds of normalization - on the whole input. For instance, the average size and orientation could be initially calculated for a given sequence. The recognition of the input could then proceed as a 'pure' synthesis - however, with this revision, the prediction of patterns can be adjusted to the known major variations in the input. This could prevent some of the errors ((a) and (b)) mentioned above. The detail to which these pre-normalization procedures need to be implemented, depends on the environment - as shown by the programs, such procedures are unnecessary if there is little variation in the input. On the other hand, it may be worthwhile obtaining a considerable amount of information for example, including the beginning-end vector and the general trend in slope difference. This latter feature, discovered from the overall trend of the numbers in the sequence, (e.g. 4-3-2-4), could be used to purge certain hypotheses 'a priori', or to 'ignore' small changes in slope difference which may have been regarded important without the knowledge of this feature.
Another alternative to the 'pure synthesis' is a kind of 'level processing'. In this method, all of the elements on one level (e.g. line segments) are extracted together, for the entire input sequence. The formation of higher level categories (e.g. curves) then follows. This 'extraction' may take one of two forms. In the first method, the input is fitted with specified elements, as Spinrad (193) has carried out with line segments in a matrix representation of the input figure. His procedure is effective in removing noise, such as odd spots or holes. In the second method, the input elements are joined to form higher level categories - this combination proceeds for each level in turn. (In the present work, however, the combination has proceeded as each element is received). Marill et.al. (134) have used this second method to form line segments (belonging to different figures) from a set of unordered (X,Y) coordinates. Such a scheme could be implemented for the author's line drawings - all the curves (and then figures) could be formed from the input line segments. The choice of which of the above two methods is employed in a given situation, appears to be governed by the number of possible 'hypothesis' (or category) generations that have to be made for each input.

The similarity (or confirmation) measure was initially conceived as being derived from only the information from the agreement between the patterns of the input and the AV. Thus the measure depended upon the match between:

(a) the curves,
(b) the slope differences (and length) for each curve,
(c) the orientation values for each line segment in the curves.

It was the problem of measuring (b) which proved difficult, and another way to define a measure for this factor was
devised. However, the author still believes it is possible that a satisfactory measure could be defined on (a), (b) and (c).

The confirmation measure used appears reasonable for the types of figures considered in the Tests, but might not be satisfactory for other kinds of input. As has been mentioned in Section 5.2.4, the size and orientation values (for a sequence) are 'approximations' and give poor values for certain inputs. The definition of such factors is itself not an easy matter - for example, the size difference between the following,

\[ \begin{align*} &D \quad D \\ &2 \quad 2 \end{align*} \]

is difficult to define, and to calculate an 'average' orientation for the whole figure (in which one curve is orientated), for example,

\[ \begin{align*} &2 \quad 2 \end{align*} \]

is perhaps unmeaningful. It would appear that the notion of 'stretching' and 'squeezing' of a figure, and the relationship between the curves to the whole figure, must be defined in conjunction with the above factors. Such definitions are a matter for further research.

It has been pointed out (Section 5.2.4) that the confirmation measures may be required to extract a wider variety of relationships than are currently extracted by the programs, in order to form a basis for decision of input class membership. The discovery of the important relationships required for a given set of figures must be performed by the designer. These relationships must
then be incorporated into a measure which can be calculated efficiently.

The need for optimizing the way in which the various measures are combined to form the total confirmation measure, has been mentioned in Section 5.3.3.3. It is possible that some form of adaptive program could be written which would optimize the measures based on the results of a sample of inputs. It is expected that this adaptation would require a longer training sequence than for the construction of the AVS, and would be performed after the construction. The adaptive process could be made more effective if information about 'correct' similarity values (between 0 and 1) is provided for the programs. This would enable the program to alter more parameters of the measures (e.g. weights for each measure) or even to change measures. These 'correct' values could be obtained by questioning human subjects; for example, by asking subjects for a similarity value, (between presented figures), which would be their best (rational) 'bet' - a method suggested by Carnap (32), for determining inductive probabilities.

One of the levels in the model involves curves, of which there are five basic types for the programs. These types appear to be meaningful and satisfactory descriptors (and variables) for the inputs considered. The curve types produced by the CVE routine (Section 5.2.6) are actually approximate forms of the 'ideal' curves required. As explained in Section 5.2.6, the necessary length for a set of segments to form a curve has been defined in terms of the number of segments (which are themselves not constant in length). A maximum tolerance of $24^\circ$ has been specified for deviations in straight lines - hence a
sequence of slope differences at 20° is considered a straight line, whereas it is actually a curve of 'large' radius. The notions of the total curvature and length of the curve are obviously required to be more specifically defined in the program to generate more acceptable curve types.

Other forms of curves may be desirable in different environments. The following set of figures, for instance,

\[2\ 3\ 4\ 5\]

is described more efficiently by straight lines and right-angles; sloppy handwriting sometimes contains few straight lines, and may be best described by curves (Types 3 and 4) and sharp corners. This definition of curve types is essentially a problem of specifying useful descriptors, to be determined by the designer for a particular class of input.

This suggestion of specifying descriptors, does not mean to say that a machine could not discover useful descriptors from the patterns in the environment. On the segment level, a means is required to detect recurrent sets of elements in the sequences, while on the curve level, the method must detect the recurrence of similar sets of slope differences in the figures. This generation could be performed by general rules (similar to R0, R1 and R2, see Section 8.3), to form such patterns on each level. Alternatively, the program could initially consider the whole sequence as a line segment and continue to subdivide the sequence according to a known criterion, when incorrect decisions are made. By successive subdivision into smaller segments which are common in the environment (i.e. included in
other figures), curve types and segments could possibly be formed.

The comparison rules, which specify the action to be taken by the program on the match of the input elements (on the various levels) to each AV, are dependent on the curve types. Therefore, if the curve types are changed, these rules would require modification.

Further modifications could be made to the rules to increase the efficiency of the recognition scheme. The limiting of possible input class membership can be controlled by more stringent rules. For example, if a sharp corner is expected, then, under the present scheme, any number of segments can be included in a curve to confirm the corner. In many cases, the hypothesis should be purged, (see Section 5.3.3.2), rather than allow it to 'stay with the input'. This purging could be performed on the value of the similarity measure (which decreases in such situations) but it could also be performed by restricting the allowable number of matches of segments in a corner. For instance, the appearance in the input of more than three additional segments at the end of a curve (increasing the total curvature of the curve) could cause the hypothesis to be purged.

It would appear that some further tolerance on the PAS-FUT range is required to accommodate local fluctuations in slope difference values. As mentioned in Section 5.3.3.2, a human subject is tolerant to certain changes, whereas the program would purge the (correct) hypothesis. Processing enabling the program to allow small variations (which are followed by a return to the underlying curve) is required. The need for the machine to 'know' the overall shape and size of the figure (i.e. a 'wholistic' view),
seems necessary before the notion of 'local' can be specified—this suggests another advantage for an initial phase of processing to extract overall features of the figures (mentioned above in this Section).

An important feature of the programs is their ability to construct a set of class models from given samples. This type of learning has been incorporated in other recognition schemes (e.g. 200). In the particular series of programs written by the author, the learning process overcomes some of the lack of generality in performance. This lack is shown, for example, in the requirement that the input is an encoding of each figure traced in a consistent manner, and the inability of the programs to cope with extraneous segments. The programs do have the ability, however, to accommodate to various styles of writing. In an alternative approach to recognition, (e.g. 80), the designer must specify powerful features capable of characterizing a wide range of (known) distorted figures. The use of a particular approach depends primarily on the purpose of the recognition device.

The programs developed in this project have always been provided with learning samples which contain the correct class name of the input figure. Some experiments performed with unsupervised learning are possible. For example, the decision to form a new class could be based on the total confirmation measure, e.g., 'if the maximum total confirmation of the input to any AV is less than a specified value, a new class is formed'. While the ability to form subsets with the same name in the same class appears impossible in such a scheme incorporating limited information, the nature of the learning may be interesting to study.

Although the programs are themselves problem solvers
(of a kind), they could form part of a more general problem solving system. The pattern recognition part could construct a model (of the environment) during the system's interaction with the environment. The machine could make decisions based on similarities recognized in the input. The resemblance, in this respect, to "BOGART" in the task of game playing has been mentioned, (Section 2.3.1). Another example of interest is the control of the scanning device producing the input sequence. A machine which could direct the scanning device to search certain areas, and along specific lines from intersections, is a possible extension, (mentioned further in Section 6.3).

6.2. Recognition Applications

The programs appear promising for use on a graphic display in an interactive system. A typical situation might involve the recognition of an alphanumerical character set or certain line-drawn characters (e.g. map lines and mathematical symbols).

Advantage can be taken of the 'on-line' processing of the data. That is, part of the synthesis could be performed while the display receives the input sequence. The coding would have to contain distance information for multi-stroke figures.

Processing such figures would require a simple modification to the program - the characteristics of the position of the additional segments can be expressed in terms of a beginning-end vector. A user normally maintains the same tracing sequence for drawing characters, so that processing for other possible sequence encodings is not necessary. There exist standard procedures (21) for determining the presence of multi-stroke figures and confirmation of the position of the strokes could be
determined by the measures at present incorporated in AN6.

The advantage of learning in the installed recognition program would allow the user to specify his own character set. During the learning phase, it would be possible to exhibit the various AVS with their curve types, on the display. The user could then erase those AVS with irregularities or poor descriptions. The number of samples required for the AV construction (not counting erasings) would be about 3 or 4. It is further desirable to have the AVS recorded on more permanent store (e.g. on cards or disc) so that they can be read in as data when the user requires the display later. This input is required for AN5, for which each AV is specified by about 20 values.

As previously mentioned, the main objection to the scheme would be the time taken to process an input, but there are some ways in which this might be reduced. The use of 3-element segments appears satisfactory (see Appendix D) and can be readily extracted by a graphic device. The incorporation of some associational techniques for reducing the storage and processing time has been discussed, (in Section 6.1). The numerical calculations could be revised and improved upon to reduce their speed of execution. It is anticipated that the whole program could be written in an assembly language to decrease the execution time further.

As mentioned in Section 5.2.2, an 'on-line' experiment has been performed by the author, with a modification of program AN3. This experiment furthers support for the ability of a program, (operating in a similar way to the programs discussed in Chapter 5), to be useful as part of a graphic system, (see Appendix D).

Handwriting could be recognized (on-line) by the programs if suitable segmentation techniques could be
incorporated. Note that the recognition of extraneous segments is a special case of the segmentation problem. The obvious way to segment the characters using the programs presented here is to test hypotheses for the occurrence of a certain figure at various positions in the sequence. These positions could be restricted to situations where a curve change appears. Knowledge of the context of letters could reduce the number of hypotheses generated. However, the processing involved is expected to be too time-consuming for practical purposes. This project has suggested a more efficient approach to the problem, (see Appendix A).

The programs might form part of a reading machine, but many problems would have to be overcome. (The reading problem is essentially different from on-line recognition. Two main problems - the extraction of continuous thin lines and making the scanning consistent in direction of tracing - have to be overcome). The types of figures acceptable to the device are those which could be recognized by their shape and whose shape could be expressed conveniently in terms of the curves extracted by the program. Examples are handprinted characters (which can be reduced to line drawings), and contours of certain objects, (e.g. biological cells).

A curve follower is the obvious choice of device for extracting the input code. However, there are types of inputs, such as those with holes and breaks in the line, which present difficulties to a follower. An alternative is to scan the input and obtain a matrix representation. Operations such as hole filling can then be executed on the matrix by local operators, (23). Extraction of line segments could then follow by well-known methods, (157).
The direction code as used in this Thesis is inherently limited in the information it can give about the distance between unconnected points. Thus, multi-stroke figures and the presence of many figures in an input require the addition of some convention about the distances between the various lines. As mentioned above, the programs can handle these separations by recording them as a beginning-end vector.

Unfortunately, a reading machine will not always scan figures of the same class in the same way, especially if the figures are oriented. Further processing is required to ensure correct correspondence between the input strokes and the stored parts. In particular, the direction of line tracing and the end-points which form the vector specifying line separations, must correspond. One method would be to have the coding of the lines controlled by the stored expectations so that at each stage, the input coding is the 'best match' to the AV. However, this would not be satisfactory if suggestions (possibly conflicting) were made from different AVS. An alternative is to store all the possible codings of the character in the machine. A particular input coding could then be compared to the stored values for the 'best match'. In either case, the amount of processing required is considerable, and would increase for more strokes in the input figures. The procedure incorporated by Grimsdale et al. (79) is similar to the latter method - it is somewhat restricted by having the input coding independent of what is stored in the AVS. Other programs have attempted to avoid the search procedure by segmenting the parts of a figure in a prescribed way, (e.g. Spinrad (193), Minneman (140)).

A similar search procedure is necessary in situations where many figures may occur in the input. The problem is
more difficult if figures happen to overlap - a search procedure must be incorporated, (in this case), for the figures rather than strokes. Some alternative approaches have been devised to distinguish well-formed objects, such as cells, when some of them may be overlapping in the input, (e.g. 164).

There are some figures which can be made 'stick-like' but which would appear to require different descriptions in the model. Examples are Chinese characters, fingerprints and chromosomes. The program could be changed to incorporate a model for these inputs and to provide an appropriate similarity measure. However, the problem of search once again arises, and in Chinese characters and fingerprints especially, where the number of line segments is large, other methods (33,81,220) appear better suited for practical purposes. It is felt by the author, that the programs in this Thesis are limited to simpler shapes, because of the search procedure involved.

An added feature of the input data is that information about the input figure can be extracted. Freeman (65) has shown, for example, how the beginning-end vector, the moments and centroid of a figure, and the area of a closed figure, can be obtained from the sequences.

6.3. Use in Interpretive Situations

Perhaps the most important feature of the programs is their ability to construct a detailed model of the input figures. The description of an input could be given in terms of its constituent line segments or curves. This latter level is considered to be useful in general discourse about line drawings.

Various relations between two figures (in particular, the input and the AV) are extracted in the program, by
virtue of the comparison of an input to the stored model. The dissimilarity between figures (often caused by differences in the sign of the slope changes) is noticed by the purging of an hypothesis. If this is not the case, then the figures are said to be similar and a numerical value for the similarity, determined from certain specified factors, is calculated. The factors include the size of the figure, and the orientation and beginning-end vector for each curve in the input. The shape of corresponding curves is compared, and stretches, squeezes and additional line segments are noted. Associational techniques can provide information about the similarity between various AVS. Additional processing could extract relationships which are not dependent on the direction of travel, (e.g. a '2' and a '5' both contain two straight lines and a curve). Another example, dependent on the expected orientation of the figures, is the difference between an 'M' and a 'W'.

An interesting development of the current program is a scheme which could 'talk about' pictorial information, in a readily understood manner, with a number of users. There seem to be two major problems:

(a) being able to extract the necessary information symbolically, in a form which can be readily manipulated,

(b) providing a language for the output of (a), such as a subset of English.

and it is the first of these that will be discussed here.

A simple extension to the programs is the presentation by the machine of the reasons why a particular decision of input class membership is made. This would entail a 'conversion' of the relationships found by the similarity measure, into meaningful statements, and
outputting the associated numerical values. For example, the machine could note that 'the class 'M' received the decision from the 'W' because of the large difference in the values for the orientation measures, namely, 0.8 to 0.2 (averaged for the figure). At a more sophisticated level, a statement such as 'the '5' received the decision (from the 'S') because of the presence of sharp corners in the input', could be formed.

Further extensions to the machine could be made to provide more information (mentioned earlier in this Section), gained from associational processing. Thus the machine could discover similarities between given figures on a display (e.g. \(2, 2\)), or even more complex drawings, such as,

\[
\begin{array}{c}
\Delta \\
\circ \\
\end{array} \quad \text{and} \quad \begin{array}{c}
\circ \\
\Delta \\
\end{array}
\]

This latter problem has been tackled in a specific manner by Evans, (54).

Instead of 'talking about' similarities between figures, a simpler problem for the machine (although the simplicity is not obvious) is that of 'talking about' patterns in a drawing, e.g.

\[
\begin{array}{cc}
\times & \times \\
\times & \times \\
\end{array}
\]

The input to the machine would be a collection of vectors and the aim of the machine would be to form patterns in combinations of these vectors - e.g. the four crosses and the set of vectors delineating four rectangular regions in the above diagram. In effect, this problem (similar to the search for patterns in the sequences discussed in this
project, (in Chapter 7), includes the relationship of spatial (2-dimensional) information. It is a major problem in Artificial Intelligence for a machine to grasp the notion of such patterns, (42).

Some rearrangement of the data structure of the information is anticipated for the above scheme. In the first instance, the various relations between the stored parts of an AV could be made explicit by incorporating a list-structure representation of the model. Note that the hierarchical structure of the model lends itself to this representation. Evans (54) has shown how specifying and manipulating descriptions of line drawings can be performed in the LISP language. Gray (75) has reviewed types of ring structures (a special type of list) which appear more suited to the specification of graphics. In the second instance, an account of absolute position of a figure and distance between figures on one display would need to be incorporated into the data-structure.

The problem suggested here is a simplification of the general problem of machine interpretation of pictorial data. There appears to have been little done in designing a system which can describe pictures in a general manner. Narasimhan (147) has devised "Bubble Talk", a language for describing various parts and relations in bubble chamber photographs. This is the type of scheme envisaged here with the numerics, where a user can interact with the system. Kirsch (106) has developed a program for testing the truth of an English statement about a given picture. Considerable interest is being given to this and similar problems such as data-structures for question-answering schemes, (182), the manipulation and generation of graphics (197), and languages for performing these
latter operations, (165).

The main problem for the scheme suggested here is to devise a method to operate on the models of the figures to form the description. The program would know the names of the relations; in effect, the designer's problem is to write a program to detect known relationships. This is a contrast to the program "Namer" of Londe and Simmons (127), which can learn a limited set of relations between pictorial figures from their occurrence in the input samples.

6.4. Features of the First Series of Programs

The above Sections have discussed various extensions to the work of Chapter 5. The following is a summary of the important features in the author's programs which represent improvements over current character recognition schemes when applied to the sequences considered: -

(a) The construction of a detailed 'wholistic' model of the environment. Each class is represented by an 'average member', which is the program's 'concept' of typical class members.

(b) The use of 'experience' to generate a description of the input. Each input is described in terms of patterns found in a particular figure; each input is related to every known class.

(c) The similarity between the input and the 'average' class member is extracted by numerical measures, defining certain factors such as orientation, size and shape, in the input description. It is these measures which give the programs their independence to orientation, size and a variety of stretching and squeezing. The relations between parts of the figure, as well as the parts themselves, are required to exist in an input member; (some of these relations...
and parts are required to exist in the input by virtue of the generation of the description, ((b) above)).

(d) The use of supervised learning to define the class names and average member for each class. The process of 'averaging' is incorporated to remove noise and irregularities from the class model. The number of learning samples per class is about 3 or 4.

The various programs have been given a number of Tests, involving various characters. Their ability to learn and to generalize to various distortions has been considered promising.

Two important applications of the programs have been discussed in this Chapter:-

(i) recognising line drawings (the classes of which are specified by the user) on a graphic display, as part of an interactive system.

(ii) describing line drawings and, in general, discovering similarities between line-drawn figures as a step to man-machine communication.

Several extensions to the programs, some of which are Artificial Intelligence problems in their own right, have been outlined. While the sequences discussed in this project are a convenient vehicle for many problems, they possess few relationships, and the approach, originally designed for the sequences, appears in need of modification when more complex inputs are presented. This aspect is considered in Chapter 10.
Chapter 7

THE APPROACH TO THE SECOND SERIES OF PROGRAMS

This Chapter describes the second approach to programming a computer to recognize the sequences discussed in Chapter 3.

7.1. General Considerations

In the previous approach to programming (Chapter 4), the construction of a certain model was desired and the devising of abilities to perform that particular construction was the main problem. In the approach of this Chapter, the conditions are essentially reversed. The machine does not 'know' that the number elements represent particular figures or shapes - in fact, the elements are treated as being abstract entities. Consequently, the description of the sequences will be in terms of internal relations - in particular, those of identity and succession. The aim of the approach to this series of programs is to produce an adequate model, employing the ability to detect these relations, in order to recognize the classes of sequences.

Naturally there was some initial notion that an admissible solution to the problem could be obtained. Some of the support came from noticing that a person could detect differences between sequences belonging to different classes. This fact suggested the study of a concept learning task with human subjects. In particular, the aims of the experiment were to:

(a) study the features or descriptors (reported by the subject as having been used in the discrimination).
(b) examine the performance of the subject in terms of
The types of error and the number of errors made.

The particular experiment chosen is described:

Each sequence is typed on a separate card. A subject (S) is initially presented with two sequences, which, he is told, are typical members of a class A, and a class B, respectively. Further sequences are presented to the S, and after each one is given, the S must decide (with no time limit), to which class (A or B) the sequence belongs. The initial two sequences remain in the S's view throughout the experiment, but the others are covered by the successive cards. The decision of the S is immediately validated, and another sequence is presented.

This experiment was performed by the author using four different Tests, (Nos. 1,2,3,14,2: see Table 3,2) on five to ten subjects (University undergraduate and postgraduate students) per test. There was evidence of an improvement in performance from all Ss, with more samples, and most Ss could be said to have grasped the concept satisfactorily. Some Ss were vague about the class differences but knew the general region in the sequences which were important for discrimination.

There are many modifications, (such as not keeping any previously presented sequence in the Ss view) which could be made to this experiment, but these were not attempted. The experiment can be considered as a special case of a task in which an organism must come to know the environment. For the particular experimental conditions outlined, the S treats each sequence in turn - an extension to the notion that the organism accepts sequentially 'small' elements of the environment (Section 2.1). This difference, however, is of little consequence, because the theory of model construction can nevertheless apply to a machine which processes successive input sequences.

The approach adopted when developing the programs, was to use the above 'psychological experiment' as a vehicle.

"Validate" as used herein implies 'confirm or disconfirm'.

*
for discovering a suitable method to incorporate in the programs. The experiment was considered to be of interest because the types of patterns obtained by the S, are based on internal relations (as for the machine).

The author has, in fact, derived a great deal of help from the experiments. This is not to say that the programs are a simulation or an "optimal model" (101) for the concept learning task, although it is suggested in Chapter 8 that the programs developed could be extended in this direction. The main desire, however, has been to produce an efficient recognizer. Reference to relevant human performance will be made as a matter of interest.

The sequences present an environment on which, as far as the author is aware, no concept learning studies have been made. The environment is more complex than those usually considered in such studies; other differences will be mentioned in the following Sections. Studies such as perception of regularities by human beings (and machines) are related to the author's program and brief references will be made to some of this work.

7.2. The Model of the Sequences

The problem for the machine - to recognize the sequences - has been framed as one of model construction. That is, the machine must build a coded representation of each class of sequences, which will allow it to discriminate effectively the class members. This ability implies that those features of each class which are characteristic, can be contained in the coded representation.

Some economical descriptors for the sequences are the following patterns:
Type 1 (T1): 1 element recurring e.g. -22-
Type 2 (T2): more than 1 element recurring e.g. -3434-
Type 3 (T3): elements recurring not consecutively e.g. -34-34-
Type 4 (T4): elements recurring in different sequences e.g. -34- / -34-

The distinction between T1 and T2 patterns has been made to facilitate later discussion. It is suggested (as explained in Section 2.1.3), that an efficient concept learner (either a human being or a machine) operating on the sequences, should concentrate on describing the inputs in terms of some of the above patterns - the types employed will depend on the level of redundancy in the environment. The patterns used in the models (for the programs discussed here), are ordered, because members of the same class can be expected to have similar patterns in the same order of occurrence. Various models have been incorporated, and each model will be discussed in turn, (Chapter 8).

The above types of patterns correspond to certain parts of the pictorial representation: T1 and T2 are line segments; T3 represents a repetition of the same line in another part of the figure; T4 represents the same line in different figures.

It is very difficult to discover with what patterns the S (in the 'psychological experiment') describes the sequences. The descriptions offered by the S are usually quite vague; for example, "There were a lot of 3's in the middle" and "The class usually commenced with a couple of 6I patterns". However, it would appear that the suggested pattern types, particularly T1 and T2, are among those commonly suggested by different Ss. Others were employed, such as the general trend between numbers in the sequence (e.g. 4-3-2-4), especially in Test 14.2 (Table 8.14, p.181)
but these in general, give poor representations of the sequences.

Other possibilities for representing the sequences have been briefly considered by the author. The storing of n-grams is not suitable because of their lack of order. Incorporation of the n-grams into a model (similar to Foulkes (63)) is expected to give a poor description, because it is doubted that the sequences (for each class) are ergodic. Furthermore, the generation of the model would be inefficient (compared to the generation of the author's coded model). This latter reason is also expected for finite-state machines, as employed by Fogel (61). These suggested models (other than the author's) while being representative of the sequences, do not emphasize their significant features. Much economy in storage and processing is achieved in the author's programs, by concentrating on the coding aspect.

The form of the suggested model is equivalent to a set of context dependent phrase structure rules, (36). The pattern 'B' (= '34'), for example, may be defined in terms of its elements and the preceding and succeeding patterns, viz. $34 \leftrightarrow B/A-C$, (Chomsky's notation (37)). However, it will become apparent that the approach in the programs developed in this project is not 'syntactic'.

7.3 Model Generation

The following explains how the programs (of this series) can construct the model, discussed in the preceding Section.

If no experience is recorded, then the (particular) program must discover patterns in a given sequence. A simple rule:
RO: 'Find sets of elements which recur consecutively', can generate T1 and T2 patterns. The generation, however, is not unique, as shown by considering the sequence, '3334334' from which the patterns '3', '33' and '334' could be formed. A criterion to select one of the possibilities must be incorporated in the program; for example,

CO: 'Choose that pattern which has the greatest number of elements, and which appears first in the sequence'.

Once a pattern is found, the controlling rule can be changed to:

R1: 'Find the generated description in later parts of the sequence'.

Such a rule detects patterns which have been found previously (as T1 or T2) in the sequence. There is no attempt in any of the developed programs to discover T3 patterns (although those generated by R1 are a kind of T3 pattern). Furthermore, R1 is employed only on that part of the sequence between patterns generated by RO.

Patterns found in the initial sequence of a class (known by subsequent validation), become the descriptors of that class. This model is then used to describe the following inputs, by incorporating the rule,

R2: 'Find those patterns (preserving the order in which they were discovered) which have occurred previously'.

The decision of class membership is made on the basis of the ability to describe an input with each class model.

A correspondence between the above rules RO, R1 and R2, and those for the first series of programs (Section 4.2) can be noticed. A difference exists between them—the rules in this Section generate only those patterns which can be detected by identity and succession. Those of Section 4.2 group input elements into one of a number of general
forms (known by the program), although their form is influenced by these relations, (identity and succession).

The ability to transform the parts of the input is incorporated in the first series of programs. There exists no such operation in those of the second series, because their abilities have been limited. In particular, the relationship of arithmetic order in the number elements cannot be extracted by the programs, thus preventing the generation of 'curves' within the model.

Concept learning experiments are usually not concerned with the abstraction of features by the S. The features in the given items are usually well-defined, (25) - examples are the colour and size of rectangular regions. The features of the author's sequences, while not presenting any problem to the S (in the 'psychological experiment') to describe, are not as well-defined as for the features of the concept learning tests. It is possible to extract many properties (from the sequences) which describe the sequences. This Section has suggested rules by which a machine could discover certain patterns.

Several concept learning programs for simulating human behaviour have been reported in the literature. A common problem is the discovery of a conceptual rule in a series of items, containing five binary-valued elements, (e.g. '01100', '00101' - - -). The machine (or human S) is told which of the items, (presented sequentially), entail the concept (e.g. a binary '1' in the third position) and which do not. This problem is the subject of many programs. Hunt (97,99) has employed the generation of decision trees (by noting similarities or differences between elements of the same class), as the basic technique for discovering the conceptual rule. Many variants of
the main theme, such as limiting the memory of the programs, have been performed, and several applications (e.g. classification of medical symptoms) have been found for the basic method, (99). Kochen (110,111,112) has incorporated procedures for changing the current hypothesis (or hypotheses) of the concept, based on information gained from each sample. Johnson (101) has developed a simulation program, (for the above binary element problem) exhibiting 'behaviour' which is sometimes 'optimum' (as in Kochen's program), and sometimes is what one might expect from a particular human subject. Much data has been collected on the various patterns discovered by human subjects. Baker (13) has incorporated the 'conservative focussing' method, mentioned by Bruner et al, in his simulation.

Gregg (78) has reported a program to simulate a human subject on a variant of the binary sequence problem. A certain pattern exists between the successive sets of items presented; for example, the series may present the ordered set of numbers, 0 to 15, expressed in binary notation. Much consideration is given to the types of patterns noticed by the subject. The problem is actually posed as one in which switches have to be set manually (by the subject); Gregg incorporates in his program, patterns in the manual changing of the switches, as well as patterns of a conceptual nature. Hypotheses are formed by the program by testing for a match between the specified patterns (and combinations of them) and the input series.

Some interest has been shown in areas, related to concept learning, which examine human ability to discover patterns in sequences. Some of the experiments have been concerned with:

(a) detecting recurrent patterns in sequences of single
elements, with various types and amounts of noise in the patterns, (e.g. 31).

(b) determining the degree of complexity of structure (or randomness) in a given set of sequences, (e.g. 180).

The rules R0, R1, R2, are effectively means for the discovery of simple relations in a body of data. This problem is one of induction, (Solomonoff (188), (189)) and many attempts have been made to mechanize the process. The programs of Fogel et al. (62) and Foulkes (63) described in Section 2.2.1, are cases in point.

The ability to extrapolate correctly a given sequence, is a method of testing if an S has the 'concept' embodied in the presented sequence. Simon and Kotovsky (183) have simulated the human behaviour of finding the periodicity and patterns of changes in alphabetic order, in sequences of letters. Similar rules to those suggested above by the author, appear to be incorporated in their program. Pivar and Finkelstein (159) have programmed a more powerful set of subroutines (than have Simon and Kotovsky), which can accept a wider variety of sequences, containing letters or numbers, for this extrapolation problem. Operators work on the given elements to discover (in some cases, to approximate to) patterns. This particular attempt is more concerned with a mathematical model for generating regularities than a simulation of human behaviour.

Another prediction problem is that of 'guessing' the next element in a binary-choice sequence. Feldman (58) incorporates various sets of patterns, containing different proportions of the binary values. The program, an attempt to simulate human behaviour, alternates between the sets of patterns in order to provide a prediction of the sequence.
The 'analogy' problem - namely, if A is to B, then C is to either one of D, E, --- ' where A, B --- are each a certain kind of item - is another type of inductive task, requiring regularities to be found. Two attempts - Evans (54) on complex line drawings and Reitman (163) on words (and hence involving semantics) - have been reported in the literature, but neither method is specifically devoted to an 'inductive generation' of regularities. For example, Evans simply employs a specified search for defined correspondences in LISP language descriptions of the drawings.

As has been mentioned (Section 2.2,2), a theory or concept is a kind of regularity. Amarel (7) has been interested in designing a machine which will devise 'programs' to calculate a (logical) transformation (e.g. the greater lower bound) on a fixed set of values. The transformations are learnt by the machine from a set of training examples. Some of the ideas involved appear quite interesting - in particular, the need for a hierarchy of languages for the functioning of a machine. Solomonoff (186,187) has discussed problems involved in the automatic formation of a program to perform arithmetic operations, from a set of sample calculations.

Essentially the above work (dealing with pattern generation), differs from the author's approach in the nature of the regularities, and the method by which these regularities are extracted. Most programs cited above have been primarily concerned with the generation of certain patterns, meaningful to the given problem - in the present work, the generation is employed as a step towards recognition.
7.4. The Decision

To decide the class membership of the input, a measure of similarity between the 'average member' represented by the class model, and the described input, is computed. In the first series of programs, (Chapter 5), this measure depends on various 'external' properties of the input's description. In the present series, the programs are restricted to measuring internal properties of the match between the input and the stored class model.

An obvious suggestion (the one used by Fogel et.al. (62) and Foulkes (63)) is to base the measure on the match between the element predictions from each class model and the input elements. This suggestion was not employed here for several reasons. In the first place, the confirmation of predictions on the element level is not in correspondence with the angular variations between given sets of elements. For example, the similarity of '3333' and '3334' to '3332' is the same,* but their respective angular variations are not equal. Furthermore, the similarity of '3323' to the input '3332' is less than the first two mentioned, but vectorially '3332' is equivalent to '3323'. Thus a confirmation measure based on the success of element prediction would be a poor measure of shape.

Another reason why the above suggestion is poor for the sequences of this project, involves the problem of generating a prediction for every input element. In order to cope with all the sequence variations, the prediction would have to be dependent on the current state of the input. Random predictions would be necessary in cases where the current element is not a member of any known pattern. Furthermore, the decisions as to when to commence predicting with the next stored pattern, are not

*Given the '3', their similarity value is 2/3.
at all obvious.

A more natural approach to the confirmation measure follows from the discussion of Chapter 2, and involves the coding of the input with the stored patterns. The measure is, accordingly, some function of the reduction in coding of the input for each class model. Thus, given one class with the patterns, A='43', B='22' and another with patterns C='44' and D='32', for an input of 4434332222, the respective codings are:

- class 1: 4AA3BB
- class 2: C343D222

A measure, such as the length of the coded sequences, can now be employed in a comparative manner to decide the 'best' coding and hence the input class membership, (class 1 in the example).

The criterion, therefore, assesses whether the patterns in each class model are typical of those found in the members of that class. The best (ideal) coding would make each symbol in the final (coded) sequence equi-probable - that is, the final sequence would be random. As explained in Section 2.1.3, this sequence, although random (in an objective sense), becomes more 'comprehensible' because the inserted patterns are known, and the amount of storage for the input is reduced.

This coding method does suffer from the requirement of an 'exact match' of the stored patterns to the input. Thus a pattern of '443' cannot code a set of input elements -434 - even though these two sets are vectorially equivalent. However, the correspondence of the measure to the angular variations in sets of elements does not arise, and the insertion of the codes can be easily programmed.

This suggested approach is quite different from that taken in concept learning programs. The machine (or human s)
is faced with two main problems in a concept learning experiment, (97):-

(a) to discover the relevant features,
(b) to discover the conceptual rule relating these features.

In computer simulations, the first problem, (a), is usually achieved by a focussing on the agreements or differences between the various features in the sequence of items (99,101,110). In the author's approach, the discovery of relevant attributes is implicit in the construction of the class model. Discrimination between the classes depends on the differences between the discovered patterns in the class models. Given suitable training samples, the nature and degree of the differences should correspond with the characteristic features and their relative importance.

The conceptual rule in (b) above is well-defined in concept learning experiments, typically being a logical combination of the values of certain features or elements in the items. The sequences in this project do not, however, have a well-defined rule separating the classes. The correct assignment of an input sequence is determined by the pictorial similarity to the various classes of figures, shown by the figure which the sequence represents. For most tests discussed here, this decision of class membership is not difficult to make. However, the basis for the decision is not easily defined in terms of the number elements. The 'rule' usually involves the presence (or absence) of sets of numbers (expressible in the pattern types) in the sequence, (see Table 8.14; p.181).

The ill-defined nature of this 'rule' presents a problem for a machine to attempt to discriminate the sequences by delineating a rule concerning the sequence elements.
7.5 Learning

As mentioned in Section 7.3, the patterns found in the class members code the input to determine the input class membership. Each decision is validated by the outside teacher. The information obtained is used to define,

(a) the classes, and

(b) the patterns in each class model.

Each program has its own method for defining (b).

Because the approach is one involving model construction, the programs update the class model after every sample, independently of the success of their decision. On the other hand, in approaches incorporating 'hypothesis testing', the information from feedback is used to refine the hypothesis; this current hypothesis often remains unchanged if the decision is correct, and is changed in some manner if the decision is incorrect, (30, 97).

The author's programs can operate in the working phase by not incorporating the feedback loop, (see Figure 7.1). The effectiveness of the generated model can then be tested.

7.6 Summary

Figure 7.1 summarizes schematically the various functions described in this Chapter, to be incorporated in the programs. Code 2 refers to the implementation of the rule R2 (inserting patterns in the input) which uses the model (Model). If no experience is present, control passes to Code X whose patterns formed by the rule R0 are generated. When the inputs are coded by R2, the similarity measure is calculated (Det. Sim.) and the class membership of the input is based on this measure (Decision). The dotted line between Det. Sim. and Code 2 is a reminder that there is an optimization procedure (in each program) which selects
the best insertion of codes in the input. The decision is validated by the outside teacher (O.T.) and the information obtained is used to find further nodes for that correct class, (in Code X).

The overall approach to the program in the second series has been presented in the preceding sections. The following Chapter gives the details of the program, whose functions have been summarized in Figure 7.1, (discussed above).

Figure 7.1. Outline of Functions in Programs of the Second Series
the best insertion of codes in the input. The decision is validated by the outside teacher (O.T.) and the information obtained is used to find further codes for that correct class, (in Code X).

The overall approach to the programs in the second series has been presented in the preceding Sections. The following Chapter gives the details of the programs, whose functions have been summarized in Figure 7.1, (discussed above).

To help in explanation, worked examples are given for EC and EF. Considerable time was taken by the author in the development, mainly in order to:

(a) simplify the outlined theory for programming,
(b) modify various program parameters.
A discussion of (a) and (b) is given in Sections 5.3 and 5.5.

Results are shown in Table 5.11 (at the end of this Chapter) for EC, ED, ES and EF and for certain programs from the first series, (see Chapter 3). In Figure 8.4, the pictorial characters of Test 14.1 are presented (at end of Chapter). Table 8.14 gives the corresponding number of sequences for these characters; Numbers 1 and 2 are the given numbers for the 'psychological experiment', and the others follow in order. The characters of this Test 14.2 are more variable especially in non-characteristic regions, than those of Tests 1, 2 and 3.
Chapter 8

PROGRAMS IN THE SECOND SERIES - INPUTS
TREATED AS SEQUENCES OF ABSTRACT ELEMENTS

The programs which incorporate the theory outlined in Chapter 7, are now described. There are two main programs, designated EC and EE, and one modification to each of these, designated ED and EF respectively. EC operates on data presented in the same way as the 'psychological experiment' of Section 7.1. ED is a modified version to operate without the emphasis on the first two initial sequences. EE and EF are recognition programs, incorporating a model different from that used in EC.

To help in explanation, worked examples are given for EC and EE. Considerable time was taken by the author in the development, mainly in order to:

(a) simplify the outlined theory for programming,
(b) modify various program parameters,

A discussion of (a) and (b) is given in Sections 8.2 and 8.5.

Results are shown in Table 8.13 (at the end of this Chapter) for EC, ED, EE and EF and for certain programs from the first series, (see Chapter 5). In Figure 8.1, the pictorial characters of Test 14.2 are presented (see end of Chapter). Table 8.14 gives the corresponding number sequences for these characters; Numbers 1 and 2 are the given members for the 'psychological experiment', and the others follow in order. The characters of this Test 14.2 are more variable especially in non-characteristic regions, than those of Tests 1, 2 and 3.
8.1. Operation of the Program, EC.

8.1.1. Extraction of the Model from Initial Examples

A subject in the 'psychological experiment' is initially given one exemplar of each class to be discriminated. In the same way, the machine is given two sequences, which it is told are members of class 1 and class 2 respectively.

The patterns of type 2 (T2; Section 7.2) are extracted from each of the above exemplars. Because there are possibly many sets of patterns that can be extracted, (Section 7.3), a criterion based on a measure of information capacity is used. Each set of T2 patterns, codes (with letters replacing the pattern elements) the input sequence. That set which reduces the information capacity (ICAP) of the sequence the most, is retained, where:

\[
\text{ICAP} = \log_{10} \left( \frac{6 + N}{10} \right) \times \text{JS},
\]

where \( N = \text{number of patterns, and} \)
\( \text{JS} = \text{number of elements in the coded sequence}. \)

ICAP is an approximation to the information capacity because:

(a) not all 6 direction codes are present in many sequences,
(b) not all N letters are possible in any position because order of patterns is considered,
(c) some of the 6 direction codes are not possible in certain positions; in some cases, the presence of a given code would have led to the insertion of a symbol.

However, the effects for the given Tests are small, and because of the comparative use for the measure, the extra calculations were deemed unnecessary. Given a choice between patterns, the above criterion will choose the longer T2 pattern. An alternative measure (to ICAP) is the information content per symbol, but this appears to require more programming to calculate.
Because the two given members, (G1 and G2), remain in the view of the subject, it was decided (for this initial program) that the machine should remember details of the entire sequence. It did this in the coded (with T2 patterns) form. It seemed natural that since this information is recorded, T4 patterns (Section 7.2) could be extracted from the input and stored coded sequences.

Table 8.1 shows the given members for Test 1 together with the extracted T2 patterns and the coded sequence for each class. An alternate set of patterns is shown for class 1. Alternatives which have the same ICAP value are retained, (two are shown for class 2).

8.1.2. Coding the Input Sequences

The first operation on an input sequence is to code it with the stored sets of T2 patterns. For each pattern in a set, the following procedures are performed:

(a) the pattern is coded at the first position for which a match can be found in the input, and this pattern continues to code the sequence to the end.

(b) beginning from the last coding position of the patterns (found by (a)), the succeeding patterns in the set are, in turn, coded (preserving order) into the sequence.

Actually, before (b), that part of the current coded sequence (the input sequence initially), CS, which is prior to the first coding position found in (a), is retained to form part of the coded sequence for the current iteration. After the input sequence has been coded by the set of patterns, the ICAP is obtained, and that coded sequence with the minimum value (best pattern coding for that set) is retained.

The above operations (a) and (b) are performed for each
TABLE 8.1. ENCODING OF THE GIVEN SEQUENCES, IN EC.
set of patterns in the class model. Table 8.2 gives an example for the T2 set of patterns found in G1. Notice that the initial part ('AA') of the second coding is the same as for the first, except that an element of it ('2') is in the pattern B. It will be noted that the procedure (a) can commence with a test to determine if the first match position corresponds to that in the current coded sequence - if it is, then (b) is cancelled and the next pattern in the set is used for the iteration. This is the case in coding patterns C and D in Table 8.2.

As explained in Section 8.1.1, T4 patterns are found by matching the input and the given sequences, coded with T2 patterns. An ordering condition exists on the matching - each T4 pattern must be generated within the same T2 pattern bounds, but it is possible for a T4 to include a T2 pattern.

The match consists of the following steps:

(i) the corresponding T2 pattern code bounds are found for both sequences, (see Table 8.3 for definition).

(ii) a match is sought for each element within the bounds of the given sequence, to all the elements in the input sequence's bounds.

(iii) if a match occurs, then the following elements from the matched ones are compared. If these are not the same, then control reverts to (ii); otherwise:

(iv) an operation testing for any longer T4 patterns including the above matched elements, is performed by:

. each element of the above match in the given sequence being compared (as in (ii), (iii)) for a match, and

. each element of the above match in the input sequence being compared to all elements
INPUT: 22222222233344545555555 (TEST 1, NO. 3)

CLASS 1:-

SET 1: A B C D

<table>
<thead>
<tr>
<th>NO.</th>
<th>CODE</th>
<th>Coded Input</th>
<th>JS</th>
<th>RETAINED</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>A</td>
<td>AAA33344CD0D</td>
<td>12</td>
<td>NO. 1</td>
</tr>
<tr>
<td>2</td>
<td>B</td>
<td>AA22B3344CD0D</td>
<td>13</td>
<td>NO. 1</td>
</tr>
<tr>
<td>3</td>
<td>C</td>
<td>(AS FOR 1)</td>
<td>-</td>
<td>NO. 1</td>
</tr>
<tr>
<td>4</td>
<td>D</td>
<td>(AS FOR 1)</td>
<td>-</td>
<td>NO. 1</td>
</tr>
</tbody>
</table>

TABLE 8.2. CODING THE INPUT WITH T2 CODES, BY EC.

NO 1 2 3 4 5 6 7 8 9 10 11 12 13
G A A B B 3 3 4 4 4 C C D D
I A A A 3 3 3 4 4 C D D D
(THESE NUMBERS ARE USED IN TABLE BELOW)

<table>
<thead>
<tr>
<th>PATTERN BOUNDS</th>
<th>T4 CODES EXTRACTED</th>
<th>MATCH BY</th>
<th>NUMBER BOUNDS</th>
<th>SYMBOL</th>
</tr>
</thead>
<tbody>
<tr>
<td>A-B</td>
<td>AA</td>
<td>G 1</td>
<td>0-3</td>
<td>Z</td>
</tr>
<tr>
<td>A-C</td>
<td>33</td>
<td>G 5</td>
<td>2-5</td>
<td>3-10</td>
</tr>
<tr>
<td>A-D</td>
<td>44C</td>
<td>G 7</td>
<td>4-12</td>
<td>3-10</td>
</tr>
<tr>
<td>B-C</td>
<td>3344</td>
<td>G 5</td>
<td>4-12</td>
<td>3-10</td>
</tr>
<tr>
<td>B-D</td>
<td>CDD</td>
<td>G 11</td>
<td>4-12</td>
<td>3-10</td>
</tr>
<tr>
<td>C-D</td>
<td>-</td>
<td>-</td>
<td>13-14</td>
<td>11-13</td>
</tr>
<tr>
<td>D-1</td>
<td>-</td>
<td>-</td>
<td>13-14</td>
<td>12-13</td>
</tr>
</tbody>
</table>

TABLE 8.3. EXTRACTING T4 PATTERNS BY EC (TEST 1).
following the first element of the match from (iii) in the given sequence's bounds.

If a longer match is found, then it is stored and (iv) is iterated again; then control reverts to (i). This procedure is iterated for each set of bounds defined by T2 patterns (calculated in (i)). In the case where an expected pattern is missing in the input, the bounds are defined by the following patterns.

Table 8.3 show the match between the given (G) and input (I) coded sequences of Table 8.2. Each element in the sequences is numbered in the upper table. The lower table presents the details of the match. For each pattern bound, there is a corresponding number bound in each sequence. Note that because the B pattern is missing in I, the bounds for A-B and B-C are the same. The C-D bounds are reduced by the 'CDD' T4 pattern. Other T4 patterns generated, but not necessarily used in the final set, are shown in the T4 column. In the B-C range, four patterns are generated - three overlap, so that the longest '3344' is taken, while the remaining pattern matched has elements in the next set of bounds.

8.1.3. The Decision

The T4 patterns are coded into CS, to produce a final sequence, FS. The FS which has the minimum information capacity, (MCAP), for all the sets of patterns in the class, is retained. The calculation of MCAP is similar to ICAP:

\[ \text{MCAP} = \log_{10} \left( \frac{(6+N+M) \times JF}{10} \right) \]

where

- N is the number of T2 patterns,
- M is the number of T4 patterns, and
- JF is the number of elements in FS.

That class which has the minimum MCAP is given the decision. In Table 8.4, the CS, T4 patterns, FS, and MCAP are given
input: 2222222233344545555555 (TEST 1, NO. 3)

class 1:

CS: AAA33344CDDD
coded g1: AABBB33444CCDD
T4 codes: AA(=z), 3344(=Y), DD(=X)
fs: ZA3YCXD MCAP = 6.68 *

class 2:

set 1: CS: 22222222A3344C55555555
decoded g2: AABBB5CC4CC
T4 codes: -
fs: 22222222A3344C55555555 MCAP = 19.08

set 2: CS: 22222222A334C55555555
decoded g2: AABBB5CC54
T4 codes: C5(=Z)
fs: 22222222A334Z55555555 MCAP = 18.00

class 1 ("U") receives the decision, (correct).

table 8.4. final coded sequence and measure (by EC).
for the input (I), for each of the pattern sets of Table 8.1. The class 1 receives the decision.

8.1.4. Feedback to the Model

The program's decision is validated by the outside teacher. In the example (of Table 8.4), the decision is correct.

The model is refined by adding sets of T2 patterns, found in the current input, to those already stored. The T2 patterns are found in the input in the same way that these types are found in the given sequence, (Section 8.1.1). This new set is inserted into the given sequence of its class, and the resultant coded sequence is also stored. Thus the class model consists of sets of T2 patterns and the corresponding coded (given) sequence for each set.

For the example (of Table 8.4), the T2 patterns found in the input, and the coded given sequence are shown in Table 8.5. The full set of T2 patterns for the 12 inputs in Test 1 formed by EC, are shown in Table 8.6. It can be seen that the main differences (on the T2 level) between the classes are the existence of '22' and '55' kinds of patterns (in class 1) and the '23' and '45' kinds (in class 2).

8.2. Discussion of Program EC

8.2.1. Comments on the Programming in EC.

The coding of T2 patterns as employed in EC, is not an 'optimum' procedure, in the sense that certain sequences exist for which the 'best' placing of the codes is not formed. Consider, for example, the input, 2232323232323 and the codes, A= 223, B=23.
INPUT:  2222222233344545555555
G1:    222222223334445455555555

SET 1: T2:  A  B  C
        222 45  555

CODED G1: AA23233344BB5BC5

SET 2: T2:  A  B  C
        2222 45  555

CODED G1: A2223233344BB5BC5

TABLE 8.5. STORE ADDITIONS FOR EC (TEST 1, NO. 3).

<table>
<thead>
<tr>
<th>CLASS 1:</th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>A</td>
<td>B</td>
<td>C</td>
<td>D</td>
</tr>
<tr>
<td>1</td>
<td>222</td>
<td>23</td>
<td>545</td>
<td>55</td>
</tr>
<tr>
<td>2</td>
<td>222</td>
<td>45</td>
<td>555</td>
<td>55</td>
</tr>
<tr>
<td>3</td>
<td>2222</td>
<td>45</td>
<td>555</td>
<td>55</td>
</tr>
<tr>
<td>4</td>
<td>222</td>
<td>23</td>
<td>555</td>
<td>55</td>
</tr>
<tr>
<td>5</td>
<td>2223</td>
<td>45</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>2223</td>
<td>55</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>2222</td>
<td>32</td>
<td>34</td>
<td>45</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CLASS 2:</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>A</td>
<td>B</td>
<td>C</td>
</tr>
<tr>
<td>1</td>
<td>223</td>
<td>23</td>
<td>54</td>
</tr>
<tr>
<td>2</td>
<td>223</td>
<td>45</td>
<td>454</td>
</tr>
<tr>
<td>3</td>
<td>2323</td>
<td>454545</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>2323</td>
<td>454545</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>2323</td>
<td>45555</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>2323</td>
<td>45555</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>2323</td>
<td>545</td>
<td>55</td>
</tr>
<tr>
<td>8</td>
<td>2323</td>
<td>455</td>
<td>55</td>
</tr>
<tr>
<td>9</td>
<td>2323</td>
<td>54545</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>3232</td>
<td>54545</td>
<td></td>
</tr>
</tbody>
</table>

TABLE 8.6. T2 PATTERN STORE IN EC AFTER TEST 1.
Then the alternatives produced would be,

\begin{align*}
A1: & \quad \text{AA23AB} \\
A2: & \quad 2 \text{B2BB2BB}
\end{align*}

but the 'best' coding is \text{AAB2B} using the criterion of ICAP. Such situations are found to occur, but often, either the class of the pattern set is the incorrect one, or the approximation (A1) is close to the 'best' value, and few errors result.

More troublesome is the lack of optimization between the T2 and T4 patterns. In some cases, elements corresponding to a T2 pattern could have, if not coded, been included into a T4 pattern. If A=334 is of T2, then a given sequence of

\[ \text{233445} \quad : \text{S1} \]

is coded into

\[ \text{2A45} \quad : \text{S2} \]

An input of

\[ \text{2343446} \]

is not coded by the T2 pattern and, assuming the input lies between the appropriate bounds, is not coded by any of T4. However, if the input is compared to S1, then the T4 pattern '344' could be generated. It will be noticed in the above example that the requirement of an exact match to A has meant '343' (vectorially equivalent to '344') is not extracted. This is a limitation of the approach, (see Section 7.3).

Further difficulties with the above optimization can arise through registration in the matching of T2 patterns. A code, A=33, can be inserted into an input of

\[ \text{23334} \]

in two ways, viz.

\begin{align*}
A1: & \quad -2A34- \\
A2: & \quad -23A4-
\end{align*}

It may happen that one of these alternatives will allow a T4 pattern to be generated, but the other will not. Alternative possibilities are not considered in EC, and the first insertion is always made, (i.e. A1).
The use of MCAP (and ICAP) as a measure of similarity (actually it is dissimilarity) has the advantage of relating the number of codes to their effectiveness in reducing the length of the sequence. This is important in discriminating between two classes both having the same T2 codes, but one possessing some extra codes, which however are useful only for the class with longer sequences. Such examples may arise in sequences for figures (e.g. 7, 2; 7, 1).

However, the above measure has the disadvantage of not being dependent upon the number of insertions of each code in a sequence. Thus, whether an input is coded _ABCD_ or _AAAA_, the MCAP of both is the same (for patterns A, B, C, D). The measure would therefore tend to be ineffective in cases where the relative number of insertions of a particular pattern is important, (e.g. 6, 0; 7, 7).

8.2.2. General Comments on EC

Perhaps the most serious problem with program EC is the lack of generality in its model. This lack is manifested in various ways. In the first place, there is no compact means for expressing the class model. The nature of the T2 patterns and the data required for the T4 patterns separate the store into two parts. Of course, it is unnecessary to store the coded sequences if the given sequence is retained, but if this method is adopted, more processing is required to recognize an input.

Many of the T2 pattern sets are generated from the same sequence. Those which are, commonly have a difference of order in the elements of one of the patterns, (see Table 8.6). Such variations could be grouped into some generalized form to reduce storage and processing time, but there is no facility for this in EC.
It would appear that the T2 patterns extracted are not good descriptors of the members of the class, for a combination of two reasons:

(a) the longest length patterns are sought,
(b) the 'exact match' is required for coding.

Thus, long patterns (e.g. '4545', '3334', '45555' of Table 8.6) have little chance of being inserted into many members of the class. It seems more appropriate to minimize the number of elements in a pattern, so that the pattern can at least be inserted into a greater number of members. Another idea is to find those patterns which are 'similar' (in terms of having the same elements) to those already found, rather than generate them independently of the current class model.

Finally, on this point of a 'generalized model', there is no common set of T4 patterns for a class, in the model - this reflects the variability of the patterns in the members. The effectiveness in description of the T4 discovered, depends on the 'goodness' or representativeness of the given members. The extent to which the patterns can be useful in discrimination depends on the differences in the given sequences. Fortunately, for the Tests considered here, the given members are 'typical' - a designed property. It was found, however, that matches over non-characteristic areas for one class could lead to incorrect decisions. For example, one result (Test 2, No.5) is given mainly on the basis of a T4 match of 7 elements in the final loop of the input '5' and the given 'S'.

Despite the problems discussed in this Section 8.2, the number of errors in Tests 1, 2 and 14.6 have been found to be comparable to those of an 'average subject'. The
errors in Test 3 are caused by the poor T2 patterns generated from the given ('Z') member. The reason for the good performance of EC would, as mentioned in the previous paragraph, reflect the 'goodness' of the given members for the T4 patterns. A more important reason perhaps, is that the T2 patterns, although not very good descriptors, are nevertheless characteristic of the class, (see Table 8.6). The lack of ability of each pattern to provide a description, is offset to some extent by the large number of T2 patterns stored.

The relative effectiveness of the T2 and T4 patterns can be gauged from Table 8.7. For the pattern set which receives the decision from EC, (i.e. the 'best' coding), the number of patterns per sequence, the average pattern length, and the number of times each pattern occurs, are given for both types. From these figures, the average number of elements by which the original sequence is reduced (allowing for the one code letter inserted per pattern) can be found, and is shown in Table 8.7. In all cases, the T2 patterns are, on the average, more effective in coding. The average coding susceptibility to both types of patterns is reflected, for each Test, in the reduction values when these values are compared to the average length of the input sequences.

8.3. The Program ED

Program EC obviously does not form a satisfactory recognition program, because of its dependence on the first given class members. A modification has therefore been made - viz. during the learning phase, the T2 patterns and the coded sequence for the input (instead of the given member as in EC) are added to the class model. In addition, to reduce the storage for the model, only one set of T2
The results of ED are considerably worse than those from EC. This can be attributed to the dependence of T4 patterns on various members of the input, which gives the input more generality and ability to find different regions in the known sequences and input patterns. It is possible for EC to be more sensitive to the class of pattern used, whereas T4 is independent on this. (See Section 6.4.)

Another program (KE) was developed in an attempt to overcome some of the problems found in EC and ED. In particular:

(a) if possible, an 'averaged' class model for each class,

(b) an adequate measure — another measure incorporating the number of repetitions of the patterns was desired.

The programs, KE and EC, possess these features.

The operation of KE is presented by working through an example (the same example as given for EC, Section 8.3.1), 8.9.1. The Model in KE

The class model consists of an ordered set of T2 patterns, which are generated under the conditions—

<table>
<thead>
<tr>
<th>T2 PATTERNS</th>
<th>T4 PATTERNS</th>
<th>AV. NO. OF ELEMENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>A</td>
<td>B</td>
</tr>
<tr>
<td>TEST 1</td>
<td>2.6</td>
<td>3.5</td>
</tr>
<tr>
<td>TEST 2</td>
<td>4.0</td>
<td>2.0</td>
</tr>
<tr>
<td>TEST 3</td>
<td>3.5</td>
<td>3.4</td>
</tr>
<tr>
<td>TEST 4</td>
<td>6.0</td>
<td>2.4</td>
</tr>
</tbody>
</table>

A: NO. OF PATTERNS IN SEQUENCE.
B: AVERAGE NO. OF ELEMENTS PER PATTERN.
C: NO. OF RECURRENCES PER SEQUENCE (=1 FOR T4).

TABLE 8.7. RELATIVE EFFECTIVENESS OF T2 AND T4 PATTERNS IN CODING THE SEQUENCES GIVEN THE DECISION.
patterns per figure is added to the store.

The results of ED are considerably worse than those from EC. This can be attributed to the dependence of T4 patterns on various members of the input, which gives the input more scope for matching over non-characteristic regions in the known sampled and input members, (see Section 8.2.2). It is possible for a member of one class to have more 'correlation' with a member of another class than with a member of its own class. This is reminiscent of Sebestyen's "proximity algorithm" (175) which, in its original form, is too dependent on specific members, (see Section 4.4).

The effect of having only one set of T2 patterns per input added to the model, does not appear to be marked.

8.4 Operation of Program EE

Another program (EE) was developed in an attempt to overcome some of the problems found in EC and ED. In particular, two points were considered:  
(a) generality - it was desired to have a compact and, if possible, an 'averaged' class model for each class.
(b) an adequate measure - another measure incorporating the number of repetitions of the patterns was desired.

The programs, EE and EF, possess these features.

The operation of EE is presented by working through an example (the same example as given for EC, Section 8.1).

8.4.1. The Model in EE

The class model consists of an ordered set of T2 patterns, which are generated under the conditions:
(a) the first to occur in the sequence is taken,  
(b) if there is a choice, the shortest pattern is taken. 
Initially, the patterns are found in the first members given for each class; the process for the succeeding inputs is discussed in Section 8.4.3. Also stored in the model is the expected number of repetitions for each pattern - after the first member, this value corresponds to the number of times the pattern has been found in that sequence.

Table 8.8 shows the two set of patterns generated from the first members of each class in Test 1.

8.4.2. Coding the Input to Make the Decision

An input is coded with the patterns in the class model. This process follows that used in EC (see Section 8.1.2), with a modification allowing patterns, which may occupy the same ordered position in the set ('OR' combinations; see Section 8.4.3), to be alternatives for coding in that position.

The 'similarity' measure, which is also the criterion for the 'best' coding, is the product of two factors:-

(a) $F_1$ : the average difference (for each pattern) between the number of expected and obtained repetitions,

(b) $F_2$ : the ratio of the number of uncoded elements in the coded sequence, to the expected number of coded symbols.

This measure is evaluated in the coding procedure after each insertion of a pattern set. The second factor, $F_2$, is incorporated to reduce the similarity to those classes which leave much of the input uncoded, but which have a high $F_1$ value. This would occur in discriminating 9 and 2.

Table 8.9 gives the results of the coding procedure for the third input in Test 1. The difference between the
<table>
<thead>
<tr>
<th>NO</th>
<th>CODE</th>
<th>NO. OF REPETITIONS</th>
<th>SYMBOL</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>22</td>
<td>3</td>
<td>A</td>
</tr>
<tr>
<td>2</td>
<td>23</td>
<td>2</td>
<td>B</td>
</tr>
<tr>
<td>3</td>
<td>45</td>
<td>2</td>
<td>C</td>
</tr>
<tr>
<td>4</td>
<td>55</td>
<td>2</td>
<td>D</td>
</tr>
<tr>
<td>1</td>
<td>223</td>
<td>2</td>
<td>A</td>
</tr>
<tr>
<td>2</td>
<td>23</td>
<td>2</td>
<td>B</td>
</tr>
<tr>
<td>3</td>
<td>54</td>
<td>2</td>
<td>C</td>
</tr>
<tr>
<td>4</td>
<td>45</td>
<td>2</td>
<td>D</td>
</tr>
</tbody>
</table>

**TABLE 8.8 CODES FORMED BY EE (TEST 1, NOS.1, 2).**

**INPUT:** 2222222223344455555 (TEST 1, NO.3)

<table>
<thead>
<tr>
<th>CODE</th>
<th>CODED SEQUENCE</th>
<th>DIF</th>
<th>MEASURE</th>
<th>SEQ.KEPT</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CLASS 1</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>AAAAB334CCDDD</td>
<td>3</td>
<td>0.25</td>
<td>A</td>
</tr>
<tr>
<td>B</td>
<td></td>
<td></td>
<td></td>
<td>A</td>
</tr>
<tr>
<td>C</td>
<td></td>
<td></td>
<td></td>
<td>A</td>
</tr>
<tr>
<td>D</td>
<td>AAAAB334C4DDDD</td>
<td>5</td>
<td>0.55</td>
<td>A</td>
</tr>
<tr>
<td><strong>CLASS 2</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>2222222A334C555555</td>
<td>6</td>
<td>3.37</td>
<td>A</td>
</tr>
<tr>
<td>B</td>
<td>2222222B334C555555</td>
<td>6</td>
<td>3.56</td>
<td>A</td>
</tr>
<tr>
<td>C</td>
<td>(AS FOR 'A')</td>
<td></td>
<td></td>
<td>A</td>
</tr>
<tr>
<td>D</td>
<td>2222222A334DD55555</td>
<td>5</td>
<td>2.50</td>
<td>D</td>
</tr>
</tbody>
</table>

**TABLE 8.9. CODING AN INPUT, IN EE (TEST 1, NO.3).**
expected and obtained repetitions for all patterns used - a 'linear error' - is shown in the DIF column.

The decision of input class membership is given to that class which has the minimum value for the measure. In the example (Table 8.9) class 1 receives the decision.

8.4.3. Feedback to the Class Model

To update the class model, a search for T2 patterns, which are not simply repetitions of the known ones, is made in the coded input sequence. This procedure operates under the same conditions mentioned for the initial generation in the given members ( (a) and (b), Section 8.4.1).

Several situations may occur in which a new pattern is found: -

(a) the repetition is between two consecutive patterns, e.g. - A2222B -. The pattern '22' is inserted between A and B in the class model.

(b) the repetition includes a known pattern, e.g. - AB2B2C -. The pattern 'B2' is stored as an alternative to 'B' (i.e. as an 'OR' pattern).

(c) the repetition occurs between two non-consecutive patterns, but is a subset of one of the missing patterns, e.g. - A2222D - and B=223 and C=33. The pattern '22' is stored as an 'OR' combination to B.

(d) other cases not included above; in which the generated patterns are dismissed by the program as having an ambiguous position (in the class model).

The above procedures specify how patterns are included in the class model.

The coded sequence for the correct decision (class 1) does not produce any more patterns in the example (see
Table 8.10), that is, none of the above conditions (a) to (d) exist.

Also stored for each class is a set of parameters whose values determine the expected repetition of a pattern in the input. A count of the number of times a pattern is discovered in each member is stored in an array, \( F \) - the values for each element of \( F \) indicate the count for that number of repetitions of the discovered pattern. Table 8.10 gives the updated \( F \) array for class 1. The code \( A=22 \) has been found 4 times in one member and 3 times in another.

A count of the number of inputs in which the code has been found, \( (Q) \) is also stored. \( Q \) therefore commences its count from the input from which the pattern is generated.

To determine which of the patterns are to be used in the current class model, those with a frequency of occurrence of greater than a half (0.5) are chosen. The number of expected repetitions is the maximum value, \( X \), of the \( F \) array for which,

\[
\sum_{n=M}^{X} \frac{F(n)}{Q} \geq 0.5
\]

where \( M \) is the maximum number of repetitions found for that pattern. Table 8.10 shows the updated expected repetitions for the codes of class 1. It should be noted that any newly-generated pattern will immediately be incorporated into the current model.

It is possible for a pattern to have an expected repetition of zero (\( X=0 \)), in which case it is not included in the current model. Such patterns are searched for in the input coded sequence and their \( F \) array and \( Q \) value updated, as for the other patterns. This procedure allows a pattern which may have 'died' at one stage, to be present
INPUT: A A A A B 3 3 4 C C 0 0 0 (CODED NO.3)

F ARRAY:-

<table>
<thead>
<tr>
<th>CODE</th>
<th>F(4)</th>
<th>F(3)</th>
<th>F(2)</th>
<th>F(1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>22</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>23</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>45</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>55</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

UPDATED MODEL:-

<table>
<thead>
<tr>
<th>CODE</th>
<th>CURRENT REPETITION</th>
<th>OBTAINED REPETITION</th>
<th>UPDATED REPETITION</th>
</tr>
</thead>
<tbody>
<tr>
<td>22</td>
<td>3</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>23</td>
<td>2</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>45</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>55</td>
<td>2</td>
<td>3</td>
<td>3</td>
</tr>
</tbody>
</table>

TABLE 8.10. UPDATING THE MODEL IN EF (TEST 1).
in the model at a later stage. A pattern which has an
expected occurrence of unity \((X=1)\) may be compared to a
\(T^4\), but the difference is that the former has to be
generated as a \(T^2\) initially. Thus some common \(T^4\) patterns
are not incorporated into the class model of EE.

Table 8.11 presents the discovered codes at the end
of Test 1, and the expected form for a member of each
class. Note that there are some patterns (e.g. '455',
'54', '45555') which have been omitted from the current
model, because they do not occur often enough.

8.5. Discussion of the Program EE

8.5.1. Programming Problems

The coding of the patterns in the input is (as in
EC) not an optimum procedure. One reason for this is
the approximate adjusting of code positions mentioned
in Section 8.2.1. Furthermore, in EE, not all
combinations with the OR patterns are used to optimize
the value of the 'similarity' measure. Thus, the first
patterns are not optimized with respect to later OR
combinations. This can cause errors especially if the
OR combinations are 'good' descriptors.

The form of the 'similarity' measure received many
modifications during its development. Some of the
incorporated features and their alternatives considered,
are:

(a) 'linear' rather than a 'squared' difference,
between expected and obtained occurrences for a pattern.

(b) the average difference per pattern rather than the
total difference,

(c) the counting of all codings for a pattern as against
the counting of only consecutive repetitions. The
TABLE 8.11. PATTERN STORE IN EF (TEST 1).
problem is whether the A in ___AA\textsuperscript{34}A___ occurs 3 or 2 times. The former is taken (3 times) because it appears a more general rule,

(d) F2 incorporated as part of a product (or some other form). It was later thought that the square root of the sum of the squares of the F1 and F2 would have been better, because when the factors are taken, as a product, if either factor is zero, so is the total - in one case, for example, (Test 2, No.10), F1 is zero (producing an incorrect decision), and yet F2 has a high value.

There was (initially) some doubt as to whether the Q count for a pattern should have been the total number of sampled members, rather than the number of sampled members since the occurrence of (and including) the generating sequence. The latter has been chosen in order to allow the new patterns to affect the class model immediately. It has been assumed that further training samples would be given to stabilize the occurrence value of the patterns.

An alternative to determining the expected occurrence of a pattern is to take the mean of the values in the F array. Thus an array,

$$F(3)=1$$
$$F(2)=0$$
$$F(1)=1 \text{ and } Q=2$$

would have an expected occurrence of 2, compared to 3 under the present scheme in EE. The above suggestion has not been implemented because it appears to indicate an occurrence (e.g. 2 in the above example), which may not be characteristic of the class.

The effects of the above two points have caused errors
in some tests. A newly generated pattern is often expected to occur too many (or not enough) times, because its occurrence value is unstabilized. The averaging of this value takes typically 3 to 5 samples. It often happens (Test 14.2, in particular) that extraneous segments in later samples generate new patterns which remain in the class model at the end of the learning phase.

The condition of taking the 'first-up' pattern in the generation of new patterns, does not appear to be a problem, as those generated are usually 'good' descriptors.

8.5.2. General Comments on EE

It was found that some sequences did not lend themselves to description solely by T2 patterns. For example, the initial part of the 'S' in Test 2, is often too small (compared to the size of the grid) to show many repetitive elements. Furthermore, a curve (as in the 'S') instead of straight lines, tends not to show patterns. The consequence is that the F2 value (in the 'similarity' measure) is increased. It may be better to store an 'expected' F2 value with which the actual one obtained can be compared.

In the case of the figure '6' (Test 14,2), the smaller (or latter) curve requires many samples (for the reasons discussed above) in order to generate a suitable description. On the other hand, the '0' forms an adequate model more quickly and patterns in the '0' model fit well to the members of the '6' class; consequently few inputs of Test 14.2 are classified correctly as a '6'.

The use of the coded sequence from which to extract patterns, proved to have disadvantages. Some possibly useful patterns are 'covered' by the known ones - for
example, possession of a '43' pattern covers the generation of a '34', as can be seen in an input of \_\_\_343434\_\_, which is coded as \_\_\_3AA4\_\_, and the better fitting '34' pattern cannot be subsequently generated. This problem is not serious because the program adapts to the expected occurrences of the '43', and therefore the consequence is that only the F2 value is increased.

In certain situations, the better fitting pattern is not generated until later in the learning phase. An example of this is shown in Test 1, (Table 8.11), for 'V', where the '545' pattern is found to be better than the '54', later in the learning. Again, this problem does not appear to be serious provided that suitable training samples are presented.

An advantage of the generation procedure is that it does preclude many alternatives which are not very good descriptors, but which are found by EC simply because they are repetitive. The size of the model is therefore kept compact, and the storage and processing times are reduced.

The class model is obviously limited in the types of 'concepts' it can handle, (see Section 9.1). In Test 1, the second stroke of the 'V' can range from '45555' to '45' in patterns. The compromise found by EE is to store an intermediate pattern, '545' and a '45'. The '45555' code does not occur often enough in the samples and is purged.

The results from EE have not been as good as expected. The main problem which appears to affect the results is the poor generation of patterns, which leads to unstabilized class models, and hence to errors.
8.6. The Program EF

The reason for having the minimum number of elements in the patterns at 2 (in EE), is a 'carry-over' from EC (and ED). This minimum is not profitable in EC (and ED), because there is no MCAP reduction for the coding of a single element. For the measure of EE, however, there is an advantage in coding single elements, because the number of occurrences of a pattern is considered. This modification to allow 1-element patterns was made to EE to form EF. The 'similarity' measure (in EF) was simplified to include only F1, (see Section 8.4.2). The programming details of EF are given in Appendix C.

The results from EF are much better than those of EE. This can be attributed to the fact that 1-element patterns in the tests are better descriptors for the sequences considered, than patterns with more elements. Somewhat poorer results are given by EF in Test 3, where the second (slanting) line of the 'Z', is partly described by '21' and partly by '1', but the better fitting pattern of '211' is not generated, (e.g. Test 3, No.10).

In general, more patterns are generated in the class model with the T1 descriptors. Consequently, it is found more difficult to insert the generated patterns into their correct order, and quite often, 'ambiguous patterns' (Section 8.4.3) are formed. In some cases, alternatives are formed, but are later purged because they are not to be found in the following members - however, their presence in the model can cause errors (e.g. Test 14.2, No.12).

Other problems which cause errors, are the lack of averaging of the pattern's occurrence values, (e.g. Test 3, No.3), and the poor similarity measure, as
explained in Section 8.5.1, (e.g., Test 2, No. 12).

Of considerable interest are the actual class models which are generated by EF. The description of the 'average' expected figures for some Tests is given in Table 8.12. The asterisked patterns are ones which have been generated within the last two samples in the learning phase, and consequently their occurrence values are regarded as being unstabilized. It can be seen from the exemplars of Test 14.2, that the main difference (the characteristic difference) between the classes, centres on the '54' and '5' patterns existing in the figure '0'. The asterisked patterns in Test 14.2 are, in fact, caused by extraneous segments in the samples; their occurrence values are expected to decrease with more learning. In the other Tests, the differences are emphasized on the regions which are important in discriminating the figures. The absence of many OR patterns is due to the occurrence values for these combinations being below 0.5 and hence these patterns are not included into the current model. OR patterns of this kind do exist in many of the models.

Other recognition tests have been given to EF, in order to test the adequacy of the program's model - namely, Tests 8, 9 and 14. The performance of EF on these Tests is not much worse than, and certainly is comparable to, that obtained from the first series of programs, (Table 5.11). However, EF does lack generality, for example, in describing members whose lines vary considerably in orientation, and in describing classes which may have subsets. For such reasons (more fully discussed in Section 9.3), the number of tests given to EF, has been limited.
<table>
<thead>
<tr>
<th>TEST FIG</th>
<th>AVERAGE MEMBER</th>
</tr>
</thead>
<tbody>
<tr>
<td>'U'</td>
<td>222222 3232 333 444 5454 5555555</td>
</tr>
<tr>
<td>'V'</td>
<td>22 32323232 44 5454 555555 4545</td>
</tr>
<tr>
<td>'S'</td>
<td>616161 21 333333 22 1111 666</td>
</tr>
<tr>
<td>'Z'</td>
<td>44 343434 21212121 1 43434343</td>
</tr>
<tr>
<td>'0'</td>
<td>1111 1212 2222 333 444 5454 5555 666 1111</td>
</tr>
<tr>
<td>'6'</td>
<td>55 61 1111 2121 2222 3333 4444 555 666 1111 2222 212212</td>
</tr>
<tr>
<td>'G'</td>
<td>6161 1111 21 2222 3333 444 5555 6161</td>
</tr>
<tr>
<td>'E'</td>
<td>4444 555 66 111111 222222 333333 4343 44</td>
</tr>
<tr>
<td>'C'</td>
<td>4 55 666 11111 222222 333333 4444</td>
</tr>
<tr>
<td>'0'</td>
<td>11 2121 222222 32 333333 4343 4444 5454 5555555 666</td>
</tr>
<tr>
<td>'6'</td>
<td>111 2121 222222 3333 4343 444 5555 66 6161</td>
</tr>
</tbody>
</table>

*: PATTERN (UNDERNEATH) GENERATED WITHIN THE LAST 2 SAMPLES.

**TABLE 8.12.** AVERAGE MEMBERS PRODUCED BY EF.
8.7. Summary

Four programs have been discussed in this Chapter. These have an ability to learn to recognize the members of the classes in various discrimination tests. EF produced the best results on the tests given - it did show that discrimination can be achieved effectively as a process of model construction. The characteristic features of each class become apparent when one class model is compared to that of another class. Processing incorporated in the programs highlights these differences.

It would appear, however, that EF is a weaker pattern recognition program than AN6 of the first series.
### RESULTS FROM TEST 1.

<table>
<thead>
<tr>
<th>NO</th>
<th>FIG</th>
<th>AM</th>
<th>FA</th>
<th>GA</th>
<th>AN6</th>
<th>EC</th>
<th>ED</th>
<th>EE</th>
<th>EF</th>
<th>S</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>U</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>V</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>U</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>V</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>U</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>V</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>7</td>
<td>U</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>8</td>
<td>V</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>9</td>
<td>U</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>10</td>
<td>V</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

S: HUMAN SUBJECT ON THE 'PSYCHOLOGICAL EXPERIMENT'.

### RESULTS FROM TEST 2.

<table>
<thead>
<tr>
<th>NO</th>
<th>FIG</th>
<th>AM</th>
<th>FA</th>
<th>GA</th>
<th>AN6</th>
<th>EC</th>
<th>ED</th>
<th>EE</th>
<th>EF</th>
<th>S</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>S</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>S</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>S</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>S</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>S</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>S</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>7</td>
<td>S</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>8</td>
<td>S</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>9</td>
<td>S</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

S: HUMAN SUBJECT ON THE 'PSYCHOLOGICAL EXPERIMENT'.

### RESULTS FROM TESTS 1 AND 2.

TABLE 8.13. RESULTS FROM TESTS 1 AND 2.
### RESULTS FROM TEST 3.

<table>
<thead>
<tr>
<th>NO</th>
<th>FIG</th>
<th>AN6</th>
<th>EC</th>
<th>ED</th>
<th>EE</th>
<th>EF</th>
<th>S 1</th>
<th>S 2</th>
<th>S 3</th>
<th>S 4</th>
<th>S 5</th>
<th>S 6</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>'6'</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>'0'</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>'0'</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>'6'</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>'0'</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>'0'</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>'6'</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>8</td>
<td>'6'</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>9</td>
<td>'0'</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>'6'</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>11</td>
<td>'6'</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>12</td>
<td>'0'</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>13</td>
<td>'6'</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>14</td>
<td>'0'</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

S: HUMAN SUBJECT ON THE 'PSYCHOLOGICAL EXPERIMENT'.

### RESULTS FROM TEST 14.2.

**TABLE 8.13. (CTD). RESULTS FROM TESTS 3 AND 14.2.**
Figure 8.1. The Input Figures of Test 14.2
Figure 8.1 (Ctd.). The Input Figures of Test 14.2
Chapter 9

DISCUSSION OF THE PROGRAMS OF CHAPTER 8

Various aspects of the programs in the second series (in particular, EF) are now discussed. The main aspect will be the performance of EF in different types of environments. In Section 9.2 the possibility of an extension to producing a simulation of performance of the human subject in the 'psychological experiment' (Section 7.1), is briefly mentioned, while in Section 9.3 a comparison of EF with AV6 (of the first series of programs) is made.

9.1. General Discussion of the Programs

The results of EF (and to some extent, the other programs in the second series), indicate that a program can be written which will learn to recognize the class membership of certain sets of sequences. These programs are constrained to treat the numbers as abstract symbols. Apart from the results, examination of the class models which EF constructed (Table 8.12) would support the above contention.

To be meaningful, the degree to which EF can learn, must be related to human performance. However, this comparison is difficult to make for a variety of reasons. In the first place, the program EF is not consistent in its results for figures, due to the inability of some routines to cope with certain sets of patterns, (see results of Test 3, Table 8.13). This decrease in performance is not evident in the human subject. It is possible, with more sophisticated programming procedures, for such inconsistencies to be removed in the machine. On the other hand, human performance is expected to
decrease when, for instance, the memory load increases - such as when more classes are present or the length of the sequences increases. These loads naturally cause no problem to the machine. Further difficulties arise in the comparison because of the conditions of the particular experiment from which human performance is being extrapolated. The conditions allow the subject to refer continually to the initial two sequences, whereas the program, EF, cannot do this. Again, the human performance is related to a set of specific tests, with a restricted sample of subjects being used. In spite of such difficulties, it is the author's opinion that the performance of the machine is close to what could be achieved by a human subject, for the given Tests, (Nos. 1,2,3,14.2).

Some deviations in program performance is expected for other extensions, such as an environment having a different redundancy. If the sequences were less redundant, the proportion of T1 and T2 patterns would decrease, and the class model would become less representative - eventually to the degree where presumably the important characteristics of the classes would not be extracted. In such a situation, T4 patterns may become important - their inclusion (which was originally considered) into the model for EF, could be performed by noting their regular occurrence between T1 (and T2) patterns. The presence of a set of elements occurring in more than 50% of the samples, would ensure that they were incorporated in the current class model.

For more redundant sequences, the class model would be suitable. This is because totally redundant sequences consist of one element repeated:-
and T1 patterns are perfect descriptors of this environment. Therefore, as the redundancy increases, the frequency of the T1 patterns, adequately represented in the model, would also increase. The operation of the program would consequently not be limited in this case. It has been assumed in the design of the programs, that the 'concept' for the class of given sequences, resides in a detailed description, provided by the inherent patterns in the sequences. However, one could devise a set of sequences in which their characteristic properties do not lie in T1 or T2 patterns.

\[\text{e.g. C1: 22422, in general, xx4xx,} \]
\[\text{C2: 22322, in general, xx3xx.} \]

It has further been assumed that the concept can be defined in terms of an ordered set of patterns. In the cases of EE and EF, some generality to incorporate OR patterns is allowed. However, if the difference between two classes resides in the property of 'A & B' where A and B are two separated patterns, the program cannot distinguish the classes. Sometimes, a satisfactory approximation can be made, as in the case of a rule 'A \lor B'. In this case, one class would expect A and B (an approximation) while the other class would expect neither. In some situations, the non-optimum insertion procedure for the codes (see Section 8.5.1) can cause a failure to define a concept, because an early (T1 or T2) pattern is not always inserted with a later OR pattern. Some problems would also be found in specifying when subsets of a class are to be formed. A suggestion (not expected to be very satisfactory), is 'less than 20% of the total number of patterns can be inserted into the input'.

The class models constructed by EF, contain specific
characteristics of the various classes for use in distinguishing the figures considered - these specific characteristics appear to correspond to a given human being's notion (for example the author's) of class differences, although these latter are ill-defined.

There is the problem, however, of averaging the frequency of the generated patterns, as explained in Chapter 8.5.2. Three to five samples are adequate to provide enough patterns for a class model. More are required if the samples differ considerably, (such as by having extraneous segments), to allow the frequency of occurrence to be stabilized. In most cases, however, the extra patterns which are generated in further samples are purged or become OR patterns, which are not usually employed in coding the inputs.

The programs do possess a degree of generality. EF, for example, allows any set of elements in the T1 and T2 patterns - there is no limitation on the number or combination of elements. Furthermore, discovery of the patterns is possible in any sequence of abstract symbols, having approximately the same redundancy as those discussed here. However, the programs are limited in their generality firstly, by the type of model formed (discussed above) and secondly, by the rather simple types of regularities discovered. Sequences of symbols are not necessarily structured by internal relations; sentences in a language are a good example. Further discussion of this limitation, which is common to the approach to number sequences used in this project, is given in Chapter 10.

The coding measure is really a measure of dissimilarity between given patterns. Some difficulty was found in devising a suitable function, and the possibility of
discovering a better measure still exists. The incorporation of frequency of occurrence is felt to be an important factor. It is interesting to note that totally redundant sequences, e.g. Class 1: 222
Class 2: 222222
must be discriminated on the basis of the recurrence of the symbol. The measure, (F1 and F2) including a linear difference in frequency, can be considered as a modification of a measure for the above, applied to partly redundant sequences.

9.2. Psychological Extensions

The programs have shown that concept learning can be considered in terms of model construction, as employed in Section 7.3. It is not suggested here that the human subject operates in this way - his behaviour indicates the adoption of an 'hypothesis testing' procedure, which is a suitable coding method for his limited memory. The method allows him to 'rule out' various parts of the sequence as being unimportant and to concentrate on the relevant parts. This behaviour was evidenced in the experiments, and is commented upon in many psychological studies, (25,30).

An extension to the programs could make them operate in a similar way. This could be considered as a suggestion for an attempt at simulating human behaviour. The emphasis on the 'characteristic' parts could be established by a process which makes two patterns in different class models the same when there is little difference between these patterns. Thus, in Test 14.2 (Table 8.12), the occurrence of '333' in one class and '3333' in another, together with the existence of 'larger' differences
( '5454 5555' compared to '55' ), could suggest making the former two the same at, say '333'. In effect, the process would be incorporating an hypothesis about the most important differences between the class models. By testing various possibilities similar to that above according to a strategy, the program could discover and emphasize the characteristics. (It was interesting to note in the 'psychological experiment', that most subjects employed a 'strategy' which tended to look for differences at the beginning or end of the sequence before the middle region).

By changing strategies and hypotheses, higher level concepts, involving logical combinations of patterns, could be formed. The subsequent processing of a new input could be modified to avoid the insertion of the same patterns in the sequence twice. That is, the difference in coding - a search for patterns - in a particular area need be the only processing involved. This would correspond to the subject's notion of unimportant parts.

A program to simulate human behaviour in the experiments would require the possibility of generating more pattern types in the sequences. In addition to those mentioned in Section 7.2, others might be the general trend in numbers (e.g. 4 - 3 - 2 - 3 ) along the sequence, and the arithmetic differences between successive numbers at various positions. The experiment itself would have to be studied further to provide information on the subject's behaviour when there is, for example,

(a) a varying degree of redundancy in the sequences,
(b) an increase in the number of classes, or
(c) an increase in the length of the sequences.

The role of memory, as found in the above extensions, is
important in a simulation. Another aspect which might be considered, is the subject's ability to discover different types of concepts embodied in the patterns.

9.3. Comparison of EF and AN6

The program EF is a recognition device and was given a number of tests (Nos. 8, 9, 14) in which its performance could be compared with the programs in the first series. However, as these tests do not contain some of the variants found in line drawings, a false impression of the performance of EF could be obtained. The following lists the differences between EF and AN6:

(a) AN6 is capable of recognizing figures which are rotated and vary in size; EF cannot recognize figures which have any significant changes in these factors.

(b) The tolerance for matching line segments in AN6 is greater than in the coding method. An input line segment can be within a specified tolerance, usually not less than \(40^\circ\). In contrast, the line segments in EF must occur exactly. The maximum tolerance when for example a '34' is inserted into a '34444' or a '33334' is \(38^\circ\). However, if the sequence contains -43- then the pattern cannot be inserted at all.

(c) Because of the nature of the averaging process, EF would appear to require more samples than AN6, to produce an AV.

(d) It would appear, although difficult to prove, that AN6 averages are less sensitive to individual members than those of EF. This is mainly because of the averaging over a total description of the figures, rather than over selected patterns.
(e) AN6 is capable of generating subsets, which EF, at present, cannot do.

(f) Use is made in AN6 of factors, (extracted from the descriptions), which are helpful in certain cases of discrimination. An example is the beginning-end vector to help distinguish the '6' and '0'. EF does not incorporate any such factor.

Because of these differences, there was no serious attempt to experiment with EF as a recognition device.

9.4. Summary

The previous Sections have attempted to outline some modifications and extensions to the second series of programs. The various features of the programs (not all evident in work cited in the literature) to recognize sequences, have been discussed here (and in Chapter 7); the more important of these features are:-

(a) an economical (but adequate) model of each class. This model is composed of a set of related descriptors - the possibility of having alternative patterns in the same position is allowed.

(b) the extraction of the patterns (or regularities) by a set of general rules. This processing is a search for simple relations in the input sequence.

(c) application of the 'coding method' (Chapter 2) in making a decision. The input is coded with previously found patterns. A measure is used to determine the reduction in coding and to provide a basis for decision.

(d) incorporation of learning, comprising several important procedures:-

. new patterns can be inserted into the model (but not every repetition of elements is
generated),
- stored patterns are 'averaged' (over their expected occurrence in class members),
- a pattern may 'die' and then possibly be 'reborn'.

The results from the program EF have indicated that its performance on certain tests is close to that expected from a human subject - there are tests on which the program would fail, but there are also some on which the human subject would fail, (Section 9.1).

Various extensions have been discussed for the programs. Perhaps, the most important is extension to a simulation of human behaviour occurring in the experiment. The main aim of this series of programs, however, has been an investigation into the coding method on sequences of abstract elements.
Chapter 10

GENERAL DISCUSSION

10.1. The Approach to Program Design

The approach to pattern recognition taken in this project stems from considerations of certain psychologically-oriented principles of the organism in its interaction with the environment. The discussion in Chapter 2 suggests how some of the principles might be embodied in a machine to recognize patterns. The relation between the designer's requirements for the machine, and the model incorporated into the machine, is mentioned therein. It is stressed that a detailed model is necessary in a machine which is required to process the input in a variety of ways. The need for an efficient coding scheme or schema to organize experience into a model is similarly stressed.

The environment for the programs in this project has been sequences of direction codes of line drawings. In specifying this input, it was envisaged that the overall problem for consideration would be simplified and yet the problem would retain the essence of the difficult task of recognizing hand-drawn line figures.

Two specifications, stemming from the author's desire to incorporate different levels of sophistication into the design, were originally formed for a machine to recognize the given sequences. In one case, the aim was to devise a machine which would incorporate a detailed model of the environment, and which, therefore, could be 'used' in many different ways. In the other case, the specification restricted the abilities of the machine, and consequently the aim was to produce in a
machine, having these restricted abilities, a suitable model which could be employed to achieve recognition of the sequences. Two series of programs, one for each case respectively, have been developed.

In both series, the basic functioning of each program, (see Figure 2.1), is the same - two important operations form the main framework for pattern recognition. The first of these is model construction in which the device forms a representation of its experience. This construction is achieved by coding patterns occurring within this input stream. Class models are formed for the members of a particular class specified in these programs, and this is performed independently of any other set. The second operation is a processing of these models, performed when an input sequence is presented. A description of the input is generated in terms of the 'known' patterns and a comparison is made between the input description and that for the 'average member' in the model. The comparison highlights the differences which exist between the models - with adequate training, models can be formed in which these differences correspond to the characteristics of the given classes.

10.2. The First Series of Programs and Their Extensions

The programs of the first series construct a detailed model (incorporating a 'curve' level of description) of the classes of figures in the input. Each stored class model represents an 'average member' of the classes, and is employed in generating a description (similar to the 'average member') of the input. A similarity measure based on certain factors such as shape, orientation and beginning-end vector, is evaluated for each generated description, and
forms the basis of decision of class membership. The programs have the ability to average over class members, by eliminating irregularities in line segments and curve types, and by forming subsets of figures with a common name.

Some 12 programs have been developed by modifying the basic method. Results from a 'final' program (AN6), have shown that AN6 is invariant to a variety of stretches and squeezes of a figure, and is, to a reasonable degree, size and orientation independent, (Section 5.3). The main problem for a user is the specification of learning samples, which are required to be 'typical' of the class.

Another program, AN3, has been modified to accept 3-element line segments as input, for use in an on-line environment. This work is described in Appendix D. Although the system is still developmental, initial results have shown that it can achieve satisfactory recognition rates, and it will be used as a basis for further experimenting and testing of the recognition program.

These results suggest that a form of the programs would be satisfactory in an interactive system, in which the user specifies his character set, and requires this set to be recognized. This program would consequently be an important procedure in the man-machine interaction.

Another suggestion for program extension is aimed at a higher level of interaction - the extension to a program which can describe (in some suitable form) the similarities and relations between given line drawings. At a simple level, the program might relate the various features of an input drawing to known types of figures - this involves
the presentation of the factors extracted by the similarity measure, (Section 6.3).

A modification of the basic method in these programs has led to a suggestion of a method for the efficient recognition of handwriting, (Appendix A). While this method has many limitations at present, it is anticipated that a suitable program could be written which would be acceptable to a particular user of a handwriting input medium (e.g. a stylus on a tablet).

During the course of the work reported, several problems have arisen in connection with the recognition method used in the programs, (Section 6.1). Extensions to overcome these problems have been mentioned; for example, the incorporation of more suitable methods for processing the sequences, (e.g. use of an initial 'wholistic' processing), the devising of more effective methods for extracting various factors (e.g. size, orientation) and the development of other means for extracting factors not already obtained by the similarity measure, (Section 6.1).

10.3. The Second Series of Programs and their Extensions

The programs in the second series describe the sequences in terms of patterns, formed from internal relations, and discovered in the environment. Consequently, these programs treat the input as consisting of abstract symbols, and do not describe the 'whole' sequence. The processing of an input is considered to be an interesting application of the 'coding approach' outlined in Chapter 2 - each sequence is coded with the patterns (in experience), and a measure is provided to evaluate the coding effectiveness of the patterns of each class. That class whose
patterns code the input the most effectively, receives the decision of input class membership. Some programs (EE and EF) have the additional ability to build an 'average' model of the environment.

From the tests given, the results of the four programs developed in this series indicate that discrimination of various figures can be achieved using only internal relations to form the patterns. The important achievement of the programs, it is suggested, is the manner in which the discrimination is attained - the incorporation of a model construction procedure, and a coding method to accentuate the differences between class models (and hence classes), in a given input, (see Section 10.1).

A concept learning experiment for human subjects was studied and used to provide suggestions for program design. Although no attempt was made to incorporate features of the subject's behaviour in the procedure, evidence seemed to confirm that patterns similar to those extracted by the programs, were, in fact, obtained by the subject. It is suggested by the author that an efficient (ideal) subject, given the sequences, should extract patterns present in the program's model. Consequently, there is a suggestion that the program could form the basis for a simulation of human behaviour in the particular concept learning task. Possible extensions for incorporating strategies and hypothesis testing are briefly discussed in Section 9.2.

Because of the restrictions present in the design specifications, the ability of this series of programs to recognize figures with many distortions, is rather weak, (Section 9.3).

Several problems involving the method used to recognize the sequences, have been outlined in Section 9.1.
It is anticipated that revision of these methods could lead to this 'coding method' being an important procedure for recognizing sequences, whose elements are structured by internal relations.

10.4. The Approach Applied to other Problem Areas

The approach incorporated in the programs of this project, has been found suitable in providing a framework for a machine to recognize patterns in line drawings. However, several extensions appear to be required for application to other kinds of environments. Some of the problems which cause modifications to be necessary are discussed below, with reference to patterns in sentences of a language, pictures, and board positions in a game such as chess.

The task for the programs discussed herein, has been greatly simplified by the inputting of the direction codes, which themselves form the basic primitives in the environment. In other cases, however, particularly in photographs, the extraction of such primitives is not at all well-defined.

The model, in the author's approach, consists of classes of patterns, interrelated (by internal relations) in an hierarchical structure. An important difference from this form is evident in other patterns, where external relations are involved. For example, the (meaningful) patterns in a sentence are not defined in terms of the repetition of identical letters - rather, the 'patterns' are sets of letters forming words and sentences which are grouped according to spelling and grammatical rules. In a similar way, the interesting parts of a board position depend on the 'attacking' advantage held by the player's pieces; which, in turn, is controlled by the (external)
Another difference from the suggested form of a model is in the hierarchical structure. Recent work by Chomsky (37) suggests that not all of the structure of grammatical sentences can be expressed hierarchically - transformation or conditional links between 'trees' are required. Some research on pictures has also incorporated this idea,(42).

Once a particular model is defined, then a procedure for generating a description (or parsing of an input) is required. As discovered in this input, the procedure necessitates an efficient search - patterns in the model are 'coded' into, and organize the input. Some associational techniques (or heuristics) possibly necessary when the search becomes more complex, have been mentioned for the author's program in Section 6.1. It also appears that the search for complex patterns, such as interesting positions on a chess board, requires to be more wholistic, thereby avoiding the sequential processing of individual elements, (48).

The models for line-drawn figures (in this project) consist of a set of 'average members', corresponding to one of the known classes. This form does not appear satisfactory when the nature of the relations in the model changes. For instance, it does not seem appropriate to define an 'average sentence', to which other sentences can be compared for grammatical form. Similarly, the notion of an 'average near-winning' position in chess with which other positions can be compared for an evaluation of the state of the game, appears unsatisfactory. Narasimhan (147), by condemning the classificatory approach to pattern recognition, has obviously been confronted with this 'comparison to averages' problem, in bubble chamber
photograph studies. A more profitable approach has been, in these complex situations (37,42), the specification of a set of rules (a grammar) defining well-formed members.

The author found it convenient, in devising a recognition program for line drawings, to eliminate some of the class possibilities by 'rules', but to retain other possibilities for which a 'similarity' measure is computed. The final decision is based on the various values of the measure for the classes. A similar approach, but one in which only a single description of the input is generated, is that of Grimsdale et al. (79). In the alternative approach of defining a grammar, the 'breaking' of rules in the input can be considered as a class exclusion method. This procedure is presumably (but not specifically mentioned in their paper) contained in the program devised by Knoke and Wiley, (109).

The programs developed in the present project have the ability to construct a model of the environment - in this sense, they have been said to have 'learned'. It must be pointed out, however, that the structure of the model is 'known' by the programs, so that a 'discovery' of the relationships and kinds of properties in the environment is not incorporated (in the programs). On the other hand, the program achieved more than simply the assigning of class names to the various categories.

The essential differences between this model construction approach and other forms of learning has been mentioned in Section 4.4. It is considered by the author, that the need for learning must depend on the practical requirements of the device. Obviously, if the class members are known and can be specified, there is no need
for machine learning. If, however, the categories are not known (as in the case of different user-specified sets of categories) then there exists the need for machine learning (as defined in the program).

The above discussion has attempted to outline some of the problems inherent in recognizing patterns different from those studied in this project. The approach outlined in Chapter 2 has suffered in its generality by the restriction of considering only line drawings - to become more general, a change in emphasis from internal to external relations is required, and consequently the economy in coding must be measured with respect to these external rules.

A major problem confronting pattern recognition research, and stressed by the present approach, is the explication of the patterns (and, hence the model) for a given environment and the specification of this model in a convenient form for the machine. It is felt that this has been the main contribution (for the patterns in line-drawings) of this project.
Chapter 11
CONCLUSIONS

At the beginning of this project, it was evident that few pattern recognition schemes had the power to describe (and recognize) a wide range of distortions in patterns. One of the main restrictions was the design of the receptor - commonly providing 'n' unrelated features for the input description. It appeared that an approach incorporating some generality in describing patterns, would be fruitful.

The author has taken the approach of considering the part pattern recognition plays in the life of an organism in its interaction with the environment. Several principles of the organism's behaviour have been explicated, and suggestions have been made for their incorporation in a recognition machine. The role of learning has been basic to this approach, and has been employed to organize regularities (in experience), into a model which can be used to recognize patterns in the input. These principles have suggested an overall framework, for the design of a recognition machine to achieve considerable generality in recognizing patterns.

Within the suggested framework, two series of programs to operate on number sequences, (being direction codes for line drawings), have been developed.

In the first series of programs, a detailed model of line figures, is constructed. Certain original features are incorporated into the design:

(a) a detailed model incorporating a curve level of description,

(b) an 'hypothesis testing' procedure for generating
a description of the input,
(c) a similarity (or confirmation) measure based on factors, (e.g. shape, orientation), (making the program, to a certain degree, size and orientation independent),
(d) supervised learning which removes irregularities in particular members by averaging over class members - an efficient process needing about 4 samples of each class.

The results from the final program, and preliminary investigations using an 'on-line' method of inputting the sequences, indicate a useful potential for employing the recognition scheme as a character recognizer in a man-machine interactive system. Another possible extension for the programs may be directed to the machine production of descriptions of similarities and differences between presented line-drawn figures.

A concept learning experiment, performed by the author on a (limited) number of human subjects, has suggested a design approach to the second series of programs. In this series, the programs have restricted abilities in extracting patterns from the sequences. The important features of these programs are:

(a) incorporation of the 'coding method', in which
   (i) patterns found in experience are coded into the input, and
   (ii) a measure of the reduction in coding of the inserted patterns of each known class is used as the basis for deciding input class membership.
(b) generation of patterns in the classes, incorporating an 'averaging' procedure.

Some programs have been able to achieve, on certain
tests, recognition rates comparable to those achieved by the first series of programs. However, because of the design restrictions, the programs in the second series are inferior, at character recognition, to those of the first series.

The procedure to achieve recognition involves model construction by the extraction of patterns from the input, and the processing of an input to compare the similarity between its generated patterns and those stored in the class models. The two series of programs have presented different approaches to the application of the outlined principles - in one case, the approach has been dependent on the designer's intended application for the device; in the other, the approach has been modified by constraints, imposed by the designer, on the abilities of the machine.

The framework for pattern recognition developed in this project has been primarily designed for recognizing patterns in line drawings. These patterns may differ from those encountered elsewhere, such as in language and pictures, and an extension of some notions in the approach is required for such areas to be incorporated into the present framework.

The subject of this project, line drawings, has provided a convenient vehicle for the study of patterns and for an investigation of the problems in pattern recognition.
REFERENCES AND SELECTED BIBLIOGRAPHY


6. F. ALT, Digital pattern recognition by moments, in (59); 153-180.

7. S. AMAREL, On the automatic formation of a computer program which represents a theory, in (222); 107-176.


14. H. BARLOW, Sensory mechanism, the reduction of redundancy and intelligence, in (148), 2; 535-560.


27. BRITISH COMPUTER SOCIETY, Character Recognition, 1967.


29. J. BRUNER, Going beyond the information given, in (43); 41-69.


40. M. CLOWES, The use of multiple auto-correlation in character recognition, in (59); 305-318.

41. M. CLOWES, Perception, picture processing and computers, in (44); 181-187.


52. M. EDEN and M. HALLE, The characterization of cursive script, in (35); 287-299.


58. J. FELDMAN, Simulation of behavior in the binary choice experiment, in (57); 329-346.


64. G. FREDKIN, Techniques using LISP for automatically discovering interesting relations in data, in (18); 108-124.


68. L. FRISHKOPF and L. HARMON, Machine reading of cursive script, in (35); 300-316.


78. L. Gregg, Internal representations of sequential concepts, in (108); 107-142.


82. A. de Groot, Perception and memory versus thought: some old ideas on recent findings, in (107); 19-50.


89. W. Highleyman, Linear decision functions with application to pattern recognition, in (59); 249-286.
90. E. HILGARD and G. BOWER, Theories of Learning, Appleton Century Crofts, N.Y., 1966, (3rd. ed.).


92. A. HORMANN, Programs for machine learning, II, Inf. and Control, 7, 1, 1964, 55-77.


100. F. JEANS, Generation and testing of hypotheses, Final Report, Rome Air Development Centre, Griffirs Air Base, N.Y., May 1962.


110. M. KOCHEN, An experimental study in hypothesis formation by computer, in (35); 377-403.


121. R. LEDLEY, Thousand-gate-computer simulation of a billion-gate computer, in (203); 457-480.


126. R. LINDSAY, Inferential memory as the basis of machines which understand natural language, in (57); 217-233.


129. D. MACKAY, The epistemological problem for automata, in (177); 235-251.

130. D. MACKAY, Information and learning, in Proc. of a Conf. on Learning Automata, Karlsruhe, Germany, 1961; 40-49.


137. G. MILLER, The magic number seven, plus or minus two ; some limits on our capacity for processing information, Psych. Rev., 63, 1956, 81-97.


151. A. NEWELL and H. SIMON, GPS, a program that simulates human thought, in (59); 279-293.

152. A. NEWELL, J. SHAW and H. SIMON, Empirical explorations with the logic theory machine; a case study in heuristics, in (59); 109-133.


159. M. PIVAR and M. FINKELSTEIN, Automation, using LISP, of inductive inference on sequences, in (18); 125-136.


165. L. ROBERTS, Graphical communication and control languages, in (191); 211-217.


173. O. SELFRIDGE, Pandemonium: a paradigm for learning, in (148); 511-526.


187. R. SOLOMONOFF, Training sequences for mechanized induction, in (222); 425-434.


198. N.S. SUTHERLAND, Stimulus analysing mechanisms, in (148); 578-603.


208. L. UHR and C. VOSSLER, A pattern recognition system that generates, evaluates and adjusts its own operators, in (59); 251-268.

209. L. UHR and C. VOSSLER, Simulation of perception and language, in (19); 228-266.


215. C. VOSSLER and L. UHR, A computer simulation of pattern perception and concept formation, in (202); 233-243.


218. B. WIDROW and F. SMITH, Pattern-recognizing control systems, in (203); 288-317.


The author is aware of the following papers but has been unable to procure a copy. Their titles indicate that they may be of interest to this project.


The above Proceedings are unavailable at the time of writing, and the authors, although requested, have not as yet sent copies of their papers.
APPENDIX A

A SUGGESTION FOR A PROGRAM TO RECOGNIZE HANDWRITING

A.1. Description of the Scheme

The scheme discussed here is a suggestion for a method for the 'on-line' recognition of handwriting, being a development of the first series of programs, (Chapters 4, 5, 6). The method has two parts:

(a) obtaining a description of the input in terms of curves and other associated features,
(b) reducing the generated description to a set of terminal strings, which can be matched to strings known to classes.

(a) has been programmed and (b) has been hand simulated. Examples of the procedures are given and the limitations of the approach mentioned.

The description (a) is a synthesis of the direction code elements (see Chapter 3), which form the input to the program. Three factors are extracted from the input:

(i) a set of curve types. The line segments contain 3-element codes. (These segments would be satisfactory as input to the program, see Section 5.2.2) The slope differences between consecutive line segments are coded into one of six possible curve types by a modified SIMP routine, (Section 5.2.6). There is an additional curve type, No.6, to the set in Table 5.2, representing very sharp corners.

(ii) the number of direction elements in the curve,
(iii) the direction of each straight line (No. 5, Table 5.2). This factor is simply the element
value which occurs most often for the set of elements in the curve.

Table A.1 gives values for these factors for an input sequence representing the word 'deal'. A program (designated DD) has been written to form these values from any given sequence of direction code elements.

The second part, (b) involves a procedure for:

(A) inserting the reduction rules into the input description, and
(B) matching the coded descriptions to the known terminal strings for each class.

This procedure can be performed sequentially; after each curve type formed, both (A) and (B) can be processed for the allowable classes.

The rules specify obligatory changes to the input description and are dependent on a particular class. However, some of them are common to a number of classes. Application of the rules depends on the values of the three factors in the input description. In some cases, certain conditions are placed on the length value which must be satisfied before the rule can be inserted. The terminal strings specify an allowable set of descriptive values for a given class.

A set of reduction rules and terminal strings derived by the author, for the letters a,d,b,f,e,l and two types of connections, C1 and C2, between these letters, are shown in Tables A.2 and A.3, respectively. These were produced from consideration of about 20 single letters, (Test 15, Table 3.1), and 20 words, (Test 16, Table 3.1), using the above letters - all being written by the author.

Some of these rules (e.g. 'B'(3), Table A.2) specify
INPUT:-
5566661616111121212223233344344444545454545454545454545
22122212212222223233434444445555555555611121212222
233344344444545445445444454555556111212122222223
33343 4444545455556111212122222222222222222222222
2333443444445454454454443323256611121212222334444545454545
5455221222333344454545454545454545454545454555611121212212212
122122222222323343444444

DESCRIPTION:-

A:  3  5  6  5  3  5  5  3  5  4  2  3  5  6  3  5  3  5  3  5  3
B:  36  24  3  21  6  12  12  6  15  9  9  3  21  9  3  15  21  12  21  6
C: -  5  -  2  -  4  -  2  -  4  -  -  -  5  -  -  4  -  2  -

(A: CURVE TYPE; B: LENGTH; C: DIRECTION OF TYPE 5).

TABLE A.1. INPUT AND DESCRIPTION OF WORD 'DEAL'.

TABLE A.2. THE SUGGESTED REDUCTION RULES.
\begin{table}[h]
\centering
\begin{tabular}{|c|c|}
\hline
B(1) & 31-- 3  \\
XY & Z  \\
\hline
A(1) & AS FOR B(1)  \\
\hline
D(1) & AS FOR B(1)  \\
\hline
F(1) & AS FOR B(1)  \\
\hline
E(1) & AS FOR B(1)  \\
\hline
\end{tabular}
\caption{TABLE A.2. THE SUGGESTED REDUCTION RULES.}
\end{table}
Table A.3. The suggested terminal strings.

<table>
<thead>
<tr>
<th>'B'</th>
<th>5</th>
<th>3</th>
<th>5</th>
<th>3</th>
<th>5</th>
<th>2</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>Y</td>
<td>-</td>
<td>4</td>
<td>31</td>
</tr>
<tr>
<td>415</td>
<td>2</td>
<td>-</td>
<td>5</td>
<td>3</td>
<td>14</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>'F'</th>
<th>5</th>
<th>3</th>
<th>3</th>
<th>3</th>
<th>5</th>
<th>2</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>Y</td>
<td>-</td>
<td>4</td>
<td>31</td>
</tr>
<tr>
<td>415</td>
<td>2</td>
<td>-</td>
<td>5</td>
<td>3</td>
<td>14</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>'E'</th>
<th>5</th>
<th>3</th>
<th>5</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>-</td>
<td>Y</td>
<td>-</td>
<td>4</td>
</tr>
<tr>
<td>415</td>
<td>1</td>
<td>2</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>'A'</th>
<th>5</th>
<th>3</th>
<th>5</th>
<th>6</th>
<th>5</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>-</td>
<td>Y</td>
<td>-</td>
<td>Z</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>4</td>
<td>5</td>
<td>2</td>
<td>3</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>'D'</th>
<th>5</th>
<th>3</th>
<th>5</th>
<th>6</th>
<th>5</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Y</td>
<td>-</td>
<td>Z</td>
<td>-</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>4</td>
<td>5</td>
<td>2</td>
<td>3</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>C1</th>
<th>4</th>
<th>6</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>5</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>C2</th>
<th>3</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>3</td>
<td>4</td>
</tr>
</tbody>
</table>

1: An alternative direction for type 5 curve
- : 'Don't care' or no value exists
Lines give curve type, length and direction (see 'B')
Alternative curve types are placed on the next line
a reduction on curve types which would not occur in the original description but which may occur in the reduced form. Actually none of the rules depend on the direction of the straight line but it is expected that this would occur if the number of letters was increased. These reductions are designed to eliminate 'noise' from an input description. For example, 'A'(4) (Table A.2) eliminates the presence of a straight line occurring in the curved section (enclosed between two type No.3 curves) of the character.

The terminal strings exhibit the general form of the class. When the strings for two curves are compared, the characteristics of the class become obvious. By comparing the 'e' and the 'l' one can see that the difference lies in the presence of a long straight line (the downstroke), which must occur in the 'l' but not in the 'e'. Note in the case of the 'e' the length of the straight line (y) may be zero, in which case that curve type is not expected.

A.2. An Example of the Operation of the Scheme

The application of the rules and terminal strings (of Tables A.2, A.3) on the input of Table A.1 is now to be given as an example. The tree-like structure showing the generation of discovered classes after each input curve type is shown in Table A.4. The discovered classes are enclosed (in boxes) in the structure, while the possible classes at any stage are shown beside the (tree) lines. Initially, any of the six letters are possible. Note that the connections themselves form classes and that they must follow a letter. Furthermore, only certain letters can follow a connection. An 'e' is discovered in the first curve type (No.3), because of
Table A.4. Example of the Search for Input Letters.
its match to the terminal string of class 'e'. However, the next curve type is a No.5 at a direction 5. No connection can either reduce this type or be matched as a terminal string to it—therefore this path is a 'dead-end', denoted by 'X'. Other dead-ends are shown; beside each 'X' are the possibilities which have been found not to exist.

The alternative branch following the first value shows that after the third curve type, only the letter 'd' can be matched. After the fifth value, this match is completed and the letter 'd' is shown enclosed. The next possibility is a connection; C2 is discovered. The final tree generated shows that the input is composed of the classes: 'd-C2-e-C1-a-C2-l'. Two reduction rules, shown by brackets (with their name from Table A.2) are applied in the tree generation.

A.3. Discussion

The rules and terminal strings have been applied to 20 different sequences representing words of four or five letters (e.g. 'deal', 'bleed', 'leaf', 'deaf', 'fled'). There were four errors (in letters) due to 'noisy' curve types with which the reduction rules could not cope. This amount of testing is inadequate; further development of the scheme would have been time-consuming in obtaining the direction codes. It is felt that its potential has been demonstrated, and full programming and further testing of this scheme is envisaged by employing the input system described in Appendix D.

Mention must be made of the constraints on the user of the scheme:

(a) the 'pen' must remain on the 'paper' during writing,
(b) the stroke sequence must be consistent,
(c) the size of the letters must remain approximately constant. The degree of constraint imposed by (b) and (c) will be known more accurately with further testing. At this stage, all of the above factors would not appear to constrain, unduly, a user who has had some practice.

Certain problems can be foreseen and must be overcome in the implementation to a practical scheme:

(i) There is no provision for the machine learning of the rules and terminal strings. The need for the design to pre-specify these is expected to be a severe limit on the generality of the program.

(ii) Obviously there is a relation between the rules and the strings - the amount of noise in the input determines the number of the rules and strings. For a given amount, there is a compromise between the generality of the rules and that of the strings. This must be determined for a more noisy input and more classes.

(iii) There is a need to incorporate context of the letters into the program, as a means for limiting their allowable combinations into words.

(iv) The experiment has been performed on large writing, (approximately 1"-2" high). It may be that when the user writes smaller letters, the curve types are less well-defined, causing more noise in the input sequence.

(v) It would appear necessary, with increased noise in the sequence, to specify a more flexible matching procedure, dependent, for example, on a majority of curves being present in a letter.

(vi) With more classes, the transform and match procedure would become more time-consuming. Some 'heuristics' may be necessary to make the process more efficient.
The scheme is not expected to be as tolerant to 'poor' writing as other suggestions, which have appeared in the literature—Mermelstein and Eden's scheme (136) is far more sophisticated in its stroke segmentation procedure; Earnest's program (50) has the ability to operate on a large number of words; Frishkopf's proposal (68) has the advantage of more accurate height determination but at the expense of more constraints on the user. It must be pointed out in conclusion, that the above is only a suggestion for further work and little success can be claimed as yet. Although the procedure is simple, the author is unaware of any similar attempts.
APPENDIX B

AN6: A PROGRAM WHICH LEARNS TO RECOGNIZE LINE-DRAWN CHARACTERS

AN6 is written in PL/1 for the IBM 360/50 at the A.N.U., Canberra. The program occupies about 59K bytes of storage.

AN6 constructs 'average' figures (AVS) from a set of given samples in the learning phase (L.P.). About 3 to 5 samples for each class type, are usually sufficient for this construction. Once these AVS have been formed, AN6 will recognize any further input sequences, in the working phase, (W.P.).

A typical time for processing each sequence in the W.P. is 3 seconds, but the time can vary from 1 to 5 seconds depending mainly on the length of the sequence and the number of similar AVS to the input.

This appendix outlines the input/output specifications for AN6, and presents details of the actual program with comments. Reference should be made to Chapter 5 of this thesis, where various parts of the program have been mentioned.

The following gives the input/output specifications in the program — the program statements should be referenced for the format details.

INPUT REQUIREMENTS.

11. Five values are initially required (with at least 1 space between each value).
   1. The number of learning samples, (lim).
   2. Control of the output, (see below), (prit=0,1,2).
   3. Tolerance for the pas-pd (=ut-pd) range, (tol).
   4. Limit on the number of AVS, (stolim).
   5. Limit on the total similarity below which a new subset will be formed for an incorrect decision, (gm).

12. The set of input sequences then follow:
   1. Direction code elements, (limit of 79).
   2. A '/ ' after the last element in the sequence.
   3. Name of the class in inverted commas, (e.g. '2').

OUTPUT SPECIFICATIONS.

The output can be controlled by the input variable 'prit'.

01. When prit=0, the following is outputted,
   1. A list of the 5 specified input values.
   2. The successive number of each input sequence, and the computer time code.
   3. The purged AVS.
   4. Size and total confirmation values for each similar AV.
   5. The input sequence and the computer time code.
   6. The decision (AV name) and whether correct or not.
   7. Notice that a new AV cannot be formed, because the
NUMBER EXCEEDS STORLIM; 'NO STORAGE' IS OUTPUTTED.

8. IN L.P., THE AV OF THE INPUT CLASS,

02. WHEN PRIT=1, THE OUTPUT FOR C1 PLUS,
  1. PROGRESSIVE CONFIRMATION VALUES FOR EACH AV (AFTER EACH CURVE); APPEARS WITH 01.3.
  2. FINAL CONFIRMATION VALUES (ORIENTATION, BEGINNING-END VECTOR AND SHAPE MEASURES); APPEARS WITH 01.4.
  3. IN L.P., THE STORED LINE SEGMENTS FOR THE CORRECT (OR NEW) AV; APPEARS BETWEEN 01.7 AND 01.8.

03. WHEN PRIT=2, THE OUTPUT FOR C1,02 PLUS,
  THE LINE SEGMENTS AND CORRESPONDENCES BETWEEN THE INPUT AND ALL SIMILAR AVS; APPEARS BETWEEN 01.5 AND 01.6

THE INPUT/OUTPUT STATEMENTS IN THE PROGRAM HAVE BEEN REFERENCED BY THE ABOVE CLASSIFICATION.

THE PROGRAM AN6

AN6: PROC OPTIONS (MAIN);

/* THE PROGRAM HAS BEEN SUBDIVIDED INTO 16 SECTIONS, (A-P).
1. A AND B CONTAIN INTRODUCTORY STATEMENTS.
2. THE MAIN PROCESSING OF AN INPUT ITERATES AROUND SECTIONS C, D, E, F, G AND H (DEPENDING ON THE VARIABLE 'Q').
3. THE FINAL CONFIRMATION VALUES ARE DETERMINED IN I.
4. THE DECISION IS MADE AND CHECKED IN J.
5. K CONTROLS THE LEARNING (OR WORKING) PHASE.
6. L, M, N, O AND P ARE SUBROUTINES CALLED IN THE PROGRAM.

THE FOLLOWING ABBREVIATIONS HAVE BEEN USED IN THE COMMENTS:
VAL(VALS)--VALUES, NO.--NUMBER, DIF--DIFFERENCE,
SEG(SEGS)--SEGMENT, CVE--CURVE, PRED--PREDICTION,
ORIENT--ORIENTATION, FIG--FIGURE, ELS--ELEMENTS,
CONF--CONFIRMATION, SEQ--SEQUENCE, VBLE--VARIABLE. */

/* A:
THE DECLARATIONS, ON-CONDITIONS AND INITIAL ASSIGNMENT OF VARIABLES. */

DCL 1 FIG(10), 2 CV(5) FIXED BIN(15),
  /*STORES THE CURVE NUMBERS FOR EACH FIGURE*/
  1 CV(40), 2 (APAT,VC(18),TP) FIXED BIN(15),
  /*STORES THE 1ST ORIENT, SLOPE DIF VALUES AND CURVE TYPE FOR EACH KNOWN CURVE*/
  OD(10,5), /*BEGINNING-END ANGLE FOR EACH CVE OF AV*/
  XM(10), /*THE NO. OF CVES FOR EACH FIG*/
  ZM(40), /*THE NO. OF SLOPE DIFS FOR EACH CVE*/
  X(10), /*VBLES FOR THE ABOVE FACTORS*/
  Y(10), /*THE CURRENT VAL OF THE CVE NO.*/
  WM /*THE TOTAL NO. OF AVS AND VBLE FOR EACH AV*/,
  KM(10), /*NO. OF LINE SEGS IN THE INPUT*/
  CT(10), /*NO. OF DIRN ELSE INPUTTED SINCE LAST SEC*/
  APRED(10), /*THE PREDICTED ORIENT VAL*/
  PD(10), /*THE PREDICTED ORIENT VAL AND TOLERANCE VALS*/
  A1(10), A2(10), /*ORIENT VALS FOR CURRENT AND
**PREVIOUS LINE SEGMENTS.**

DL(10), /*NO. OF ELS IN CURRENT LINE SEG.*/
AS(10,20), C(10,20), U(10,20), /* THE SLOPE DIF, CONF.
AND NO. OF PREDs FOR EACH LINE SEG IN THE INPUT. */
NV(10,5), TV(10,5), /* THE NO. OF MATCHES AND TOTAL
LINE SEGs FOR EACH CVE.*/
Q(10), /*A VBLG GIVING THE STATE OF THE
PROCESSING OF THE INPUT FOR EACH AV. */
OO(10), /*THE NO. OF SAMPLES FOR EACH AV.*/
F(10,25), /*THE SLOPE DIF VALS FOR EACH AV.*/
F(*,1) ARE THE ORIENT VALS OF THE 1ST LINE SEG. */
JM(10), /*THE NO. OF VALS IN F.*/
T1,T2,AF,XX,YX) FIXED BIN(15), /*OTHER VBLES.*/
(FQ(10,25), /*FREQUENCY OF EACH SEG IN THE STORE. */
D(10,5), /*BEGINNING-LENGTH LENGTH OF EACH CVE.*/
DX(10,5), DY(10,5), /*BEGINNING-LENGTH LENGTHS FOR EACH
CVE IN THE INPUT.*/
CF(10,5), CL(10,5), CO(10,5), /*THE SHAPE, LENGTH AND ORIENT CONF VALS. */
TOT(10), /*TOTAL CONF FOR THE AVS. */
(P(10,25), /*LINE SEG ELS.*/
V,PS(10,2,5)) PACKED CHAR(5), /*VBLES FOR EACH P.*/
(A(80), */INPUT SEQUENCE OF DIRECTION ELS. */
CH(10), /*NAME OF EACH AV.*/
COR) CHAR(1), /*DECISION OF CLASS MEMBERSHIP.*/
(B1,B2,B3,B4) BIT(1); /*BITS USED IN COMPARISON. */

ON ENDOFILE (SYSIN) GOTO FINAL;
ON ERROR GOTO FINAL;
FIG=C; CV=0; YMWM,VAL=O; CH='.'; F,FQ=O;
/*I1*/ GET LIST(LIM,PRIT,TOL,STORLIM,GM); PUT PAGE;
/*I0.1*/ PUT LIST (LIM,PRIT,TOL,STORLIM,GM);

/* B:
START OF ITERATION FOR EACH INPUT SEQUENCE. */

START:
A1, A2, CT, X, Y, Z, KM, APRED, PD, TCT=O; AS, C, U=O;
P='  A='; t; C1, CL, CF, DX, DY, OR, TV, NV=O;
PAS, FUT, KK, I, N1, COM=O; VAL=VAL+1; Q=I; PS='00000';
/*I1.2*/ PUT SKIP(3) LIST('INPUT NO',VAL,TIME);

/* C:
THE PRED OF THE NEXT ORIENT VAL FOR EACH FIG WITH Q<3.
PREDs ARE MADE FOR THE FIRST VAL (X=O), WHEN THE AVS ARE
FINISHED (C>5) AND FOR THE NEXT LINE SEG ORIENT. */

PIN: W=I;
PRED: IF Q(I)<3 THEN DO;
    IF WM=O THEN GOTO ADON;
    IF X(W)=O THEN DO; Y(W)=FIG(W).CV(I); X(W)=I;
    APRED(W)=CV(Y(W)).APAT;
    END;
ELSE
    IF C(W,KM(W))>5 THEN DO;
    IF KM(W)>1 THEN APRED(W)=2*A(I(W))-A2(W);
    ELSE APRED(W)=A(I(W));
    IF Q(W)=2 THEN C(W,KM(W))=7; ELSE C(W,KM(W)+1)=7;
    ADON: IF KM(W)>1 THEN APRED(W)=2*A(I(W))-A2(W);
    ELSE APRED(W)=A(I(W));
    IF Q(W)=2 THEN C(W,KM(W))=7; ELSE C(W,KM(W)+1)=7;
    END;

ON ENDOFILE (SYSIN) GOTO FINAL;
ON ERROR GOTO FINAL;
FIG=C; CV=0; YMWM,VAL=O; CH='.'; F,FQ=O;
/*I1*/ GET LIST(LIM,PRIT,TOL,STORLIM,GM); PUT PAGE;
/*I0.1*/ PUT LIST (LIM,PRIT,TOL,STORLIM,GM);

/* B:
START OF ITERATION FOR EACH INPUT SEQUENCE. */

START:
A1, A2, CT, X, Y, Z, KM, APRED, PD, TCT=O; AS, C, U=O;
P='  A='; t; C1, CL, CF, DX, DY, OR, TV, NV=O;
PAS, FUT, KK, I, N1, COM=O; VAL=VAL+1; Q=I; PS='00000';
/*I1.2*/ PUT SKIP(3) LIST('INPUT NO',VAL,TIME);

/* C:
THE PRED OF THE NEXT ORIENT VAL FOR EACH FIG WITH Q<3.
PREDs ARE MADE FOR THE FIRST VAL (X=O), WHEN THE AVS ARE
FINISHED (C>5) AND FOR THE NEXT LINE SEG ORIENT. */

PIN: W=I;
PRED: IF Q(I)<3 THEN DO;
    IF WM=O THEN GOTO ADON;
    IF X(W)=O THEN DO; Y(W)=FIG(W).CV(I); X(W)=I;
    APRED(W)=CV(Y(W)).APAT;
    END;
ELSE
    IF C(W,KM(W))>5 THEN DO;
    IF KM(W)>1 THEN APRED(W)=2*A(I(W))-A2(W);
    ELSE APRED(W)=A(I(W));
    IF Q(W)=2 THEN C(W,KM(W))=7; ELSE C(W,KM(W)+1)=7;
    ADON: IF KM(W)>1 THEN APRED(W)=2*A(I(W))-A2(W);
    ELSE APRED(W)=A(I(W));
    IF Q(W)=2 THEN C(W,KM(W))=7; ELSE C(W,KM(W)+1)=7;
    END;

ON ENDOFILE (SYSIN) GOTO FINAL;
ON ERROR GOTO FINAL;
FIG=C; CV=0; YMWM,VAL=O; CH='.'; F,FQ=O;
/*I1*/ GET LIST(LIM,PRIT,TOL,STORLIM,GM); PUT PAGE;
/*I0.1*/ PUT LIST (LIM,PRIT,TOL,STORLIM,GM);

/* B:
START OF ITERATION FOR EACH INPUT SEQUENCE. */

START:
A1, A2, CT, X, Y, Z, KM, APRED, PD, TCT=O; AS, C, U=O;
P='  A='; t; C1, CL, CF, DX, DY, OR, TV, NV=O;
PAS, FUT, KK, I, N1, COM=O; VAL=VAL+1; Q=I; PS='00000';
/*I1.2*/ PUT SKIP(3) LIST('INPUT NO',VAL,TIME);

/* C:
THE PRED OF THE NEXT ORIENT VAL FOR EACH FIG WITH Q<3.
PREDs ARE MADE FOR THE FIRST VAL (X=O), WHEN THE AVS ARE
FINISHED (C>5) AND FOR THE NEXT LINE SEG ORIENT. */

PIN: W=I;
PRED: IF Q(I)<3 THEN DO;
    IF WM=O THEN GOTO ADON;
    IF X(W)=O THEN DO; Y(W)=FIG(W).CV(I); X(W)=I;
    APRED(W)=CV(Y(W)).APAT;
    END;
ELSE
    IF C(W,KM(W))>5 THEN DO;
    IF KM(W)>1 THEN APRED(W)=2*A(I(W))-A2(W);
    ELSE APRED(W)=A(I(W));
    IF Q(W)=2 THEN C(W,KM(W))=7; ELSE C(W,KM(W)+1)=7;
    ADON: IF KM(W)>1 THEN APRED(W)=2*A(I(W))-A2(W);
    ELSE APRED(W)=A(I(W));
    IF Q(W)=2 THEN C(W,KM(W))=7; ELSE C(W,KM(W)+1)=7;
    END;
END;    
ELSE DO;      
Z(W)=Z(W)+1;    
IF Z(W)>ZM(Y(W)) THEN DO;    
IF X(W)=XM(W) THEN GOTO ADDON;    
X(W)=X(W)+1;    
Y(W)=FIG(W).CV(X(W));    
Z(W)=1;    
PD(W)=CV(Y(W)).VC(1);END;    
ELSE PD(W)=PD(W)+CV(Y(W)).VC(Z(W));    
APRED(W)=A1(W)+CV(Y(W)).VC(Z(W));    
IF Q(W)=2 THEN U(W,KM(W))=U(W,KM(W))+1;    
ELSE U(W,KM(W)+1)=1;END;    
IF Q(W)=2 THEN Q(W)=4; ELSE Q(W)=3;    
call norm(APRED(W),KK);END;    
w=W+1; IF w<=wm THEN GOTO PRED;

/* D: */    
IF ANY AV HAS FORMED ANOTHER PRED (Q=2) TO COMPARE TO THE CURRENT INPUT SEG THEN PROCESSING MOVES TO PART G.    
DO W=1 TO WM; IF Q(W)=4 THEN GOTO CSIN; END;

/* E: */    
ANOTHER INPUT VAL IS OBTAINED. BLANKS BETWEEN SEQS ARE IGNORED. THE CT ARRAY IS UPDATED.

/* I2.1 AND I2.2 */    
INP: I=I+1; GET EDIT(A(I))(A(I));    
IF A(I)=" " THEN DO; I=0; GOTO INP; END;    
CT=CT+1; IF I=1 THEN GOTO INP;

/* F: */    
IF ANOTHER LINE SEG IS REQUIRED (Q=3) THEN THE CURRENT NO. OF ELS SINCE THE PREVIOUS SEG (CT) ARE TESTED FOR FORMATION INTO A POSSIBLE SEG. THIS PROCEDURE IS ITERATED AFTER EACH NEW INPUT UNTIL THE CLOSEST LINE SEG TO THE PRED IS FORMED; THEN Q=4. THERE ARE 2 PARTS TO THIS PROCEDURE, (F1 AND F2)

INPUT: IF Q(W)=3 THEN    
/* F1: */ SPECIAL PROCESSING TO FORM AN ALLOWABLE SEG OCCURS WHEN THE DIF BETWEEN ELS IS 2,3 OR 4, OR WHEN THE END OF THE INPUT SEG IS REACHED.

IF A(I)=" " THEN IF CT(W)<=2 THEN Q(W)=9;    
ELSE DO;    
ALS2:    
K=CT(W)-1;    
A2(W)=A1(W);    
IF PS(W,K)="00000" THEN    
IF K=2 THEN PS(W,K)=A(I-1)||A(I-1)||"000";    
ELSE PS(W,K)=A(I-1)||A(I-1)||"00";    
call FINDANG(PS(W,K),A1(W),DL(W)); CT(W)=1;    
goto ALSd;    
ELSE IF CT(W)>1 THEN DO;    
K=ABS(A(I)-A(I-1));    
IF K=2 | K=3 | K=4 THEN IF CT(W)=2 THEN CT(W)=1;
ELSE GOTO ALS2;
/* F2: IF NONE OF THE F1 CONDITIONS HOLD, THEN THE ROUTINE 
PATGEN IS CALLED TO FORM AN ALLOWABLE LINE SEG. */

ELSE DO;
CALL PATGEN(PS(W,CT(W)),CT(W));
IF CT(W)=NSOP THEN DO; A2(W)=A1(W);
DO M=2 BY 1 WHILE (PS(W,M)="00000"); END;
IF PS(W,NSOP)="00000" THEN N=NSOP-1; ELSE N=NSOP;
DO K=M TO N;
CALL FINDANG(PS(W,K),N1,L);
CALL NORM(N1,APRED(W));
IF K=M|NO>ABS(N1-APRED(W)) THEN DO;
NO=ABS(N1-APRED(W)); DL(W)=K; A1(W)=N1; END;
END;
CT(W)=NSOP-DL(W);

ALSO:
K(W)=KM(W)+1; P(W,KM(W))=PS(W,DL(W));
PS(W,*)="00000"; Q(W)=4;
IF CT(W)>1 THEN CALL PATGEN(PS(W,CT(W)),CT(W));
END; END; END;

W=W+1; IF W<=WM THEN GOTO INPUT;

G:
THE COMPARISON BETWEEN THE INPUT AND EACH AV WITH Q=4.
THIS PART IS DIVIDED INTO 5 SECTIONS, (G1-G5).

CSIN: W=1;
CSSC: IF Q(W)=4|Q(W)=9 THEN DO;
*/

G1: TESTS TO FIND WHETHER A COMPARISON IS POSSIBLE.

/* G1: TESTS TO FIND WHETHER A COMPARISON IS POSSIBLE. */

IF Q(W)=9&WM>0 THEN GOTO CFIN; Q(W)=1;
IF KM(W)=1 THEN DO;
A(W,F)=A1(W); IF WM=0 THEN GOTO PIN;
DX(W,F)=DL(W)*COS(A1(W));
DY(W,F)=DL(W)*SIND(A1(W));
GOTO OUT; END;
AS(W,KM(W))=AF=A1(W)-A2(W); CALL NORM(AF,P,D(W));
CALL NORM(AS(W,KM(W)),K);
IF WM=0 THEN IF A(I)=0 THEN GOTO PUT3;
ELSE GOTO PIN;
*/

G2: PAS-FUT RANGE IS CALCULATED.

/* G2: PAS-FUT RANGE IS CALCULATED. */

IF X(W)>1 THEN T2=CV(Y(W)-1).TP; ELSE T2=0;
IF X(W)<KM(W) THEN TO=CV(Y(W)+1).TP; ELSE TO=0;
T1=CV(Y(W)).TP;
IF C(W,KM(W))<6 THEN DO;
IF Z(W)=ZM(Y(W)) THEN K=CV(Y(W)).VC(Z(W));
ELSE K=CV(Y(W)).VC(Z(W)+1);
IF ABS(PD(W))>TOL THEN PAS=PD(W)/2;
ELSE
IF T1=3|(T1>=4&P(W)>=0) THEN PAS=PD(W)-TOL/2;
ELSE PAS=PD(W)+TOL/2;
IF ABS(K)>TOL THEN FUT=PD(W)+K/2;
ELSE
IF T1=3|(T1>=4&P(W)>=0) THEN FUT=PD(W)+TOL/2;
ELSE FUT=PD(W)-TOL/2;
G3: THE COMPARISON TO DETERMINE WHERE THE INPUT LINE SEG LIES IN RELATION TO THE RANGE, AND CORRESPONDENCE TO THE AV AV ON THE SEG AND CVE LEVELS.

IF AF*PD(W)<0 THEN
  IF (PAS-AF)*(AF-FUT)<0 THEN Q(W)=4; ELSE;
    IF (PD(W)=0) & (T1=5) THEN GOTO SS2;
    ELSE
      IF (PAS-AF)*(PAS-PD(W))<0 THEN Q(W)=3;
      ELSE
        IF (FUT-AF)*(FUT-PD(W))<0 THEN Q(W)=2;
      END;
    END;
  END;
ELSE
  IF Q(W)=2 THEN
    IF T1=1 THEN
      IF (T2=2) & T2=4 & AF>180 THEN Q(W)=5; ELSE Q(W)=1;
      ELSE
        IF T1=2 THEN
          IF (T2=3) & AF<-180 THEN Q(W)=5; ELSE Q(W)=1;
          ELSE
            IF T1=5 THEN IF Z(W)=1 THEN
              IF ((T2=1) & T2=3) & AF>0 THEN Q(W)=5; ELSE;
              ELSE IF T1=2 THEN IF (T2=4) & AF<31 THEN Q(W)=5;
                ELSE
                  IF T1=1 THEN IF B2='01'B THEN Q(W)=5;
                    ELSE
                      IF B1='01'B THEN Q(W)=5;
                        ELSE
                          IF B2='01'B THEN Q(W)=5;
                            ELSE
                              IF PD(W)<0 THEN Q(W)=2;
                                ELSE IF Z(W)=1 THEN
                                  IF B2='01'B THEN Q(W)=5;
                                    ELSE
                                      IF PD(W)>0 THEN Q(W)=2;
                                        ELSE IF B1='01'B THEN Q(W)=5;
                                          ELSE
                                            IF B3='01'B THEN Q(W)=2;
                                              ELSE
                                                IF PD(W)>0 THEN Q(W)=2;
                                                  ELSE IF B3='01'B THEN Q(W)=2;
                                                    ELSE GOTO PG2;
                                                      END;
                                    END;
                                  END;
                                END;
                              END;
                            END;
                          END;
                        END;
                    END;
                  END;
                END;
              END;
            END;
          END;
        END;
      END;
    END;
  END;
END; END;

PG1:
  IF AF<-30 THEN C(W,KM(W))=6; ELSE Q(W)=3;
  ELSE IF B4='01'B THEN Q(W)=2; ELSE GOTO PG1;
  ELSE
    IF T1=4 THEN IF PD(W)>0 THEN Q(W)=2;
      ELSE IF Z(W)=1 THEN
        IF B1='01'B THEN Q(W)=5;
          ELSE
            IF B3='01'B THEN Q(W)=2;
              ELSE IF Z(W)=1 THEN
                IF ((T2=1) & T2=3) & AF>0 THEN Q(W)=5;
                  ELSE Q(W)=2;
                    ELSE Q(W)=2;
                      END;
                END;
              END;
            END;
          END;
        END;
      END;
    END;
END;

PG2:
  IF AF>30 THEN C(W,KM(W))=6; ELSE Q(W)=3;
  ELSE IF B3='01'B THEN Q(W)=2; ELSE GOTO PG2;
  ELSE IF Z(W)=1 THEN
    IF ((T2=1) & T2=3) & AF>0 THEN Q(W)=5;
      ELSE Q(W)=2;
        ELSE Q(W)=2;
          END;
    END;
  END;
ELSE
IF ((T1=1)\&\&(T1=3)\&\&(AF<-31)|((T1=2)\&\&(T1=4)\&\&(AF>31))\&\&(T1=5)\&\&ABS(AF)>30) THEN C(W,KM(W))=6;

/* G4: DETERMINATION OF CONF VALS. */
IF C(W,KM(W))<6 THEN C(W,KM(W))=Q(W); ELSE Q(W)=1;
CFIN: IF Q(W)=5 THEN DO: XX=X(W)-1; YX=FIG(W)*CV(XX); END;
ELSE DO: XX=X(W); YX=Y(W); END;
IF C(W,KM(W))<6\&\&Q(W)=1 THEN NV(W,XX)=NV(W,XX)+1;
IF Q(W)=2\&\&Q(W)\geq 9 THEN DO:
DX(W,XX)=DX(W,XX)+DL(W)*COSD(A1(W));
DY(W,XX)=DY(W,XX)+DL(W)*SIND(A1(W));
TV(W,XX)=TV(W,XX)+1; END;
IF A1(I)=1\&\&Q(W)=1 THEN Q(W)=7;
IF Q(W)=5\&\&Z(W)=ZM(Y(W)) \&\&Q(W)\geq 7 THEN DO:
IF TV(W,XX)>0 THEN
CF(W,XX)=1/ZM(YX)/TV(W,XX)*NV(W,XX)*NV(W,XX);
IF CF(W,XX)=0\&\&I=5\&\&XX<XM(W) \&\&Q(W)=2
THEN C(W,KM(W))=6;

/*02.1*/
IF PRIT>0
PUT SKIP LIST("CF",W,XX,CF(W,XX),TV(W,XX));
END;
IF Q(W)=5\&\&Z(W)=1\&\&C(W,KM(W)-1)\&\&Q(W)=3 THEN
OR(W,XX)=A2(W)-CV(YX).APAT;

/* G5: SET Q (AND PD) FOR EITHER ANOTHER PRED, ANOTHER LINE SEG FROM THE INPUT, OR BOTH. */
IF Q(W)=3 THEN PD(W)=PD(W)-AF;
ELSE IF Q(W)=2\&\&Q(W)\geq 5 THEN PD(W)=0;
IF Q(W)=5 THEN Q(W)=3;
END;
ELSE IF Q(W)=0 THEN GOTO INP;
OUT: W=W+1; IF W<WM THEN GOTO CSSC;

/* H:
FIND ALL PURGED AVS, (C=6); SET THEIR Q=6. */
M,N=0;
DO W=1 TO WM; IF C(W,KM(W))=6 \&\&Q(W)=6 THEN N=N+1; END;
IF N<WM THEN DO;
DO W=1 TO WM;
IF C(W,KM(W))=6 \&\&Q(W)=6 THEN
DO; Q(W)=6;
/*01.3*/
PUT SKIP LIST("P",W,KM(W),X(W),AS(W,KM(W)));
END; END; END;

/* L:
DETERMINES IF PROCESSING HAS FINISHED (INCLUDING ALL PREDs). IF SO, THEN FINAL CONF VALS ARE CALCULATED FOR EACH FIGURE, AND THE MAXIMUM (COM) OF THESE FIGURE VALUES IS DETERMINED.
IF NOT, THEN CONTROL RETURNS TO PART C. */
DO W=1 TO WM; IF Q(W)=7 \&\&Q(W)=6 THEN M=M+1; END;
IF M=WM THEN DO;
DO W=1 TO WM;
IF Q(W)\(!=6 \text{ THEN DO;)}
SUM,K,N1=0;
DO N=1 TO X(W);
L=FIG(W),CV(N);  K=K+ZM(L);
CALL NORM(OR(W,N),KK);
END;  SIZE=(KM(W)-1)/K;
/
01.4/*
PUT SKIP DATA (W,SIZE);
DO N=1 TO X(W);
IF SIZE>1 THEN CF(W,N)=CF(W,N)*SIZE;
ELSE CF(W,N)=CF(W,N)/SIZE;
IF CF(W,N)\(!1 \text{ THEN CF(W,N)=1/CF(W,N);)}
IF DX(W,N)=0 THEN GOTO ORT;
TO=ABS(D(W,N)-SQR(T(DX(W,N)*DX(W,N)+DY(W,N)*DY(W,N)))/
IF TO<1 THEN TO=1;
IF D(W,N)>4 THEN DO;
IF DX(W,N)<0 \text{ THEN NO}=180+ATAND(DY(W,N)/DX(W,N));
ELSE IF DX(W,N)>0 \text{ THEN NO}=90; \text{ ELSE NO}=-90;
IF PRIT=2 THEN PUT SKIP LIST(D(W,N),TO,NO);
NO=NO-90; \text{ OR} (D(W,N)=0, CALL NORM(NO, KK);
IF ABS(NO)>90 \text{ THEN CL(W,N)=0;)
ELSE CL(W,N)=SQR(T(COSD(NO)/TO));
END;
ELSE IF TO<3 \text{ THEN CL(W,N)=1;)
ELSE CL(W,N)=1/TO;
ORT:
CO(W,N)=1-ABS(OR(W,N))/180;
END;
DO N=1 TO XM(W);
TOT(W)=(CO(W,N)+CF(W,N)+CL(W,N))/3+TOT(W);
END;  TOT(W)=TOT(W)/XM(W);
/
02.2/*
IF PRIT>0 \text{ THEN PUT SKIP EDIT)((CO(W,N),CL(W,N),CF(W,N)
DO N=1 TO X(W))/(3 (E(12,4,5),X(9)),SKIP);
/
01.4/*
PUT LIST(TOT(W));
/
01.5/*
PUT3:
PUT SKIP EDIT (A,TIME)(80 A(1), F(10));
/
03/*
IF PRIT>1 \text{ THEN DO J=1 TO WM;)
IF Q(J)\(!=6 \text{ THEN}
PUT SKI P(2) EDIT((P(J,K),AS(J,K),C(J,K),U(J,K)
DO K=1 TO KM(J))(X(3),A(5),3 F(10),SKIP);
/
J:
THE DECISION IS MADE AND CHECKED WITH THE READ-IN NAME.  */
/
01.6/*
PUT SKIP EDIT( 'THE INPUT FIGURE IS A ',CH(W))
(A(22),A(11));
/
I2.3/*
GET LIST (COR);
/
K:
IN L.P. \text{ THEN EITHER A NEW AV IS FORMED (UNLESS THE NO. EQUALS}\n\text{STORLIM), OR THE INPUT IS AVERAGED WITH THE CORRECT AV. \text{ IN}\nW.P. CONTROL REVERTS TO START TO PROCESS ANOTHER SEQ.}  */
IF CH(W)=COR THEN DO;
/*01.6*/
PUT LIST('','CORRECT DECISION',COR);
IF VAL>LIM|TOT(W)<GM|X(W)<XM(W) THEN GOTO START;
PUT2:
CALL FEED(AS(W,*),C(W,*),U(W,*),QQ(W));
END;
/*01.6*/
ELSE DO;
/*01.7*/
IF CH(W)=INCORR DEC THEN DO;
PUT LIST('','INCORR DEC',COR);
IF VAL>LIM THEN GOTO START;
COM=GM;
DO K=1 TO WM;
IF CH(K)=COR THEN 
IF TOT(K)>COM&X(K)=XM(K) THEN DO; W=K; COM=TOT(K); 
END; END;
IF COM>GM THEN GOTO PUT2;
/* L: 
ROUTINE FOR FORMING THE ALLOWABLE TYPES OF LINE SEG. THIS 
PARTICULAR ONE FORMS THE EXTENDED SET, (SEE SECTION 5.2.2). */
PATGEN: PROC (X,N);
DCL X CHAR (5),N FIXED BIN (15);
X='000000'; NSOP=0;
IF A(I)=A(I-1) THEN IF N>2 
THEN IF A(I)=A(I-2) THEN IF N>3 
THEN IF A(I)=A(I-3) THEN IF N>4 
THEN IF A(I)=A(I-4) THEN NSOP=N; ELSE DO; NSOP=N;
ELSE X=A(I-4) || A(I-3) || A(I-2) || A(I-1) || A(I) || '0'; 
ELSE IF N=4 THEN X=A(I-3) || A(I-2) || A(I-1) || A(I) || '0'; 
ELSE IF A(I)=A(I-4) THEN DO; NSOP=N;
ELSE X=A(I-4) || A(I-3) || A(I-2) || A(I-1) || A(I); 
END;
ELSE NSOP=N;
ELSE;
ELSE IF N>3 
THEN IF A(I)=A(I-3) THEN IF N>4 
THEN IF A(I)=A(I-4) THEN DO; NSOP=N; 
ELSE X=A(I-4) || A(I-3) || A(I-2) || A(I-1) || A(I); 
END;
ELSE NSOP=N;
ELSE X=A(I-2) || A(I-1) || A(I) || '00';
ELSE;
*/
ELSE IF N>2
  THEN IF A(I)=A(I-2)
  THEN IF N>3
  THEN IF A(I)=A(I-3)
  THEN IF N>4
    THEN IF A(I)=A(I-4) THEN DO; NSOP=N;
        X=A(I-4)||A(I-3)||A(I-2)||A(I-1)||A(I);
      END;
      ELSE NSOP=N;
      ELSE X=A(I-3)||A(I-2)||A(I-1)||A(I)||'0';
    ELSE NSOP=N;
    ELSE X=A(I-2)||A(I-1)||A(I)||'00';
  ELSE IF A(I-1)=A(I-2)
    THEN IF N>3
      THEN IF A(I-1)=A(I-3)
        THEN IF N>4
          THEN IF A(I-1)=A(I-4) THEN DO; NSOP=N;
              X=A(I-4)||A(I-3)||A(I-2)||A(I-1)||A(I);
          END;
          ELSE NSOP=N;
          ELSE X=A(I-3)||A(I-2)||A(I-1)||A(I)||'0';
        ELSE NSOP=N;
        ELSE X=A(I-2)||A(I-1)||A(I)||'00';
      ELSE NSOP=N;
    ELSE X=A(I-1)||A(I)||'000';
END PATGEN;

/*/ M:
ROUTINE FOR NORMALISING A GIVEN VAL TO ANOTHER ORIENT VAL. */

NORM: PROC(M,N);
  DCL (M,N) FIXED BIN (15);
  IF M-N>180 THEN M=M-360;
  ELSE
    IF M-N<-180 THEN M=M+360;
  END NORM;

/*/ N:
ROUTINE FOR CALCULATING THE ORIENT VALS FROM THE LINE SEG
CHARACTER STRING. THE ROUTINE IS AN APPROXIMATION. */

FINDANG: PROC(V,AV,M);
  DCL V Packed CHAR(5),EV(5) Packed CHAR(1) DEF V,
     (AV,K,M,N) FIXED BIN (15);
  AV=0;
  DO M=1 TO 5 WHILE (EV(M)='0');
    N=(BIN(EV(M))-1)*60; K=AV/M;
    CALL NORM(N,K); AV=AV+N;
  END;
  IF EV(5)='0' THEN M=M-1; ELSE M=5;
  AV=AV/M; CALL NORM(AV,KK);
END;

/*/ O:
ROUTINE FOR AVERAGING INPUT LINE SEG (ORIENT VALS) WITH THE
CORRESPONDING AV VALS. THE CORRESPONDENCE IS KEPT IN C AND U.
WHEN COMPLETED THE OLD CVE VALS ARE REMOVED FROM THE FIG AND
CV STRUCTURES.

FEED: PROC (AR, CD, UD, Q);
DCL (AR(20), CD(20), UD(20), Q, Y) FIXED BIN (15);
K, JN, JP, JF = 1; GOTO AV2;

START: K=K+1;
DO WHILE (UD(K) > 0);
   LV = 0;
   DO N = JF+1 BY 1 WHILE (FQ(W, N) < 0.49); LV = LV + 1; END;
   UD(K) = UD(K) - 1; JF = JF + LV + 1;
END;
IF CD(K) = 7 THEN GOTO AGAIN;
IF CD(K) = 1 THEN DO; V3N = JF;

AV: DO J = JP+1 TO JN-1; FQ(W, J) = Q*FQ(W, J)/(Q+1); END;

AV2: CALL NORM(W, JN) FQ(W, K) / (Q+1);
   JN = JF - 1;
IF CD(K) = 3 THEN IF LV = 1 THEN
   IF F(W, JN) *AR(K) < 0 THEN GOTO STORE;
   IF CD(K) = 5 THEN IF LV = 1 THEN
   DO; JN = JN - 1; GOTO STORE; END;
   IF LV > 0 THEN GOTO AV;
END FEED;

STORE: DO J = JP+1 TO JN; FQ(W, J) = Q*FQ(W, J)/(Q+1); END;
   JN = JM(W) + 1; JN = JN + 1;
   DO J = JM(W) - 1 BY -1 TO JN;
   F(W, J+1) = F(W, J); FQ(W, J+1) = FQ(W, J);
   END;
   F(W, JN) = AR(K); FQ(W, JN) = 1/(Q+1); JF = JF + 1; JP = JN;
AGAIN: IF CD(K) = 3 & CD(K+1) = 3 | CD(K) = 5 & CD(K+1) = 5
   THEN DO; K = K+1; GOTO AGAIN; END;
   IF JN = JM(W) & K <= KM(W) THEN DO;
   JN = JM(W) = JM(W) + 1; K = KM(W); GOTO AV2; END;
   ELSE
   IF K = KM(W) & JN < JM(W) THEN DO; JN = JM(W);
   IF CD(K) = 7 THEN DO; FQ(W, JM(W)) < 0.49 THEN GOTO AV;
   DO J = JP+1 TO JM(W); FQ(W, J) = Q*FQ(W, J)/(Q+1); END;
   END;
   IF PRIT > 0 THEN
   /*02.3*/ PUT SKIP(2) EDIT ((F(W, J), FQ(W, J) DO J = 1 TO JM(W))
   (F(10), F(10, 3), SKIP);
Q = Q+1; N = XM(W); JN = FIG(W), CV(*) = 0; ZM(K), CV(K), APAT, CV(K), TP = 0;
END;
DO M = JN+1 TO YM;
   CV(M-N), TP = CV(M), TP; CV(M-N), APAT = CV(M), APAT;
   CV(M-N), VC(*) = CV(M), VC(*); ZM(M-N) = ZM(M);
END;
   YM = YM-N;
DO M = YM+1 TO YM+N;
   CV(M), VC(*) = 0; ZM(M), CV(M), APAT, CV(M), TP = 0;
END;
DO M = 1 TO WM; DO K = 1 TO XM(M);
   IF FIG(M), CV(K) > JN THEN FIG(M), CV(K) = FIG(M), CV(K) - N;
END FEED;
/* P: 
ROUTINE FOR FORMING CVES FROM THE SET OF ORIENT VALS OF THE 
LINE SEGS. THIS IS THE 'SIMP* CVE ROUTINE. WHEN THE CVES ARE 
FORMED, THE VALS ARE PLACED IN THE FIG AND CV STRUCTURES AND 
THE D AND OD VBLES. FINDL IS USED TO FIND THE LENGTH OF 
each SEG; PUTIN IS USED TO JOIN TWO PREVIOUSLY FORMED 
CVE'S INTO ONE. */

BCVE: PROC;
DCL (AP,Z,Y,AF) FIXED BIN (15);
Y=YM; J=2; AP=F(W,1);
BEG: IF FQ(W,J)>=0.49 THEN DO;
AF=F(W,J); IF Y=YM THEN GOTO NEW;
IF CV(Y).TP=1 THEN IF AF>=SUM/2
  THEN IF Y>YM+1
    THEN IF CV(Y-1).TP=3 THEN CALL PUTIN;
    ELSE GOTO ADNOW; ELSE GOTO ADNOW;
  ELSE GOTO FORNC;
ELSE IF CV(Y).TP=2 THEN IF AF<=SUM/2
  THEN IF Y>YM+1
    THEN IF CV(Y-1).TP=4 THEN CALL PUTIN;
    ELSE GOTO ADNOW; ELSE GOTO ADNOW;
  ELSE GOTO FORNC;
ELSE IF CV(Y).TP=3 THEN IF AF>0
  THEN IF AF>2*SUM/(Z-1) THEN GOTO FORNC;
  ELSE GOTO FORNC;
ELSE IF CV(Y).TP=4 THEN IF AF<0
  THEN IF AF<2*SUM/(Z-1) THEN GOTO FORNC;
  ELSE GOTO FORNC;
ELSE IF ABS(AF)>24 THEN IF Z=1
  THEN IF AF>0&Y>YM+1
    THEN IF CV(Y-1).TP=1|CV(Y-1).TP=3
      THEN CALL PUTIN; ELSE CV(Y).TP=3;
    ELSE IF Y>YM+1
      THEN IF CV(Y-1).TP=2|CV(Y-1).TP=4
        THEN CALL PUTIN; ELSE CV(Y).TP=4;
      ELSE IF AF>0 THEN CV(Y).TP=3;
        ELSE CV(Y).TP=4;
  ELSE GOTO FORNC;
GOTO ADNOW;
FORNC: ZM(Y)=Z;
NEW: Y=Y+1; Z=1;
IF ABS(AF)<24 THEN CV(Y).TP=5;
ELSE IF AF>0 THEN CV(Y).TP=1;
  ELSE CV(Y).TP=2;
CV(Y).APAT=AP; CV(Y).VC(1)=AF; SUM=AF;
GOTO ANO;
ADNOW: IF CV(Y).TP=1 THEN CV(Y).TP=3;
ELSE IF CV(Y).TP=2 THEN CV(Y).TP=4;
IF Y>YM+1 THEN IF CV(Y).TP=CV(Y-1).TP THEN CALL PUTIN;
Z=Z+1; CV(Y).VC(Z)=AF; SUM=SUM+AF;
AND:  AP=AP+AF;  CALL NORM(AP,KK);

END;
J=J+1;  IF  J=JM(W)  THEN  GOTO  BEG;
ZM(Y)=Z;  IF  CV(Y).TP=5&Z=1&(CV(Y-1).TP=3|CV(Y-1).TP=4)
THEN  CALL  PUTIN;
OD(W,*)=OD(W,*)+DX(W,*)+DY(W,*)=0;
DX(W,1)=3*COSD(CV(YM+1).*APAT);
DY(W,1)=3*SIND(CV(YM+1).*APAT);
DO  N=YM+1  TO  Y;
AP=CV(N).*APAT;  K=N-YM;
DO  J=1  TO  ZM(N);
AP=AP+C\( CV(N).VC(J)\)  CALL FINDL;
DX(W,K)=DX(W,K)+GL*COSD(AP);
DY(W,K)=DY(W,K)+GL*SIND(AP);
END;
D(W,K)=SQR(DX(W,K)+DY(W,K)+DY(W,K);  IF  D(W,K)>4  THEN
IF  DX(W,K)<0  THEN  OD(W,K)=180+ATAND(DY(W,K)/DX(W,K));
ELSE  OD(W,K)=ATAND(DY(W,K)/DX(W,K));
FIG(W).CV(K)=N;
END;
*/01.8*/
PUT  SKIP  EDIT((CV(J),VC(K)  DO  J=YM+1  TO  Y)
DO  K=1  TO  18))((Y-YM) F(9),SKIP);
PUT  SKIP  EDIT((CV(J),APAT DO  J=YM+1  TO  Y)) (F(9));
PUT  SKIP  EDIT((CV(J),APAT DO  J=YM+1  TO  Y)) (F(9));
PUT  SKIP  EDIT((OD(W,J) DO  J=1  TO  Y-YM)) (E(12,4,5));
PUT  SKIP  EDIT((OD(W,J) DO  J=1  TO  Y-YM)) (F(9));
XM(W)=Y-YM;  YM=Y;
FINDL:  PROC;
GL=MOD(AP,60);  IF  GL<6|GL>54  THEN  GL=4;
ELSE
IF  GL<14|GL>46  THEN  GL=5;
ELSE
IF  GL<18|GL>42  THEN  GL=4;
ELSE
IF  GL<26|GL>34  THEN  GL=3;  ELSE  GL=2;
END FINDL;
PUTIN:  PROC;
Y=Y-1;
DO  K=1  TO  Z;  CV(Y).VC(ZM(Y)+K)=CV(Y+1).VC(K);  END;
CV(Y+1).VC(*)=0;
CV(Y+1).*APAT,CV(Y+1).TP,ZM(Y+1),SUM=0;
Z=ZM(Y)+Z;
DO  K=1  TO  Z;  SUM=SUM+CV(Y).VC(K);  END;
FINAL:  END  CHEC;
APPENDIX C

EF: A PROGRAM WHICH LEARNS TO RECOGNIZE NUMBER SEQUENCES WHICH ARE THE DIRECTION CODES FOR HAND-DRAWN FIGURES.

EF IS WRITTEN IN PL/1 FOR THE IBM 360/50 AT THE A.N.U., CANBERRA. THE PROGRAM OCCUPIES ABOUT 20K BYTES OF STORAGE.

EF CONSTRUCTS A MODEL OF EACH CLASS OF FIGURES, FROM A SET OF SAMPLE SEQUENCES PRESENTED DURING THE LEARNING PHASE, (L.P.). THIS MODEL IS AN ORDERED SET OF PATTERNS. EACH PATTERN IS A GROUP OF NUMBER ELEMENTS AND IS DISCOVERED IN THE GIVEN MEMBERS. ABOUT 4 TO 6 SAMPLES PER CLASS ARE REQUIRED TO BUILD A SUITABLE MODEL. RECOGNITION OF A SEQUENCE DEPENDS ON THE SIMILARITY OF THE INPUT TO THE STORED PATTERNS; EF CANNOT HANDLE ORIENTED FIGURES, OR FIGURES WITH THE SAME CLASS NAME BUT OF DIFFERENT OVERALL SHAPE. ONCE THE CLASS MODELS HAVE BEEN FORMED, EF WILL RECOGNIZE ANY FURTHER SEQUENCES, IN THE WORKING PHASE, (W.P.).

THE TIME TO PROCESS A SEQUENCE IS ABOUT 2 SECONDS. HOWEVER, THE TIME DEPENDS MAINLY ON THE LENGTH OF THE SEQUENCE AND THE NUMBER OF CLASS MODELS, AND CAN RANGE UP TO 4 SECONDS.

THIS APPENDIX OUTLINES THE INPUT/OUTPUT SPECIFICATIONS FOR EF AND GIVES THE DETAILS OF THE PROGRAM. REFERENCE SHOULD BE MADE TO CHAPTER 8 OF THIS THESIS WHERE VARIOUS PARTS OF THE PROGRAM HAVE BEEN MENTIONED.

THE FOLLOWING GIVES THE INPUT/OUTPUT SPECIFICATIONS IN THE PROGRAM -- THE PROGRAM STATEMENTS SHOULD BE REFERENCED FOR THE FORMAT DETAILS.

INPUT REQUIREMENTS:

I. THE NUMBER OF LEARNING SAMPLES.

II. THE DIRECTION CODE SEQUENCE; (LIMIT 49 ELEMENTS).

2. A */' FOLLOWING THE LAST DIRECTION ELEMENT.

3. THE NAME OF THE INPUT IN INVERTED COMMAS, (E.G. '2').

OUTPUT SPECIFICATIONS:

01. THE INPUT DIRECTION CODE SEQUENCE.

2. PARAMETERS INCLUDING THE NEW PATTERN BEING INSERTED IN THE INPUT, ('JS').

3. PARAMETERS RELATING TO THE DISSIMILARITY MEASURE, ('ICAP').

4. THE CODED INPUT SEQUENCE (FOR EACH CLASS).

5. THE DECISION AND WHETHER CORRECT OR NOT.

6. IN L.P., THE NEW PATTERNS GENERATED FOR THE MODEL.

7. IN L.P., NOTICE IF THE PATTERN CANNOT BE INSERTED IN THE MODEL.

8. IN L.P., A LIST OF THE PATTERNS IN THE MODEL, THEIR EXPECTED FREQUENCY OF OCCURRENCE AND THEIR POSITION.

THE INPUT/OUTPUT STATEMENTS IN THE PROGRAM HAVE BEEN REFERENCED BY THE ABOVE CLASSIFICATION.
THE PROGRAM EF

EF: PROC OPTIONS (MAIN);

/* THE PROGRAM HAS BEEN DIVIDED INTO 10 SECTIONS, (A-J). */
1. A CONTAINS IntroDUCTORY STATEMENTS.
2. THE INPUT SEQUENCE IS OBTAINED IN B.
3. THE INPUT IS PROCESSED IN C.
4. A DECISION IS MADE IN D.
6. J IS A SUBROUTINE USED IN THE L.P.

THE FOLLOWING ABBREVIATIONS HAVE BEEN USED IN THE
COMMENTS:
VAL—VALUE, NO.—NUMBER, SEQ—SEQUENCE,
INS—INSERTED, ELS—ELEMENTS, VBLE—VARIABLE. */

/* A:
 THE DECLARATION, ON-CONDITIONS AND INITIALIZING STATEMENTS. */

DCL 1 P(2), 2 T(20) CHAR(5) VARYING,
    /*THE PATTERNS (T) IN EACH OF THE 2 CLASSES, (P). */
2 (DO(20), PS(20), F(20, 9), Q(20)) FIXED BIN(15),
    /* THE EXPECTED FREQUENCY, POSITION, FREQUENCY COUNT
 AND THE NO. OF OCCURRENCES OF EACH PATTERN. */
(X(50), /*STORE FOR THE INPUT SEQ.*/
B(2,50), /*STORE FOR CODED INPUT SEQ OF EACH CLASS*/
BX(50), BS(50), /*VIBLES FOR THE ABOVE.*/
CH(20), /*THE ALPHABET CODES FOR THE PATTERNS.*/
DEC(2), /*THE NAME OF EACH CLASS.*/
CF, CP, COR) PACKED CHAR(1), /*OTHER VIBLES.*/
(RX(20), RS(20), /*END OF INS PATTERN IN INPUT SEQ.*/
LL(2), /*NO. OF INS PATTERNS IN EACH CLASS.*/
JB(2), /*LENGTH OF CODED INPUT SEQ OF EACH CLASS.*/
LX(20), LS(20), /*NO. OF ELS IN THE INS PATTERN.*/
XL(20), NL(20), /*FREQUENCY OF OCCURRENCE OF INS
 PATTERNS IN THE INPUT SEQ.*/
NOS(20), Y(2,20), /*ORDERED POSITION OF INS PATTERN
 IN THE MODEL.*/
JM(2), /*NO. OF PATTERNS IN EACH CLASS.*/
RI, SUM) FIXED BIN(15), /*OTHER VIBLES.*/
(MCAP(2), /*MINIMUM DISSIMILARITY VAL FOR EACH
 CLASS.*/
ICAP) FLOAT DEC(5), /*VIBLE FOR ABOVE.*/
R(20) PACKED CHAR(5) VARYING, /*INS PATTERN STORE.*/
XX PACKED CHAR(50) DEF X,
BB(2) PACKED CHAR(50) DEF B;

ON ENDFILE (SYSIN) GOTO FIN;
PUT PAGE;

/*11*/
GET LIST (LIM);
CH=' ', DEC=' ', LL,JM=0; WM,VAL,KK=0;
P*DO, P*PS, P*Q=0; P+F=0; P*T=' ';
CH(1)='A'; CH(2)='B'; CH(3)='C'; CH(4)='D'; CH(5)='E';
CH(6)='F'; CH(7)='G'; CH(8)='H'; CH(9)='I';
CH(10)='J';
THE INPUT SEQUENCE IS OBTAINED.

NEW: \( XX=(50)' ' \); \( BB=(50)' ' \);
DO IX=1 BY 1;

GET EDIT(X(IX))(A(1));
IF X(IX)='/' THEN GOTO A2;
IF X(IX)=\(' ' \) THEN IX=0;
END;
A2: IX=IX-1; VAL=VAL+1;

THE INPUT IS PROCESSED TO DETERMINE THE SIMILARITY (ACTUALLY DISSIMILARITY) OF THE INPUT TO EACH CLASS. THIS PART IS SUBDIVIDED INTO 3 PARTS, (C1-C3).

LL=0; Y=0; MCAP=0;
IF VAL=1 THEN GOTO LERN;
DO w=1 TO WM;

THE NEXT PATTERN IN THE CLASS MODEL IS SELECTED TO BE INSERTED INTO THE INPUT. A TEST IS MADE TO FIND WHETHER IT CAN BE INSERTED OR IF IT HAS BEEN INSERTED PREVIOUSLY. IF NEITHER IS TRUE, THEN THE PART OF THE CODED SEQ PRECEDING ITS INSERTION POINT IS FOUND. WHEN THE LAST PATTERN HAS BEEN CONSIDERED, CONTROL MOVES TO THE END OF THIS PART, (ON4).

JS,LF,I=0;
N,NO=0; LP=LF; BS=' ' ; RS,LS,NOS,NL=0;
A1:  N,NO=0; LP=LF; BS=' ' ; RS,LS,NOS,NL=0;
AR1:  LF=LF+1;
IF LF>JM(W) THEN GOTO ON4;
IF P(W).DO(LF)=0 THEN GOTO AR1;
IF LP=0 THEN GOTO AR4;
KN=INDEX(XX,P(W,LF).T); IF KN=0 THEN GOTO AR1;
DO I=2 TO LL(W);
  IF (Y(W,I)=LF) & KN>RX(I-1)) THEN GOTO AR1;
END;
DO I=1 TO JB(W) WHILE(N<KN);
IF B(W,I)<"0" THEN DO;
DO L=1 TO LL(W);
  IF B(W,I)=CH(L) THEN DO; NN=N; N=N+LX(L); GOTO ON2;
END; END; END;
ELSE NN,N=N+1;
ON2:
END;
IF NN=N THEN JS=I-1; ELSE JS=I-2;
DO I=1 TO LL(W) WHILE(NC=0);
IF P(W,Y(W,I)).PS=P(W,LF).PS THEN NO=I;
END;
DO L=1 TO NO-1;
  LS(L)=LS(L); NOS(L)=Y(W,L); RS(L)=RX(L);
END;
J=0; RS(NO)=NN; I=KN-1;
DO M=1 TO JS;
  J=J+1;
  IF B(W,M)=CH(NO) THEN DO;
    DO L=1 TO LX(NO);
END;
BS(J+L-1)=SUBSTR(P(W,Y(W,NO)).T,L,1);  
END;  
J=J+LX(NO)-1;  END;  
ELSE BS(J)=B(W,M);  
END;  
JS=J;  

/*01.2*/  
PUT SKIP LIST('JS',KN,P(W,LF).T,N,JS);  
NO=NO-1;  

AR4:  
LO=LF;  

/* C2: THE PATTERN (FROM C1) IS INSERTED INTO THE INPUT SEQ. 
ALL OTHER PATTERNS FOLLOWING THIS PATTERN AND NOT AN OR 
COMBINATION ARE SUCCESSIVELY INSERTED. */  
AR2:  
L,NO=NO+1;  NOS(L)=LO;  LS(L)=LENGTH(P(W,LO).T);  
DO I=I+1 TO IX;  
  IF SUBSTR(XX,I,LS(L))=P(W,LO).T THEN DO;  
    DO J=1 TO I-LS(L)-1;  BS(JS+J)=X(RS(L)+J);  END;  
    JS=JS+I-LS(L)-1;  BS(JS)=CH(L);  
    I=RS(L);  RS(L+L)=RS(L);  
  END;  
  I=RS(L);  LS(L+L)=LS(L);  
  LO=LO+1;  
END;  

AR3:  
LO=LO-1;  
IF LO<=J^ICV^I THEN DO;  
  IF P(W).00(LOI=C THEN GOTO AR3;  
  IF P(W).PS(LO)=P(PS(LP)) THEN GOTO AR3;  
  GOTO AR2;  END;  
  DO J=1 TO IX-LS(L);  BS<JS+J)=X(RS(L)+J);  END;  
  JS=JS+IX-LS(L);  

/* C3: THE DISSIMILARITY VAL IS DETERMINED FOR THE NEW 
CODING. THIS VALUE IS COMPARED TO THE CURRENT MINIMUM; IF 
LESS, THEN THE NEW CODING BECOMES THE CURRENT CODING. 
CONTROL RETURNS TO C1 TO SELECT ANOTHER PATTERN. */  
SUM,MX,DIF=0;  
DO J=1 TO JS;  
  IF BS(J)!='0' THEN DO;  
    DO L=1 TO NO;  IF BS(J)=CH(L) THEN NL(L)=NL(L)+1;  END;  
  END;  
  ELSE SUM=SUM+1;  
END;  

DO N=1 TO NO;  
  DIF=DIF+ABS(NL(N)-P(W,NOS(N)).DO);  
  MX=MX+P(W,NOS(N)).DO;  
END;  
ICAP=DIF/NO;  
IF (MCAP(W)=0) || (ICAP<=MCAP(W)) THEN DO;  
  B(W,*)=BS(*);  JB(W)=JS;  MCAP(W)=ICAP;  LL(W)=NO;  
  RX=RS;  Y(W,*)=NOS(*);  LX=LS;  XL(W,*)=NL(*);  
END;  

/*01.3*/  
PUT SKIP LIST('ICAP',DIF,NO,JS,ICAP);  
GOTO A1;  

/*01.4*/  
ON4:  
PUT SKIP EDIT(BB(W))(A);  
END;
THE DECISION OF INPUT CLASS MEMBERSHIP IS MADE ON THE BASIS OF THE DISSIMILARITY VALUES. THE DECISION IS CHECKED WITH THE READ-IN NAME. IN W.P., CONTROL RETURNS TO PART B. */

LERN: IF MCAP(1)<MCAP(2) THEN W=1; ELSE W=2; /*01.5*/
PUT SKIP LIST('FIG REC AS ',DEC(W)); /*12.3*/
GET LIST(COR); /*01.5*/
IF COR=DEC(W) THEN PUT LIST('CORRECT DECISION');
ELSE PUT LIST('INCORRECT DECISION');
IF VAL>LIM THEN GOTO NEW;

NEW PATTERNS ARE FOUND IN THE CODED SEQUENCE OF THE CORRECT CLASS. IF THE INPUT IS THE FIRST LEARNING SAMPLE FOR THIS CLASS THE INPUTTED SEQUENCE IS CONSIDERED. THIS PART HAS BEEN SUBDIVIDED INTO 3 SECTIONS, (E1-E3). */

E1: THE SEQ TO BE SEARCHED FOR PATTERNS IS FOUND, (B). */

IF COR=DEC(W) THEN DO;
DO W=1 TO WM; IF DEC(W)=COR THEN GOTO CN3; END;
W,WM=WM*1; DEC(W)=COR; JB(W)=IX; B(W,*)=X(*);
END;
CN3: I=2; RI,LP,N1=0; LS,RS=0; R=(0) ' '

E2: PATTERNS - A REPETITION OF THE SMALLEST GROUP OF ELEMENTS (AT LEAST ONE EL MUST BE A NO.)- ARE FOUND IN A SUCCESSIVE SEARCH OF THE SEQ. THE POSITION OF THE PATTERN RELATIVE TO THE OTHER INS PATTERNS IS FOUND, (N1,N2). */

AG1: DO K=1 TO 4;
IF (I>RI+K) & (I+K-1<=JB(W)) THEN DO;
IF SUBSTR(BB(W),I-K,K)=SUBSTR(BB(W),I,K) THEN DO;
LP=LP+1; LS(LP)=2; RI=I+K-1; N1=0; J=0;
DO M=0 TO K-1; IF B(W,I+M)>'0' THEN GOTO OT2; END;
GOTO OT2;
AG2: DO N=0 TO K-1;
J=J+1; IF J>5 THEN GOTO OT2;
IF B(W,I+N)<'0' THEN DO;
DO L=1 TO LL(W); IF CH(L)=B(W,I+N) THEN M=L; END;
J=J+LENGTH(P(W,Y(W,M)).T)-1;
IF J>5 THEN GOTO OT2;
R(LP)=R(LP)||P(W,Y(W,M)).T;
END;
ELSE R(LP)=R(LP)||B(W,I+N);
END;
DO N=I-K-1 BY -1 TO 1 WHILE(N1=0);
IF B(W,N)<'0' THEN DO;
DO L=1 TO LL(W); IF CH(L)=B(W,N) THEN N1=L; END;
END;
AG3: IF RI+K<=JB(W) THEN IF SUBSTR(BB(W),RI+1,K)=R(LP) THEN DO; RI=RI+K; LS(LP)=LS(LP)+1; GOTO AG3; END;
I=RI+1; N2=0;
IF LP>1 THEN IF R(LP)=R(LP-1) THEN DO;
LS(LP-1)=LS(LP-1)+LS(LP);
IF LS(LP-1)>9 THEN LS(LP-1)=9; GOTO OT2; /*01.6*/
PUT SKIP LIST('NEW',LP,R(LP),LS(LP),N1E);
DO N=R(I)+1 BY 1 TO JB(W) WHILE(N2=0);
IF B(W,N)<0 THEN DO;
DO L=1 TO LL(W); IF CH(L)=B(W,N) THEN N2=L; END;
END; /*01.7*/
END;

/* E3: AN ATTEMPT IS MADE, WHEN A NEW PATTERN IS FOUND, TO INSERT THE PATTERN IN THE CLASS MODEL. IT MAY BE INSERTED BETWEEN PATTERNS, AS AN OR COMBINATION, OR IT MAY BE REJECTED. */
IF N2=0 THEN IF LL(W)>0 THEN N2=P(W,LL(W)).PS+1;
ELSE N2=1;
IF N1>0 THEN N1=P(W,Y(W,N1)).PS;
IF N1=N2 THEN DO;
/*01.7*/
PUT SKIP LIST('POSN AMBIGUOUS'); GOTO OT2;
END;
ELSE
IF N2-N1=1 THEN DO;
IF K<LENGTH(R(LP)) THEN GOTO OT2;
LX(LP)=N1; RS(LP)=1;
END;
ELSE DO;
DO M=1 TO JM(W) WHILE(P(W,M).PS<N2);
IF (P(W,M).PS=0) & (R(LP)=P(LP)) THEN DO;
P(LP).PS=(LS(LP)).PS+1;
END;
DO M=1 TO JM(W) WHILE(P(W,M).PS<N2);
IF P(W,M).PS>N1 THEN DO;
IF LENGTH(R(LP))>LENGTH(P(W,M).T)
THEN KN=INDEX(R(LP),P(W,M).T);
ELSE KN=INDEX(P(W,M).T,R(LP));
IF KN>0 THEN DO;
LX(LP)=P(W,M).PS; RS(LP)=0;
GOTO OT3; END;
END;
/*01.7*/
PUT SKIP LIST('CANNOT FIND POSN');
OT2: RL(LP)=(0) ' '; LS(LP)=0; LP=LP-1;
GOTO OT3; END;
OT3: I=I+1; IF I<JB(W) THEN GOTO AG1;

/* F: PATTERNS WHICH ARE STORED BUT NOT IN THE CURRENT CLASS MODEL ARE SEARCHED FOR IN THE CODED SEQ. IF FOUND, THEIR FREQUENCY COUNT IS UPDATED. */
DO J=1 TO JM(W);
IF P(W,J).DC=0 THEN DO;
KN=INDEX(BB(W),P(W,J).T);
IF KN>0 THEN DO;
N=LENGTH(P(W,J).T); MX=0;
KN=KN+N;
IF KN<JB(W) THEN IF SUBSTR(BB(W),KN,N)=P(W,J).T
THEN DO; MX=MX+1; GOTO OT5; END;
IF MX>9 THEN MX=9;
OT5:
THE FREQUENCY COUNT OF THE INS PATTERNS IS UPDATED.*/

DO K=1 TO LL(W);
  IF XL(W,K)>9 THEN XL(W,K)=9;
  IF XL(W,K)>0 THEN
    P(W,Y(W,K),XL(W,K)).F=P(W,Y(W,K),XL(W,K)).F+1;
  END;
  M=C;

THE NEW PATTERNS ARE INS INTO THE CLASS MODEL.*/

DO L=1 TO LP;
  CALL INVALUX(L)+M,K,RSU(L));
  P(W,K,*)..F=0;
  P(W,K,*)..R=L(L); P(W,K,LS(L)).F=1; P(W,K,).Q=1;
  IF RSU(L)=1 THEN P(W,K).PS=LS(L)+M+1;
  ELSE P(W,K).PS=LS(L)+M;
  IF RSU(L)=1 THEN M=M+1;
  END;
  P(W,DO(*)=0;

THE NEW EXPECTED FREQUENCY OF EACH PATTERN IN THE CLASS
MODEL IS CALCULATED. THOSE WITH A ZERO VAL ARE LEFT OUT
OF THE CURRENT MODEL.*/

DO J=1 TO JM(W); SUM=0;
  DO N=9 BY -1 TO 1;
    SUM=SUM+P(W).F(M);
  IF SUM>=P(W,).Q/2 THEN GOTO TO1;
  END;
  GOTO T02;

TO1:
  P(W,).DO=N;

TO2:
  END;
/*01.8*/ PUT SKIP(2) LIST(P(W,).DO,P(W,).PS,P(W,).T);
  GOTO NEW;

ROUTINE FOR UPDATING THE CLASS MODEL TO INCLUDE A NEW
PATTERN.*/

INVAL: PROC(N,JSOP,M);
  DCL (N,JSOP,M,J) FIXED BIN (15);
  JM(W)=JM(W)+1;
  DO J=JM(W)-1 BY -1 TO 1 WHILE(P(W,PS(J))>N));
    P(W,J+1,*)..F=P(W,J,*)..F; P(W,J+1,).Q=P(W,J).Q;
  END;
  JSOP=J+1;
  IF M=1 THEN DO;
    DO J=JSOP+1 TO JM(W); P(W,J).PS=P(W,J).PS+1;
  END INVAL;

FIN: END;
Appendix D

DETAILS OF AN ON-LINE SYSTEM FOR INPUTTING THE DIRECTION CODES

This Appendix describes a system for inputting the direction codes of a hand-drawn line figure (see Chapter 3), to the recognition program, (designated OM), 'on-line' from the drawing of the figure. The emphasis when devising this system has been on simplicity in providing a means for inputting the codes. For this reason, some constraints have been applied to the user and various parts of the system would require modification, to overcome these constraints.

D.1. The Equipment

The system consists of a pen, (for writing), acting in a potential divider, partly immersed in a water bath. Alternate (X,Y) coordinates are transmitted to an IBM 1827 analogue multiplexer, and analogue-digital converter (to be referenced as the 1827 herein), which in turn transmits the coordinates to the University's IBM 360/50 computer. An X-Y recorder (connected through the 1827 to the computer) is available to re-draw the input figure, by plotting the(X,Y) coordinates.

Further details of the scheme follow:-

(a) Water bath: - the bath is a 15" x 15" perspex dish filled to approximately 1/2" in height with tap water and a teaspoon of salt. 12 x 1" screws are placed (about a central position) at 1" intervals, along each side, (being screwed through the side). Each screw is connected to one side of a relay, and corresponding screws on opposite sides of the bath are connected, when the relays of both screws are on,
to a 4.5V battery, (see Figure D.1).

(b) **Pen:** This is a ball-point pen capsule containing a brass rod, which is pointed at the writing end. A spring is inserted between the rod and a small screw at the top of the pen - the rod makes electrical contact with the screw when the pen (the rod) is pressed (on a writing surface). The screw is wired to the 1827, via a high impedance operational amplifier.

(c) **Circuitry:** An astable multivibrator provides the switching of the relays - alternately for the set of (multi-pole) relays connecting two opposite sides, (X axis) and for the set connecting the other sides (Y axis). The astable is interfaced to the relays by emitter followers. The switching signals of the astable operate a monostable multivibrator which provides the synchronizing pulses for the 1827 to take a sample of the pen voltage. The circuitry is shown in Figure D.2.

D.2. Operation of the System

The user draws figures by writing with the pen pressing on the bottom of the bath. After each figure, the pen must be lifted from the bottom. A subroutine, designated PG, using the 1827, alternately samples the X and Y coordinate positions of the pen, and reads them on to disc. When the pen voltage drops below 0.3 volts on both the X and Y coordinates, the subroutine PG assumes that the pen has been removed from the water bath and hence the particular input figure is finished.

A second subroutine, designated GC, operates on these stored values to obtain the direction codes. The distance between successive coordinates is calculated, and if
Table D.1. Connections between Two Corresponding Screws in the Bath, (showing the Pen in a Potential Divider).

Table D.2. Switching Circuitry for the Relays and the 1827.
greater than a certain threshold (currently 1/10"), the
direction code is computed. (Note that this calculation
produces a 'fixed-grid' direction code, which is not a
simple angular quantizing of the obtained direction).
If the distance is less than the above threshold, the
next (X,Y) coordinate pair is used to calculate a
distance to the first pair considered.

A modification of program AN3 (Section 2.2.2),
designated OM, accepts the calculated codes as input, and
attempts to recognize the drawn figure. The results are
outputted on a line printer.

D.3. Comments on the Operation

The writer's speed is constrained to be not faster
than 2 inches per second. This speed is limited by the
maximum switching rate of the multi-pole relays. It is
intended to remove the relays* and provide electronic
switching, thereby allowing fast (sloppy) handwriting.

The water bath has been found quite satisfactory -
the effective area for drawing is approximately 9" x 9",
being limited by the requirement of the pen voltage to
be greater than a certain threshold, (mentioned above).
The equipotential lines in the bath are quite parallel
for about 10" of the total 13" between the screw ends.

The sensitivity of the pen position is very high.
A pen movement of 1/10" produces approximately 0.03
volts difference in potential, and as the 1827 can
detect 0.1 millivolt change, there is no problem with
detecting small pen movements.

*Currently operating at 20 c/s.
From the experiments conducted so far, the system appears accurate in its measurements. However, small characters (2/4" high) tend to become distorted, possibly because of local variations in the bath potential due to the water disturbance, caused by the pen. For figures of about 1"-2" high, the X-Y recorder reproduces a sufficiently accurate version of the drawn figure. (It is possible, because of the transparency of the bath, to trace figures drawn on paper placed under the bath).

At present, the programs obtain the (X,Y) coordinates from disc, and hence, the time to classify a character is consequently large (3-4 seconds to recognize a numeral) and not a true indication of the potential of the system. The programs therefore do not operate 'on-line', but this can be effected with changes to the 1827 program and the computer's programming system. It is envisaged to have the programs written in IBM 360 Assembler language, in which the processing time will be accurately known, and in which the operations will be performed on the X-Y coordinates as they are received.

The system is currently being tested and various modifications appear to be necessary, as discussed above. As far as can be gauged, the performance of OM is close to that of AN3, being relatively unaffected by the inputting of line segments. It is expected that figures with sharp corners may have their shape slightly distorted.

OM does not have the ability to accept user-specified classes, as yet; the average members have been specified by the author. The extension to learning is envisaged, when the system has been subjected to considerable experimentation.
A General Approach to the Machine Recognition of Patterns†

By J. F. O'CALLAGHAN

Department of Engineering Physics, The Australian National University, Canberra, A.C.T., Australia

[Received March 24, 1967]

ABSTRACT

An approach to machine recognition, which attempts to overcome the problem of specifying suitable features for figures such as hand-printed characters, is suggested.

Patterns or features may be considered as a set of regularly occurring elements. During the learning phase the machine discovers these elements (from their regularity) in the input, and from a set of examples of a class, forms a hierarchical description of each class.

Patterns found in each category are sought in a new input. The unknown figure is recognized as a member of that class for which the match between the descriptive patterns in that category and input is most similar.

§ 1. INTRODUCTION

Formal procedures which achieve recognition consist of

(a) extracting certain features from the input,
(b) using these features to make a decision about the input category.

Results of attempts at machine recognition indicate that the designer cannot adequately specify in the programme the attributes for figures. Therefore, the machine‡ must learn from a known set of typical figures, the best features for classification. It is evident that for a varied environment (e.g. handwriting) the device must

(a) generate suitable types of features,
(b) discover those which discriminate figures best, and consequently,
(c) assign the set of features to their respective figures.

Few programmes incorporating such procedures have been devised, but these are regarded to be among the most powerful (Bledsoe and Browning 1959, Fogel et al. 1965, Uhr and Vossler 1961).

The purpose of this paper is to suggest a general approach to the operation of a machine which generates properties and selects the best from the set. The attitude is taken that “the knowledge which we must give the machine by pre-programming may be quite small” (Vossler and Uhr 1962). The programme is a structure which allows a model of the

† Communicated by the Author.
‡ The term ‘machine’ can be regarded (for practical purposes) as a computer programme, but is used for generality.
environment to be built during a learning phase, by a non-deductive process. The descriptive elements in the constructed model become the characteristics of the figures.

The approach is derived from consideration of the need for pattern recognition by the organism in its struggle for survival. An organism (in particular, the human being) recognizes patterns of success (and failure) in its experience as an aid to making decisions, which increase its chances of survival. The principles guiding the proposed theory are thus of a philosophical and psychological nature. Although many ideas have been gained from a study of the learning process in human beings (in particular, children), they are not intended to form a theory for any human activity.

The proposal is considered applicable to any problem of pattern recognition. The applications discussed in this paper are directed to hand-drawn figures. Reference is made to suitable techniques for implementing the theory.

§ 2. Outline of Principles

2.1. General Scheme

The characteristic which perhaps distinguishes the human being from other organisms is his ability to build a model of the environment. A simplified model enables him to make decisions in complex situations. The machine in a similar way must be capable of duplicating the organization of the input, in order to classify entities in the environment. The organization consists of the essential features (those necessary for discrimination) and the order of the features within the input. As Fogel has rightly pointed out, the key problem in many areas of artificial intelligence is "automating an inductive process that will generate useful hypotheses concerning the logic that underlies the experienced environment" (Fogel 1963).

The model in the machine must reflect the organization in the input. The programme for the device gives a structure; specific descriptions of the environment (of each figure) must be learnt.

The input, which is a representation of the environment, will be considered in the form of a sequential (discrete) sequence of elements. This form which may be produced by a scanning mechanism is convenient for input to present-day devices. Classifying figures is thus considered as a sequential decision process. An advantage for the sequential processing is that the decisions may be made at any stage of the input.

Because our (human) environment is to a certain degree redundant, regularly occurring associations exist between objects which are perceived. Human beings use these associations (similarity and contiguity) as a means for grouping elements, to form the basic blocks in the structural description. Although no two leaves of a tree are identical in every respect, the similarity (association) is perceived by human beings to form the concept ‘leaf’, which is used further to describe objects such as ‘trees’.
The theory proposes that the machine use associations to describe the input. The concept of the figures will be structural descriptions of the members in terms of their associated elements.

The model has a hierarchical structure, in which higher level elements are sets of associated lower level elements (or regularities). For instance, a curve can be described in terms of its similar slope-difference values of constituent straight-line segments (as lower level values). The lowest level consists of the input values, while the highest level is a coded representation of the figure.

2.2. Processing

2.2.1. Learning phase

Regularities may be found on each level by

(a) a search for associations,

(b) a prediction that the association will recur, followed by a confirmation that it does occur.

The projection is an ‘inductive’ means of determining the most regular associations and corresponding elements (patterns).

Patterns are then the descriptive features of the model and partition the input on each level. An important part of the learning phase is to discover the regular associations which provide the best description of the figure. The description in terms of patterns will be best in the sense that it will be:

(a) simple (because it is composed of regular elements),

(b) complex enough to contain the essential features for discrimination.

For example, description on a particular level of the figure ‘2’ which is simple but useful for discrimination is ‘an arc of a circle and two straight lines’. The order of these elements (arc, straight lines) may be the subject of another level.

An outside teacher (OT) tells the machine the correct decisions for each input so that the associations can be found amidst the ‘noise’, in the sequence. The values or ‘concepts’ of each class are formed as an ‘average’ of its members. A concept (which may contain more than one description) is stored for each class.

2.2.2. Working phase (recognition)

The input sequence is matched sequentially to each suggested description and is classified as a member of that class to which the sequence description shows most ‘similarity’.

§ 3. Structure

3.1. Environment and Representation

Present-day devices can accept a single dimension of value, with time as a variable. For more dimensions, scanning and sequencing of the
environment are necessary. The proposed machine receives a representation of the ‘data’ as a sequential stream of information. The elements of this sequence are an abstract code of ‘objects’ in the environment. Such codings, considered as specific applications for this paper, may be:

(a) a sequence of numbers representing the angular directions of line segments in a line drawing (e.g. a Freeman code (Freeman 1961)),

(b) a sequence of ABC... where A, B, C,... correspond to such objects as arms, legs, head, ... by which animals, human beings can be classified.

No limitations are necessarily imposed by this approach. However, for complex (more than two dimensions) data, the processing may become more difficult and time-consuming.

The representation must contain relevant features of the environment with which the machine can discriminate the input sequences. The ability to form this coding is considered to be one given to the device†. However, this does not imply that the machine must ‘pre-process’ the data. The only pre-processing considered necessary to produce the sequence is that which removes the figure from the ground. For example, such operations as colour removal and thinning may be necessary for the coding. Information concerning ‘noise variables’ such as size of the figure and distortion to parts of the figure is embedded within the code. The knowledge that such information is a variable of the concept of the figure is found as a result of the search for regularities.

3.2. Associations and Patterns

The elements in the stream exhibit some form of association (by virtue of their redundancy), from which a model of the input can be built. The associations may occur in two forms, namely, similarity (of which a special form is identity) and contiguity‡.

An advantage of the hierarchical system is that the input code may be designed on such a simple descriptive level that the resulting sequence will contain identical elements. A descriptive feature for line drawings on one level could be the angle of straight-line segments. In the code, quantization of the various angles in 45° divisions will reduce the number of variable elements to eight. Coded figures would then produce sequences which are sufficiently redundant to contain identical elements, yet retain enough information for discriminating features to be found.

Sets of recurring elements may then be detected by the machine and form patterns which are the basic descriptive parts of the model. In a

† Such abilities as coding the data (and the generated associations) are necessary parts of the production of a descriptive model, and ones for which formal procedures may be ‘impossible’. Chapter 1 of Hempel (1966) presents an argument (not entirely accepted by the author) for this ‘impossibility’.

‡ These forms, perhaps the most widely acknowledged, were apparently suggested (for human processing) by Hume (Burks 1958).
similar manner, the detection of recurring groups of different elements in some spatial arrangement (e.g. two elements always appearing consecutively) will lead to the formation of contiguous patterns. This spatial ordering is inherent in the type of descriptive code considered. Discrimination between a ‘5’ and a ‘2’ (both having a curve and two straight lines, in the same orientations) can be effected through use of the association of contiguity.

In cases where identity is infrequent and where variations in elements exist (as may occur for the coded input to higher levels), the association of similarity must be discovered. Similar patterns are found during the learning phase as a result of grouping non-identical but categorically equivalent figures. The degree of similarity, defined as a ‘distance’ between members of a class, must be determined as a consequence of discovering that class. This resulting degree may be used to detect further associations of this type in the input.

In the above situation, the discovery of similarity can be aided by a controlled learning phase, in which the OT gives the more regular members of each class first. In this way the association is more easily found.

3.3. Process on each Level

During the learning phase, the machine attempts to find descriptions for each figure, in terms of the regularities in the sequences. (An OT tells the machine the class (e.g. ‘2’) to which the input is a member.)

The process of determining the patterns for the figure follows a set of non-deductive rules (sometimes termed the projection of regularity (Ackermann 1966)):

(a) the associations are generated by noting their recurrence,
(b) a description of the association is formed (e.g. it may simply be a set of elements which have been repeated),
(c) the regularity is projected and future elements are predicted from the expected recurrence. The confirmation of the prediction by the future input sequence provides a measure of the regularity of the association.

In this way, patterns (regular associations) which become the descriptive features in the model, may be formed.

If the patterns are non-exclusive then many partitions (of the input) may result. Thus, a line drawing can be approximated in various ways by straight-line segments of variable angle. The machine must discover the best partitioning—that containing the most regular associations. This partition will constitute a ‘level’ description of the figure.

3.4. Hierarchies

The associations discovered between elements can be coded to form another sequence of elements. The process of searching for associations and discovering the best description for each figure may occur on this
higher level. By this method of coding and searching, a hierarchical structure may be formed, which can be used to classify more complicated objects in the environment. For instance,

(a) coding the similarity between angular difference of segmented lines will produce a representation for curves,

(b) coding the contiguity between arms, head, legs, ..., etc. will produce representations for animals, human beings, ....

On the highest level there will be a coded description of the figure—the figure itself constitutes a pattern. Each element represents (through the associations) a set of regular elements on the lower level.

The projection of regularity may be 'justified' on a particular level by the recurrence of the patterns on that level. It is the purpose of higher levels (with broader experience) to predict when changes of patterns will occur. In other words, because a higher level element is composed of lower level patterns, an occurrence of the former can be used to predict (with a better chance of success) the patterns.

The interstage coding must present a form from which the associations necessary for the required classifications can be found. The ability to code the association into a suitable form is an essential part of the given structure of the machine. The coding serves as a convenient means of decreasing storage requirements.

There is no theoretical limit to the number of levels in the hierarchy. The number of stages depends upon the required decisions. To classify numbers, the following three levels might be used:

(a) first for angular rotation (differentiating '∞', '8'),

(b) second for type of curves (differentiating types of curves),

(c) third for order of the curves (differentiating '5', '2').

3.5. Concepts

On each level the machine stores the patterns which form the description (in an order prescribed by the higher level) of particular figures. On the highest level only one pattern (for each figure) is stored. Thus the 'concept' of a figure is the set of descriptive patterns, stored in a hierarchical structure. Considering the three levels for numbers suggested above, the concept for the figure '2' may be on the third level, 'C_1 followed by C_2 followed by C_3' (the coding language here is incidental) where C_1, C_2, C_3 are defined on the second level as curves of certain curvature (e.g. C_2, C_3 have zero curvature) and where each curve, in turn, is defined on the first level as sets of segmented lines (where each line has a certain direction).

3.6. Decisions

Each input sequence is searched as it enters the machine for those patterns which have been found in the experienced sequences. Hypotheses that a certain pattern (e.g. a particular figure) will occur in the input are
generated from the (lower level) values discovered in the given sequence. As more of the input is obtained, the hypothesis becomes confirmed to a certain degree. Confirmation of a hypothesis (Watanabe 1960) is measured as a 'distance' between the input values and the (predicted) stored description, and is essentially a measure of similarity.

In order to make a decision, the relative preference or credibility (Watanabe 1960) of each hypothesis must be determined. A hypothesis becomes credible on the basis of:

- (a) its \textit{a priori} information, which is a measure of the degree of success that the pattern has had in the past,
- (b) a function of its confirmation, which is a measure of the success the hypothesis has on the present sample.

In the line-drawing example, a figure hypothesis is confirmed by the orientation (first level), the shapes (second level) and the order (third level) of the constituent curves. Given correct orientation\(^\dagger\) and order\(^\ddagger\) for the curves, the credibility of the hypothesis depends upon its \textit{a priori} value (i.e. contextual information) and its confirmation (i.e. the degree of similarity between the concept and the input description).

The figure corresponding to the most credible hypothesis (m.c.h.) becomes the decision for the input. Because the process is sequential, the decision may be made at any stage of the input.

Given equal \textit{a priori} information for the figures, the hypotheses with the best confirmability (best-fit) will receive maximum credibility. For unequal \textit{a priori}, one expects at the end of the sequence for a figure that the hypothesis with the best-fit will receive the decision. It is possible for the input and concept to have lack of fit for most of the comparison, but because there is better fit over the remainder, the machine may make the correct classification.

\section*{§ 4. Processing}

\subsection*{4.1. Learning}

The ability to learn allows the machine to recognize those figures (within the scope of its codings) specified by the OT. There are two stages (not distinct) in the learning process.

\subsubsection*{4.1.1. Discovery of categories}

Initially the machine has no information about the environment. Samples of each figure are given to the machine (together with their classifications). In this way, the decisions required are found. By discovering the regularities in the sequence, the machine can form the best descriptions for each input.

\(^\dagger\) The orientation of the figure may be useful in discrimination (e.g. 'M' and 'W').
\(^\ddagger\) This factor may be ensured by the order in which the matching of the input and curves is performed.

\(||\) The idea of best-fit as a decision criterion has been used (Sebeysten 1962).
4.1.2. Optimization of the description

Once a description is formed for a figure, it may be used to recognize further inputs. The concepts must be optimized, to form an ‘average’† of the members of the figure to achieve correct classification. There are two constraints—simplicity and complexity—affecting the descriptions. These occur in the following situations (at the end of processing a figure):

(a) The m.c.h. does not correspond to the correct figure. This result implies that the correct hypothesis is not the best fit to the input. The machine must alter the form of the correct hypothesis (or of the m.c.h.), thus increasing the complexity of the description. The change may be performed by altering (in an iterative manner) the pattern values receiving low confirmation.

If the figures are of different forms (e.g. 2, 2) then a procedure for constructing subset descriptions is necessary‡.

(b) The m.c.h. does correspond to the correct figure. In this case the machine has given the correct classification. The description of the figure may be simplified by incorporating into the concept any further regularities found in the input.

It is obviously necessary for the samples to be representative of their particular class. That is, in order to acquire suitable concepts, the machine must be trained on similar figures to those it will receive in the working phase.

4.2. Recognition

The learning phase may be terminated when the concepts are well-averaged—viz. when little variation occurs in their form with further samples. The descriptions are then representations of the experienced sequences.

The model of the environment can be used to classify figures. Elements in an input sequence are compared to those stored (i.e. the patterns). Hypotheses are generated on each level, which predict that certain elements will occur in the future. On the highest level, hypotheses suggest that the given sequence forms part of a particular figure. (An advantage of this scheme is that only those figure hypotheses with a close match to the input initially need be generated.) The input is recognized as that figure to which the hypothesis acquiring the most credibility corresponds.

§ 5. Conclusions

The principles of operation for a machine which recognizes learnt figures have been given. Detection or feature extraction is conceived of as the searching for patterns in a sequential flow of information and

† ‘Average’ here means a representative value (e.g. a suitable mathematical average for points (representing the numbers) in a space, may be the centroid).
‡ The formation of subclasses has been investigated by many workers (Mattson and Dammann 1965, Sebeysten 1962).
generating hypotheses about the possible classes. Classification is viewed as the making of a decision from the most credible hypothesis.

The more important advantages of the scheme appear to be:

(a) the machine can determine the best set of features; given a suitable input coding, the machine can learn any figures specified by an OT,

(b) order of the ‘features’ in the figure is considered,

(c) decisions may be made at any stage.

The main claim for the scheme is its objectivity—it searches for patterns, forms hypotheses and tests them, with the aim of achieving a model of the input. The author feels that such motivation (inherent in inductive procedures) is necessary for machines to solve the problem of recognition.

The length of the learning phase may be rather long. It must be emphasized that we are requiring a machine to achieve what the human takes many years to do, so that it would not be unreasonable to expect a long learning process. However, techniques can be devised (e.g. heuristics) which might make the process shorter.

Finally, it is reasonable to expect the machine to recognize a wide range of figures (e.g. many styles of hand writing), with the limitations being storage requirements and access to storage.

Acknowledgment

The author wishes to thank Dr. S. Kaneff for helpful discussions on aspects of this paper.

References


