View metadata, citation and similar papers at core.ac.uk
brought to you by CORE
provided by Elsevier - Publisher Connector

Mathematical Modelling, Vol. 2, pp. 383–391, 1981
Printed in the USA. All rights reserved.

0270–0255/81/040383–09$02.00/0
Copyright © 1981 Pergamon Press Ltd.

A MODEL FOR LEARNING GLOBAL PROPERTIES

PAUL EITNER AND MANFRED KOCHEN

Mental Health Research Institute
University of Michigan
Ann Arbor, Michigan 48109, USA

Communicated by Frank Harary

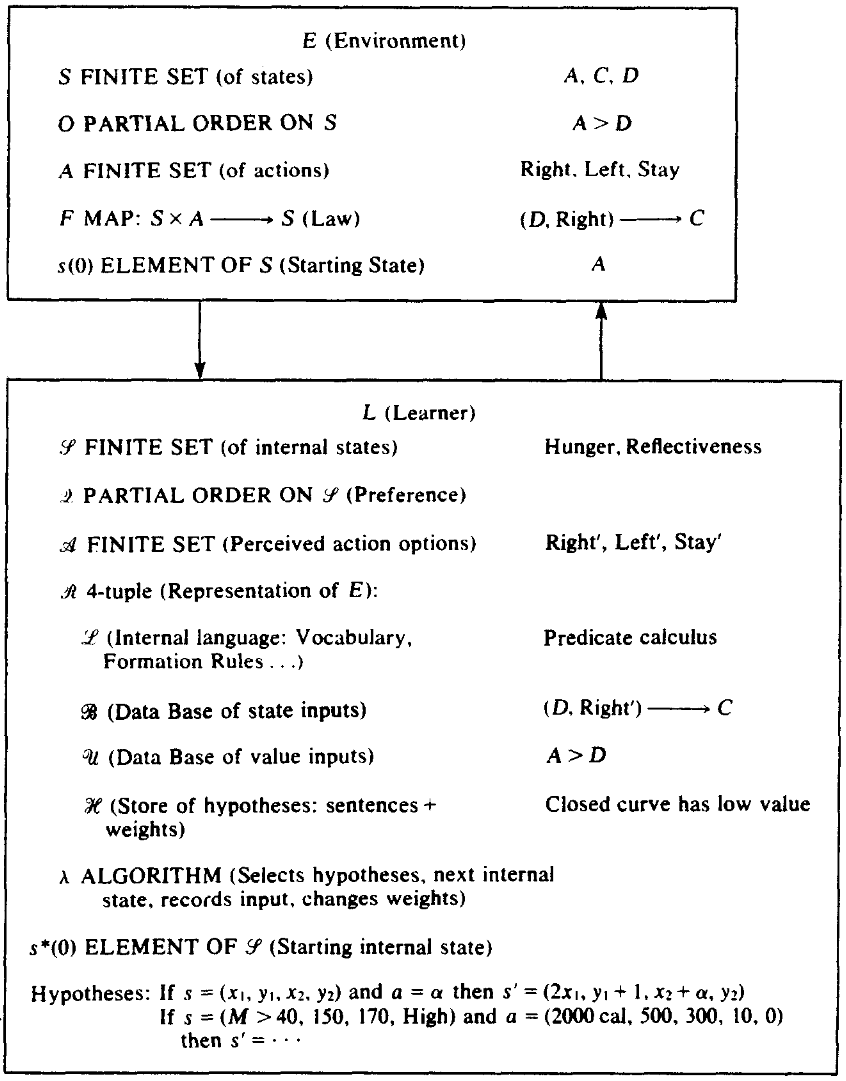
Abstract—This paper reports developments in a mathematical model of cognitive learning. The model describes the processes of formation, testing, and revision of hypotheses held by a learner attempting to understand an environment. The fundamental assumption is that learning proceeds by a feedback cycle, where hypotheses are tested for validity against external reality and reweighted according to the outcome. A particular application of this model to the case of a mechanical "beetle", mapping a geometrical environment, yields results of interest in artificial intelligence and robot design, including results on the computability of several geometrical predicates.

1. INTRODUCTION

This paper reports another step in the long-term development of a mathematical theory of learning [1, 2]. The main idea is that learning proceeds by the formation, revision, reweighting and use of hypotheses so that the learner can identify and cope with an ever increasing variety of opportunities and traps. Hypotheses have the form of propositions in an applied predicate calculus, e.g., "if the state of the world at time t is $s(t)$, and the learner takes action $a(t)$, then T time units later the state is $s(t+T)$ and this is (or is not) preferred to the present state." Hypotheses are reweighted, revised or formed on the basis of information about the actual state of the world resulting from action $a(t)$ and about how the learner evaluated the change of state.

Previous results based on this general model pertained to the convergence properties of particular schemes for reweighting hypotheses. It was proved, for example, by Hantler and Kochen [3], that if hypotheses identical with the one that generates the environment are stored along with numerous others, then there exists a reweighting scheme for which the probability of selecting the most frequently confirmed hypothesis is bounded from below by a reasonably large number.

A simplified model is sketched in Fig. 1. On the left side in each box, we specify the general definition of an environment and a learner. On the right side, there are specific interpretations of the various sets with reference to an imaginary beetle that can move along surfaces or curves. We are interested in the possibility that this "beetle" could learn to identify such concepts as "torus" or "Mobius band." The importance for a mathematical model of the distinction between the "state of the world," e.g., "a bug *is* at a certain point on the surface of a torus"; and the perceived state of the world, e.g., "the bug *thinks* it is on a sphere," has been recognized for a long time [4]. Only recently has it become practically necessary to deal with that distinction in the context of robot design. It has also been recognized for some time that a state of the world perceived through a parallel input-at-a-distance system, such as vision, is quite different from the perception of the world through a tactile sensor. Recent work in artificial intelligence [5] has stressed the idea of a "frame," also called a script or schema by various workers, as a



E (Environment)

S FINITE SET (of states)	A, C, D
O PARTIAL ORDER ON S	$A > D$
A FINITE SET (of actions)	Right, Left, Stay
F MAP: $S \times A \longrightarrow S$ (Law)	$(D, \text{Right}) \longrightarrow C$
$s(0)$ ELEMENT OF S (Starting State)	A

L (Learner)

$\mathcal{S}$ FINITE SET (of internal states)	Hunger, Reflectiveness
$\mathcal{Q}$ PARTIAL ORDER ON $\mathcal{S}$ (Preference)	
$\mathcal{A}$ FINITE SET (Perceived action options)	Right′, Left′, Stay′
$\mathcal{R}$ 4-tuple (Representation of E):	
$\mathcal{L}$ (Internal language: Vocabulary, Formation Rules . . .)	Predicate calculus
$\mathcal{B}$ (Data Base of state inputs)	$(D, \text{Right}') \longrightarrow C$
$\mathcal{U}$ (Data Base of value inputs)	$A > D$
$\mathcal{H}$ (Store of hypotheses: sentences + weights)	Closed curve has low value
λ ALGORITHM (Selects hypotheses, next internal state, records input, changes weights)	
$s^*(0)$ ELEMENT OF $\mathcal{S}$ (Starting internal state)	

Hypotheses: If $s = (x_1, y_1, x_2, y_2)$ and $a = \alpha$ then $s' = (2x_1, y_1 + 1, x_2 + \alpha, y_2)$
If $s = (M > 40, 150, 170, \text{High})$ and $a = (2000\,\text{cal}, 500, 300, 10, 0)$
then $s' = \cdots$

Fig. 1

promising approach to explaining the speed with which humans analyze scenes and recognize forms.

To explore the relative roles and utility of systems like vision, tactile sensing and construction (a torus = circle × circle) as an enrichment of the theory sketched in Fig. 1, we introduced the idea of an artificial "beetle" that is placed at a point on a curve or surface [6]. It can move over a discrete set of points that we call sites. It can drop flags at sites and thus sense if it has already been there. It can "see." It was shown that without "vision," the beetle could not detect monotonicity, peaks, convexity, or the absence of such properties of the curves on which it moves. Without the ability to "construct," it would not "imagine" a hollow spherical shell. Without the flag-dropping-sensing ability, it could not detect the closedness of curves in its environment.

Several other such results were obtained and several ideas for further mathematical investigation and implementation of algorithms were generated. In this paper, we report the results of these further investigations.

2. MAP-MAKING: AN ALGORITHM FOR DESCRIBING A SIMPLE ENVIRONMENT

We consider a "beetle" that is to form a "map" of the curve or surface on which it is constrained to move so that it can select and attain goals, i.e., desirable points on that surface or curve. It is to do this by forming and testing hypotheses about certain properties of the curve on the basis of what it "sees." The algorithm to form such maps has been implemented in FORTRAN and tested for very simple environments.

Environment

The beetle is constrained to move on a curve specified by $(x(t), y(t))$, where the integer-valued parameter t is interpreted as time. In the implemented program, t varies from 0 to 49, corresponding to at most 50 sites on the curve that the beetle can occupy in 50 successive instants.

The environments for the beetle may be described as follows. A particular environment consists of a finite collection of sites in a "space" that consists of simple closed curves joined by bridges, where the closed curves may have spars, e.g., as shown in Fig. 2.

Input

Since the sites on the curves are discrete, it is necessary to input a directed graph on the sites which gives orientation to the curves and adjacency between sites.

The "map" to be constructed is an array called SITE. It contains SITE(I) which represents the "view" from site I.

At each site a tangent is defined to the curve in the direction of orientation which determines the beetle's horizon. Thus, at a given time, the beetle's state is determined by

(i) its site at that time;
(ii) the tangent to the curve at the site (its visual horizon); and
(iii) relevant segments of the path it has taken to get to the site.

The visual input, SITE(I), consists of a sequence of coded projection-symbols of a ray from the curve onto a semicircle around site I, as the semicircle is scanned counter-clockwise. Each element of the sequence is one of the four codes: PP, if a simple projection is present; PA, if there is no projection; PNF, if there is a near-to-far discontinuity; and PFN if it is far-to-near discontinuity. This is illustrated in Fig. 3.

Procedure

First a graph traversing algorithm is used to get a decomposition of the environment into closed curves, bridges, and spars. For the sake of simplicity, we assume, in this

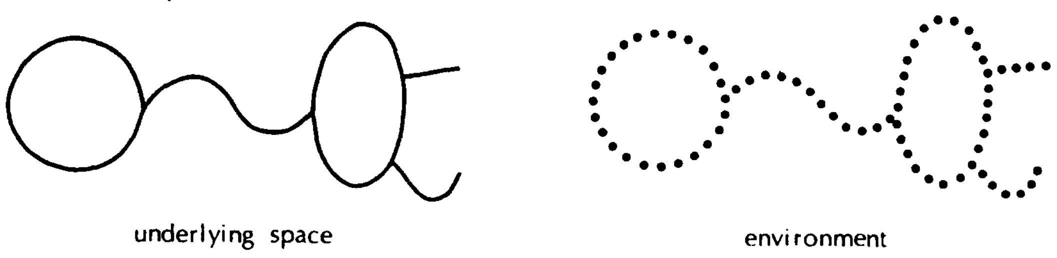


Fig. 2.

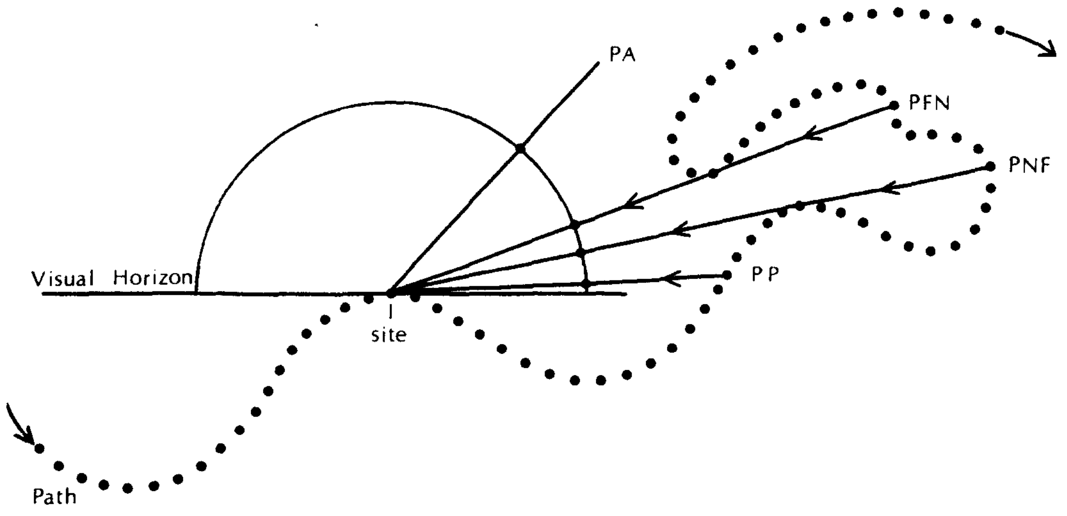


Fig. 3.

section, that the beetle already knows something of the general form that his environment may take, i.e., that it consists of closed curves, bridges, and spars, and it must determine the particular form. This algorithm is completely "tactile" given that the orientation on the environment induces a potential that the beetle can sense at each site (corresponding to the adjacency relation between sites). Then each segment is checked for concavity and convexity.

The fixed predicates are closed curve, open curve, bridge, spar, convex, concave, and straight (= convex and concave). The output is a verbal description of the environment in terms of these fixed predicates. The program is structured as shown in Fig. 4.

The environment section reads input consisting of the underlying directed graph, and also computes the sites and the angles of tangent and sight vectors at the sites. The traversing algorithm isolates a particular component of the environment, identifies it as closed curve, bridge, or spar, and then the component is tested for concavity or convexity by the visual processing section. This step is repeated until all components of the environment have been isolated and analyzed. Then the description is printed. The psychological structure, which the program models, may be pictured as in Fig. 5.

The preprocessors filter "noise" from the visual and tactile input so that only

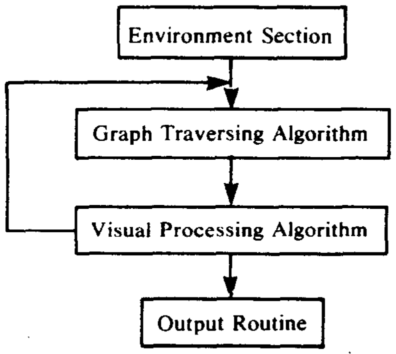


Fig. 4.

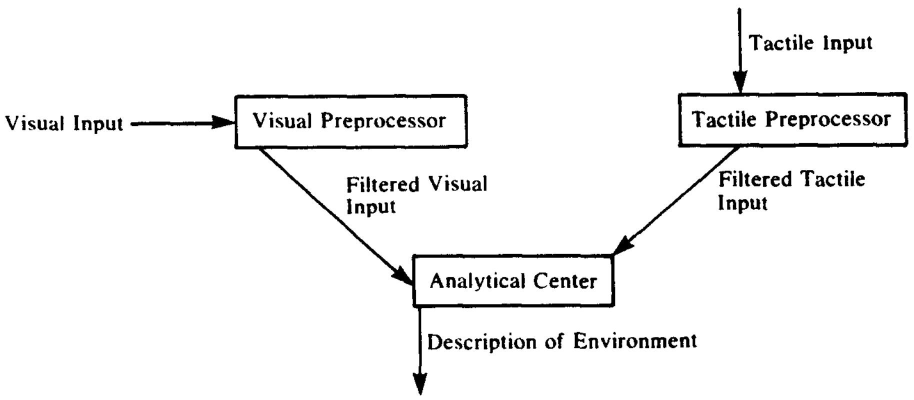


Fig. 5.

information pertaining to the particular segment of the environment that the beetle is on reaches the analytical center. The analytical center uses sensory input to generate a description of the environment.

3. LOCATING ONESELF ON A MAP

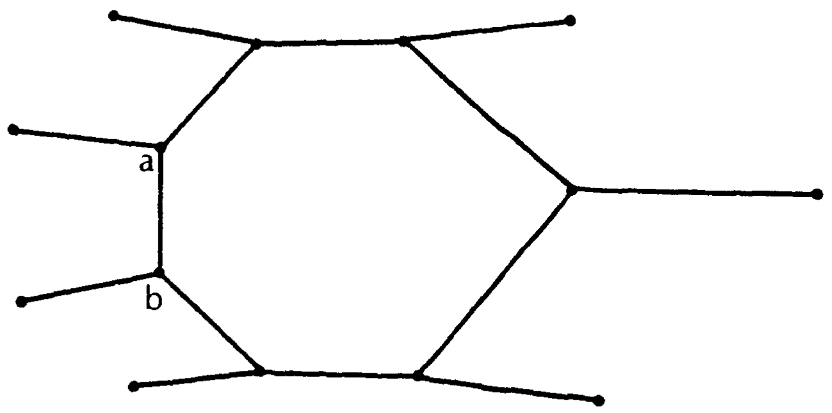
For the purposes of this section we regard an environment as a set of points, E, together with a set of relations $R_i^{\mathscr{E}}$ on these points. In other words, we consider a relational structure $\mathscr{E} = \langle E, R_i^{\mathscr{E}}\rangle$. A map of $\mathscr{E}$ is then another relational structure $\mathscr{M} = \langle M, R_i^{\mathscr{M}}\rangle$ whose relations reflect the relations of $\mathscr{E}$ in the sense that there is a function $f: E \to M$ such that $R_i^{\mathscr{E}}(a_1 \ldots a_n) \Leftrightarrow R_i^{\mathscr{M}}(f(a_1) \ldots f(a_n))$ for any $a_1 \ldots a_n \in E$. A learner (i.e., the bug or beetle) may be said to be able to locate itself on the map $\mathscr{M}$, at the location $\mathbf{a} \in E$, if $f(\mathbf{a})$ is the only element of M satisfying the same relations in $\mathscr{M}$ (modulo representation) as $\mathbf{a}$ satisfies in $\mathscr{E}$.

To find its position on a map $\mathscr{M}$ of $\mathscr{E}$, a learner begins at the point $\mathbf{a} \in E$ by finding all points $m_1 \ldots m_k \in M$ that satisfy all relations in $\mathscr{M}$ corresponding to those that $\mathbf{a}$ satisfies in $\mathscr{E}$. The points $m_1 \ldots m_k$ are the possible candidates for points representing $\mathbf{a}$. It then proceeds by matching successively larger sections of the environment against the map in order to reduce the number of candidates for points representing its location. This process terminates if the learner finds a point, $a' \in E$ with only one point, $m' \in M$ satisfying the relations corresponding to those that a' satisfies in $\mathscr{E}$, or if all points in E have been visited. In the latter case the sequence of matchings between E and M give all automorphisms of $\mathscr{M}$. The following proposition holds.

Proposition. A learner can locate himself on the map $\mathscr{M}$ at the location $\mathbf{a} \in E$, if and only if $f(\mathbf{a})$ is a fixed point of every automorphism of $\mathscr{M}$.

Proof. Suppose $f(\mathbf{a})$ is not a fixed point of every automorphism of $\mathscr{M}$. Then there is at least one other point $m' \in M$ that is indistinguishable from $f(\mathbf{a})$ according to all relations $R_i^{\mathscr{M}}$. Conversely if $f(\mathbf{a})$ is a fixed point of every automorphism of $\mathscr{M}$, any other point of M is distinguishable from $f(\mathbf{a})$ by relations holding in $\mathscr{M}$.

The content of this proposition, roughly speaking, is that the more symmetry there is in an environment, the more difficult it is to locate oneself on a map. Figure 6 illustrates this point. Of course, if an environment is highly symmetrical one may say that the problem of locating oneself loses importance since one might just as well be at one point



Points a and b are difficult to distinguish because of local symmetry

Fig. 6.

as any other. However, in an asymmetrical environment there may be local symmetries making it difficult to locate oneself exactly on a map.

4. GAUSS CODES

Various algorithms have been given for traversing and mapping graphs or mazes that are known as myopic algorithms because they are computable by diameter limited machines [7]. Most of these are based on a principle similar to that of Tarry [8]. We will describe an algorithm that in concert with Tarry's labyrinth traversing algorithm enables the myopic beetle to compute a particular property of his environment, that of being a "normal closed curve."

Define a closed curve to be normal if it has only finitely many self-intersections and these are transverse double points. The Gauss code of such a curve is the word formed by labelling the intersection points and recording these labels in the other in which they are traversed. It is known that the Gauss code of a curve and another minor condition determine the curve up to a "sense-preserving" homeomorphism of the plane [9]. The first characterization of these words that has purely combinatorial content was obtained recently by Lovasz and Marx [10]. We present an algorithm based on their characterization and its implementation in SNOBOL.

Let $w = A\alpha A\beta = AA_1 \dots A_kAB_1 \dots B_j$ be any word in which the letter A occurs exactly twice. Define the *vertex* split of w at A to be the word $\alpha^{-1}\beta = A_k \dots A_1B_1 \dots B_j$. Define the *loop removal* of w at A to be the word obtained from w by removing A and all occurrences of the letters in α. Then a *subword* of w is any word formed from w by a sequence of vertex splits and loop removals.

THEOREM. A word w is the Gauss code of a normal closed curve if and only if each letter in w occurs exactly twice and w has no subword of the form $A_1 \dots A_nA_1 \dots A_n$, where n is even [10].

The program to determine whether a word is a Gauss code of a normal closed curve (NCC) is given in the Appendix. It is assumed for simplicity that each letter in the word to be tested occurs exactly twice. Input to the program are the string OCCUR that contains each letter in the string to be tested, and the actual string to be tested. Output, for example, testing the string DABCABCDEE is

```
DABCABDCEE

VSA

DCBBDCEE

VSB

DCDCEE

FORBIDDEN SUBSTRING ENCOUNTERED

STRING IS NOT NCC
```

The algorithm generates a recursive tree search for the forbidden substring and may be charted as shown in Fig. 7.

The Gauss code characterization shows that a myopic beetle can deduce certain global properties of its environment without viewing it from the outside. A SNOBOL program for the Gauss codes problem is appended, to show how well suited SNOBOL is for this purpose. Some interesting questions pertaining to Gauss codes follow:

1. Can a more efficient algorithm be devised for testing whether a word is the Gauss code of an NCC? What is the computational complexity of the Gauss code problem?
2. How many words with n letters are the Gauss codes of normal closed curves?

CONCLUSIONS

Our purpose has been to model on the computer the process of constructing representations of an environment, and to investigate under what conditions these representations may be used to locate oneself in that environment. In doing this we have found that ease or difficulty of locating oneself on a map is a function of symmetry in the environment. We have also uncovered certain global properties which may be serially computed without viewing an environment from the outside, namely that of being a normal closed curve, and also the decomposition of an environment into closed curves, bridges, and spars. Doubtless many other examples of such properties may be found. Much of the discussion in *Perceptrons* [11] deals with the limitations of parallel computation, and we have been able to serially compute with impunity many of the properties which are not perceptron-computable, such as connectedness. This points to an ideal model of a learning machine as a combination of serial and parallel machines.

In terms of the long-term program outlined in the introduction, we have clarified the relationship between an external environment and its representation internal to a learner. This is a necessary step, for ultimately the formation of hypotheses about an environment depends on its internal representation; the hypotheses thus formed may be tested and the results of this testing used to obtain more exact representations.

It is this feedback cycle that embodies the process of a learner coming to terms with his world.

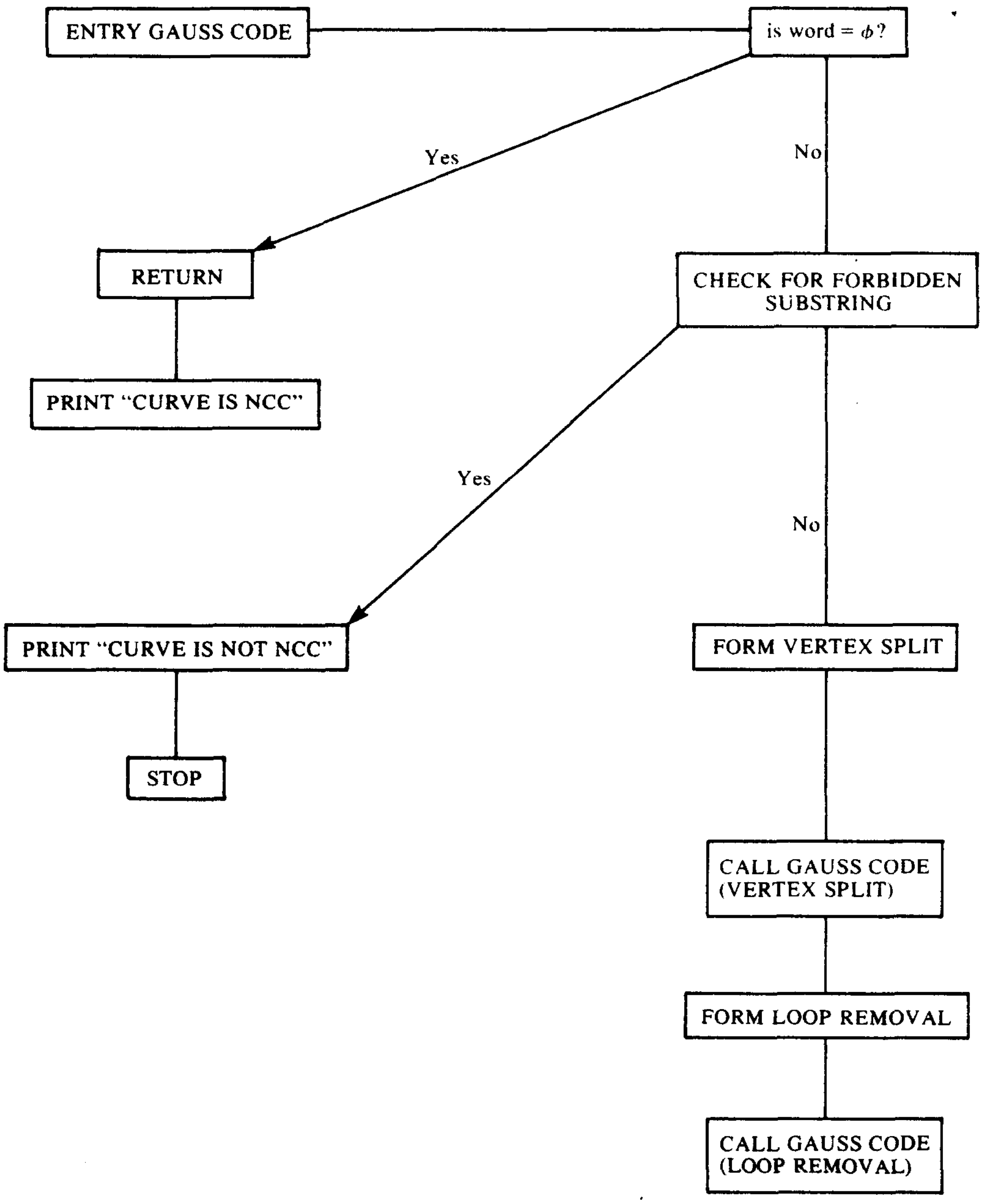


Fig. 7

REFERENCES

1. M. Kochen, Cognitive mechanisms. RAP-3, IBM Research Center, Yorktown Heights, NY (1960).
2. M. Kochen, Representations and algorithms for cognitive learning. *Artif. Intell.* 5, 199–216 (1974).
3. S. Hantler and M. Kochen, ASP: A program using stored hypotheses to select actions. *J. Cybernet.* 3, 1–12 (1973).
4. M. Kochen, An information theoretical model of organizations. *Trans. IRE PGIT-4*, 67–75 (1954).
5. M. Minsky, A framework for representing knowledge, in P. H. Winston, ed., *The Psychology of Computer Vision*, McGraw-Hill, New York (1975).

6. M. Kochen, On learning global aspects of shapes, in Proceedings of the Milwaukee Symposium on Automatic Computation and Control, pp. 91–96 (1976).
7. P. Rosenstiehl, J. R. Fiksel, and A. Holliger, Intelligent graphs, in R. C. Read, ed., *Graph Theory and Computing*, pp. 219–265, Academic Press, New York (1972).
8. G. Tarry, Le problème des labyrinthes. *Nouvelles Annales de Math.* **14**, 187 (1895).
9. H. M. Gehman, On extending a continuous (1–1) correspondence of two plane continuous curves to a correspondence of their planes. *Trans. Am. Math. Soc.* **28**, 252–265 (1926).
10. L. Lovász and M. Marx, A forbidden substructure characterization of Gauss codes. *Bull. Am. Math. Soc.* **82**, 121–122 (1976).
11. M. Minsky and S. Papert, *Perceptrons*, MIT Press, Cambridge (1969).

APPENDIX

Gauss codes program

```
               DEFINE('REVERSE(STRING1)','REV')
               DEFINE('LOOP_REMOVAL(WORD1,CH1)','L_R')
               DEFINE('VERTEX_SPLIT(WORD2,CH2)','V_S')
               DEFINE('GAUSS_CODE(WORD3,LIST)LIST2,LETTER','G_C')
               DEFINE('SUB_STRING(WORD4)','S_S')
               &TRIM = 1
               OCCUR = INPUT
               GAUSS_CODE(INPUT,OCCUR)   :S(NCC)
               OUTPUT = 'CURVE IS NOT NCC'   :(END)
NCC            OUTPUT = 'CURVE IS NOT NCC'   :(END)
REV            STRING1 LEN(1) . X1 =    :F(RETURN)
               REVERSE = X1 REVERSE        :(REV)
L_R            WORD1 (CH1 BREAK(CH1)) . DELIST = :F(RETURN)
               OUTPUT = 'LR' CH1
S2             WORD1 ANY(DELIST) =   :S(S2)
               LOOP_REMOVAL = WORD1   :(RETURN)
V_S            WORD2 BREAK(CH2) . W1 CH2 BREAK(CH2) . W2 CH2 = :F(RETURN)
               OUTPUT = 'VS' CH2
               VERTEX_SPLIT = W1 REVERSE(W2) WORD2   :(RETURN)
S_S            WORD4 LEN(1) . Y1 =   :F(RETURN)
               WORD4 BREAK(Y1) . Z1 Y1 REM . Z2   :F(S_S)
               Z2 POS(0) Z1     :F(S_S)
               N = SIZE(Z1)
               OUTPUT = NE(N / 2 * 2,N) 'FORBIDDEN SUBSTRING
                    ENCOUNTERED'  : S(RETURN)F(S_S)
G_C            OUTPUT = WORD3
               IDENT(WORD3,'')   :S(RETURN)
               SUB_STRING(WORD3)   :F(FRETURN)
               LIST2 = LIST
NEW_LTTR       LIST2 LEN(1) . LETTER = :F(RETURN)
               GAUSS_CODE(VERTEX_SPLIT(WORD3,LETTER),LIST)  :F(FRETURN)
               GAUSS_CODE(LOOP_REMOVAL(WORD3,LETTER),LIST)  :F(FRETURN)
               :(NEW_LTTR)
END
```