Computer Physics Communications 51 (1988) 443–450
North-Holland, Amsterdam

A RULE BASED APPROACH FOR PATTERN RECOGNITION IN PLANAR GEOMETRIC FIGURES

L. ALVISI and R. ODORICO

University of Bologna, Department of Physics, Istituto Nazionale di Fisica Nucleare, Sezione di Bologna, I-40126 Bologna, Italy

Received 25 February 1988

An OPS5 expert system has been developed, allowing the recognition of an arbitrary subpattern in a complex planar geometric figure under analysis. For this sake a syntactic representation for images is used by the expert system as a relational data base. The expert system looks for consistent mappings of the set of topological elements/relations representing the given subpattern into a corresponding subset of the topological elements/relations representing the geometric figure under analysis. The basic core of the expert system consists of 30 OPS5 rules.

PROGRAM SUMMARY

Title of program: TOPREC

Catalogue number: ABDG

Program obtainable from: CPC Program Library, Queen's University of Belfast, N. Ireland (see application form in this issue)

Computer for which the program is designed and others on which it is operable: VAX/VMS, Apple Macintosh

Computer: VAX 8800; *Installation*: CINECA, Bologna

Operating system: VMS

Programming language used: OPS5

High speed storage required: depends on the complexity of the problem

No. of bits in word: 32

No. of lines in combined program and test deck: 847

Keywords: pattern recognition, OPS5, rule based system

Nature of physical problem
Recognition of subpatterns in planar geometric figures.

Method of solution
Rule based system.

Restriction on the complexity of the problem
With the present version some classes of geometric figures are excluded.

Typical running time
1.50 cpu s on VAX 8800 for the test run.

Reference
[1] L. Alvisi, University of Bologna report (1987), in preparation.

LONG WRITE-UP

1. Introduction

The use of data structures such as relational data bases becomes necessary when the information to deal with is interconnected by a complex structure of logical chains. In general, structures of this type hold explicit information about the objects they refer to and their mutual relationships. It is often difficult to use relational data bases efficiently by means of traditional procedural approaches because with them the programmer has to supply a detailed planning of all the steps needed to complete the desired task.

Rule based expert systems [1] try to exploit the richer quality of the data contained in a relational data base in order to decrease the information that must be supplied as external control. For this sake they offer a set of instruments which can be used to link the elementary relations contained in the data base into more complex ones and thus answer questions on interest to the user. The advantage that is obtained using such non-procedural systems is more and more significant as the correlations to be verified become more complex, thus making an algorithmic approach hard to outline. The features allowed by rule based systems have already proved to be of considerable importance in pattern analysis and recognition (like, e.g., in aerial photo interpretation [2]).

The rule based system presented here is intended to recognize an arbitrary subpattern within a planar geometric figure under analysis, called network. Each one of them is treated as a collection of feature elements and relationships arranged in a planar graph representation. The latter is used by the rule based system as a relational data base.

Vertices, edges and cycles are the feature elements of the planar graph representation.

A vertex is a point in a plane which is connected by lines with a least three other points lying in the same plane. An ordered pair of vertices (u, v) is an edge; u is called the tail of the edge, and v the head. If (u, v) is an edge, (v, u) is called its reversal. A sequence of edges, which starts and ends in the same vertex, is called a cycle. Actually, only two kinds of cycles are considered in the representation: i) the external cycle, made up of all the edges belonging to the external border of the graph, oriented anticlockwise; ii) internal, clockwise, cycles of minimum area, i.e. such that no other cycle can lie within the region delimited by them. The terminology is the same used in conventional graph theory [3].

At the present stage of development, application of the system is limited to patterns having graph representations with no open ends (i.e. vertices with degree 1), and no articulation vertices (i.e. vertices whose removal causes disconnection of the graph) in them and in their dual graphs (i.e. the graphs obtained by associating a vertex to each one of the internal cycles considered here, and an edge between such vertices to each edge in the original graph which is common to the corresponding internal cycles).

The process of searching a given subpattern within the complex network amounts to an attempt to prove that subgraphs of the network representation are isomorphic to the graph representing the subpattern. This problem is of great interest because of the large field of application of graphs, in physics and elsewhere, and is being actively investigated [4,5]. The OPS5 approach presented here allows for good efficiency and for increased flexibility in dealing with more general

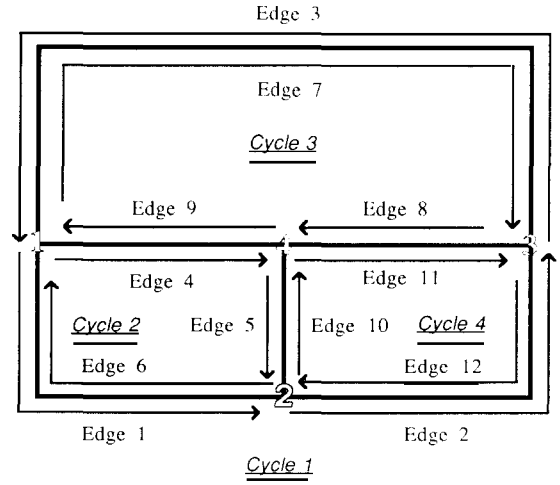

Fig. 1. Example of graph labelling.

patterns, for which the graph representation has further attributes appended to its elements. In fact, the rule based system can be easily extended to take into account such attributes as further constraints to be satisfied in the pattern recognition problem.

The system recognizes vertices, cycles and edges by numerical labels assigned to them. For each class of elements (e.g. edges) the label must range from 1 to the number of elements of the class. The external cycle must be labelled with 1. Apart from such restrictions, the labelling of vertices, edges and cycles can be arbitrary.

Fig. 1 shows a example of graph labelling. The label orderings appearing in this example and those of figs. 2 and 3 have been obtained from a FORTRAN procedure, developed by one of the authors [6], which can automatically extract the features of interest from the image, order them and arrange them in a format suitable for direct input to the system.

2. General features of the software

The production system consists of 51 OPS5 rules, but only 30 of them constitute its basic core, while the others are intended for input–output and diagnostic operations. Rules are organized in 11 classes (clusters), that correspond to the different phases the system has to go through to certify the presence of the subpattern in the geometrical figure under analysis. Within each class, the rules aim at the same task and work on the different aspects connected with its achievement. The program develops as a process of tasks creation, activation and satisfaction. Several tasks can be active at the same time during the program execution: the choice of the selected one depends on the strategy implemented in the inference engine. The MEA strategy, adopted in this program, makes the system particularly sensitive to the last activated task, and does not let it to be taken away from the considered goal. The program is entirely written in OPS5 for VAX computers [7]. Further information on OPS5 syntax, and the programming features of the language can be found in refs. [8,9].

3. Specific features of the software

The stack data structure has been largely used in the program, because OPS5's conflict resolution strategy has a built-in bias toward the most recently created (or modified) instances of an element class. A first stack is used to collect all the edges of the network that can be chosen as a suitable starting point for the process of recognition. The choice is made on the ground of considerations relating the head and the tail vertices of the edges of the network with the corresponding vertices of the first edge of the subpattern not belonging to the external cycle. In a second parallel stack are pushed the working memory elements that contain the hypotheses of correspondence between the cycle of the starting edge of the subpattern and the cycles whose edges have been pushed in the first stack. Then the process of recognition is achieved through an exploration of the cycles whose correspondence has been supposed. During such an exploration, starting from an appropriate edge, a step by step comparison is performed, parallel in both cycles, in order to check if it is possible to set up a correspondence between them. At the same time, new hypotheses of associations are generated, relating both the reversals of the considered edges and their cycles, creating in this way the premises for further explorations. The relations concerning edges or vertices which belong to the external cycle of the subpattern are carefully verified: in fact, information like the degree of the external vertices, or the number of edges joining two external vertices may be altered when the subpattern is embedded in the network.

The consistency of the set of relations and inferences produced during the process of exploration is continuously checked until one of the following occurs:

i) An inconsistency is produced: in this case all the relations developed up to that time are deleted.

ii) All the cycles have been explored and the recognition has been successfully completed: before being deleted, the associations developed during the process are saved in an output file.

In any case, the following step is the extraction

of two new elements from the first and the second stack in order to look for a further recognition from a different starting edge. The process ends when both stacks are empty.

4. Rule classes

a) Cluster for STARTUP actions (1 rule)

This rule is used to initialize the system. It sets the conflict resolution strategy used in the program, opens the input files and prepares the system to read them. It also sets the trace information to the lowest level to improve the program efficiency.

b) Cluster for READ actions (7 rules)

This class of rules is intended for input operations. Different rules are used to read information concerning vertices, edges and cycles for both the network and the subpattern.

c) Cluster for FIRST-EDGE actions (4 rules)

These rules set up the stacks that contain the information used to start the process of recognition. The first stack collects the edges of the network that can be associated with the starting edge of the subpattern. The second one contains the hypotheses of correspondence between the cycle of the starting edge of the supattern and the cycles whose edges have been pushed in the first stack.

d) Cluster for POP actions (2 rules)

The first rule extracts from the previously described stacks one of the couples of working memory elements that have been pushed in during the

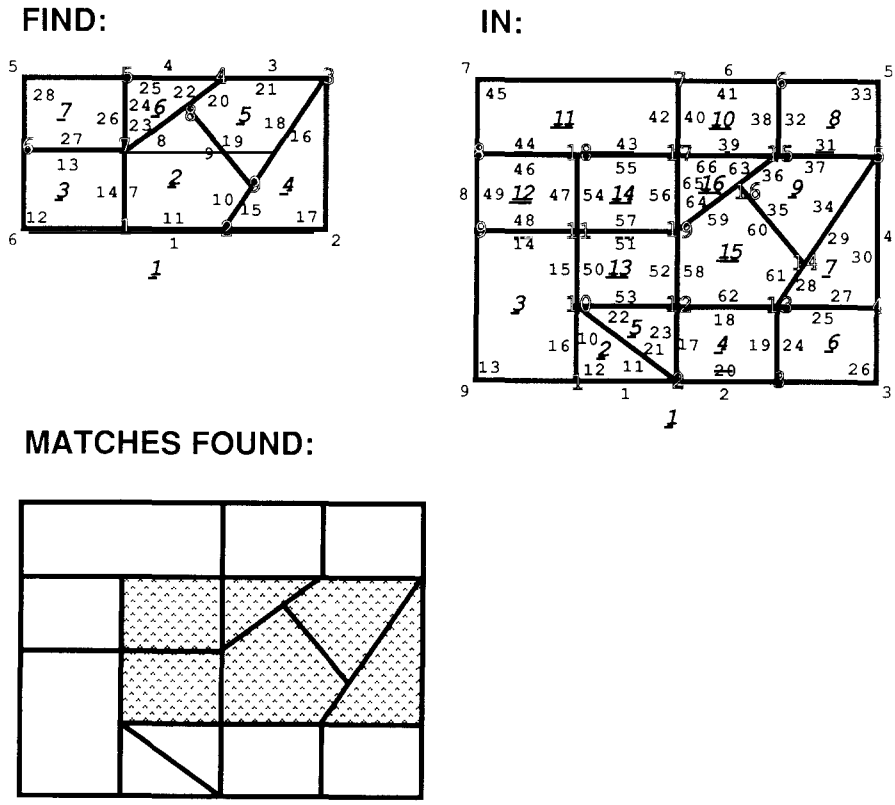

Fig. 2. Subpattern and network used in the test run. Matches found are shown.

phase of stack loading. The second one closes the file REPORT.DAT when the stacks are empty.

e) Cluster for ASSOC-CYCLES actions (6 rules)

These rules do a first check on an hypothesis of association between two cycles produced during program excution. They compare the length of the cycles (i.e. the number of edges which form the cycles) trying to find out as soon as possible an eventual inconsistency in the recognition process. If this first check is successfully passed, a new working memory element, containing the labels of the two cycles under analysis, is generated and put in a third stack, waiting for further tests.

f) Cluster for ASSOC-EDGE actions (8 rules)

The rules contained in this cluster verify if an hypothesis of association, between two edges, produced during program execution, clashes with the set of inferences generated up to that time. For this sake a comparison is made between the number of edges that leave from the tail and the head vertices of the two edges. The case of a subpattern edge belonging to the external cycle is also considered. Furthermore, a precondition for the association of the two edges is the existence of an association between the cycles they belong to.

g) Cluster for CYCLE-ACTIVE actions (3 rules)

These rules concern the couples of cycles that have passed the first check carried out by cluster E. They extract the last of the working memory elements from the stack described at point E; then they generate the control elements that will be used to perform the parallel exploration of the cycles.

h) Cluster for EXPLORE actions (4 rules)

These rules perform the parallel exploration of the two cycles extracted from the stack during the cycle-activation phase. Starting from their respective first edge, the two cycles are explored step by step, producing new hypotheses of association between corresponding edges in the two cycles. The internal cycles (i.e. the cycles which have neither edges belonging to the external cycle, nor edges whose reversal belongs to the external cycle) are explored following a clockwise motion. The exploration of the other cycles is instead controlled by two competing rules: one performs a clockwise step, the other an anticlockwise one. The system can change the direction of the motion during the exploration in order to solve the problems that raise along the external border.

Table 1
Data ordering in files containing graph representations

NE,EDGE-LABEL(K),REVERSAL(K),TVERTEX(K), HVERTEX(K),TDEGREE(K),HDEGREE(K), CYCLE-LABEL(K),CYCLE-LENGTH(K),CYCLEPE(K), CYCLENE(K), for K = 1 to NE)

Table 2
Meaning of symbols in table 1

NE	total number of directed edges of the graph
EDGE-LABEL	edge label
REVERSAL	label of the reversal of the edge
TVERTEX	label of the tail vertex of the edge
HVERTEX	label of the head vertex of the edge
TDEGREE	number of edges incident with the tail vertex of the edge
HDEGREE	number of edges incident with the head vertex of the edge
CYCLE-LABEL	label of cycle to which the edge belongs
CYCLE-LENGTH	length of cycle to which the edge belongs
CYCLEPE	label of the preceding edge in cycle
CYCLENE	lable of the following edge in cycle (clockwise order for internal cycles, anticlockwise order for the external cycle)

Table 3
Input file for the graph in fig. 1

12									
1	6	1	2	3	3	1	3	3	2
2	12	2	3	3	3	1	3	1	3
3	7	3	1	3	3	1	3	2	1
4	9	1	4	3	3	2	3	6	5
5	10	4	2	3	3	2	3	4	6
6	1	2	1	3	3	2	3	5	4
7	3	1	3	3	3	3	3	9	8
8	11	3	4	3	3	3	3	7	9
9	4	4	1	3	3	3	3	8	7
10	5	2	4	3	3	4	3	12	11
11	8	4	3	3	3	4	3	10	12
12	2	3	2	3	3	4	3	11	10

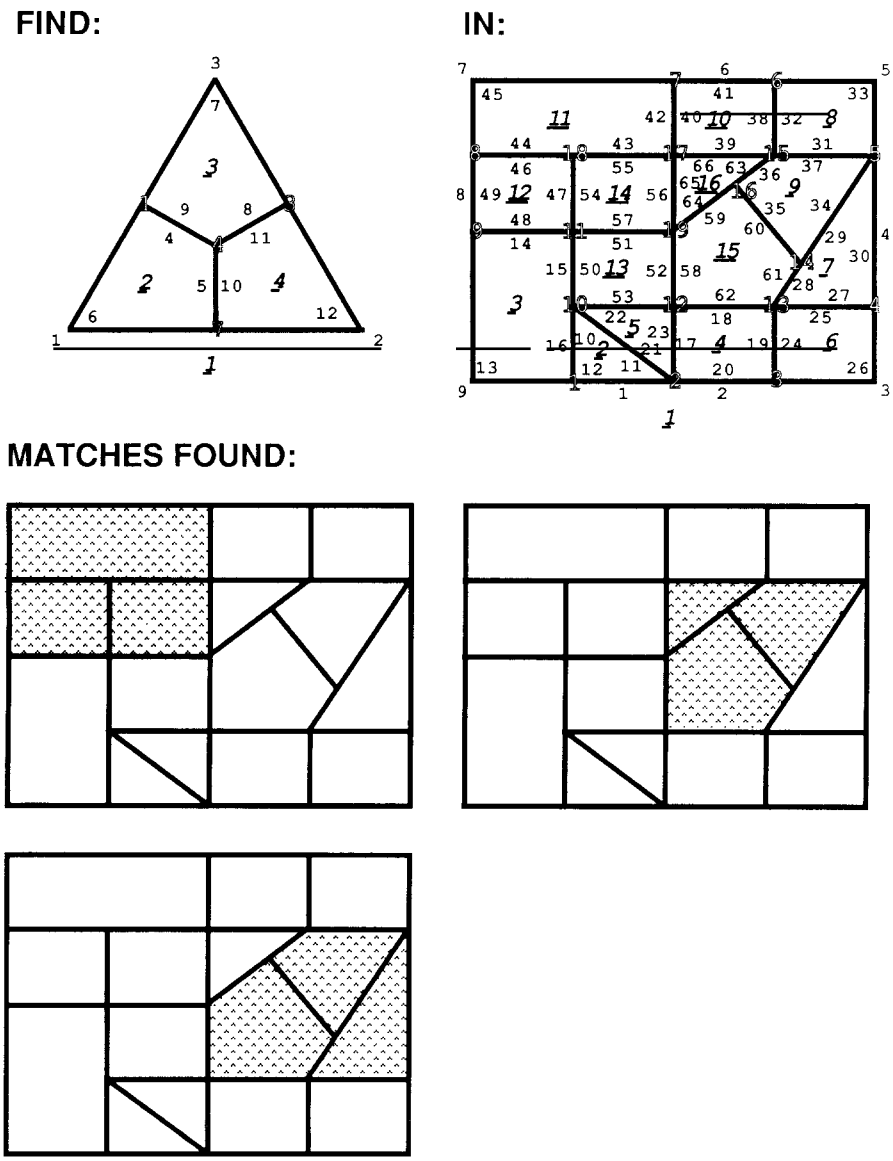

Fig. 3. Inscriptions of the graph of fig. 1 in the network.

i) Cluster for FAILURE / SUCCESS actions (4 rules)

These rules are activated when an error condition has occured, or when the recognition has been successfully completed. They provide for deleting all the working memory elements during the attempt just completed. Then, creating a task for the action POP, they prepare the system to a new exploration, beginning from a different edge.

j) Cluster for REPORT actions (7 rules)

These rules superintend the output operations. For each successful attempt of recognition, the whole set of the generated associations is saved in file REPORT.DAT. If not acceptable patterns are submitted to the system, diagnostic messages, including the labels of the vertices and edges that cause problems, are written in the file.

k) Cluster for NOT-ACCEPTABLE actions (5 rules)

This cluster contains the rules that recognize the input configurations that, at the present time, the system does not accept.

5. Description of the input data

Input data are read from two files. NETWORK.ABS contains the graph representation of the network, while SUBPATT.ABS contains that of the subpattern to be searched out in the network. Files are arranged as free format data sequences. Data points are separated by an arbitrary number of blanks. Table 1 shows how data are to be ordered in the input files. The meaning of the symbols is given in table 2. In table 3 an example of input file is presented, corresponding to the graph representation of fig. 1.

6. Test run

Fig. 2 shows the subpattern and the network used for the test run. Cycles are labelled with underlined bold numbers, vertices with outlined ones, edges with plain types. These numbers are referred in the test run output to report the associations between edges and cycles of the two graphs made by the OPS5 system. A graphic display of the only match found is also shown in the figure. A further example is presented in fig. 3, reporting on the inscription of the graph of fig. 1 in the same network as in fig. 2. Nine matches are found, each one of those shown in the figure actually representing three matches due to the symmetry of the subpattern.

References

[1] F. Hayes Roth, D. Waterman and D. Lenat, Building Expert Systems (Addison-Wesley, Reading, MA, 1983).
[2] D.M. McKeown Jr., W.A. Harvey Jr. and J. McDermott, IEEE Trans. Patt. Anal. Mach. Intell. PAMI-8 (1985) 570.
[3] F. Harary, Graph Theory (Addison-Wesley, Reading, MA, 1972).
[4] J.E. Hopcroft and R.E. Tarjan, ACM Commun. 30 (1987) 198.
[5] J.E. Hopcroft and R.E. Tarjan, J. Comput. Syst. Sci. 7 (1973) 323.
[6] R. Odorico, University of Bologna report, in preparation.
[7] Digital Equip. Co., OPS5 for VAX User's Guide (AA-BH99 A-TE) (1984).
[8] C.L. Forgy, OPS5 User's Manual, Digital Equip. Co. (1984).
[9] L. Brownstone, R. Farrel and E. Kant, Programming Expert Systems in OPS5 (Addison-Wesley, Reading, MA, 1985).

TEST RUN OUTPUT (input as in fig. 2)

```
MATCH NO. 1
-----------

CYCLE NO.  2  OF THE SUBPATTERN WITH CYCLE  NO. 15  OF THE NETWORK
CYCLE NO.  4  OF THE SUBPATTERN WITH CYCLE  NO.  7  OF THE NETWORK
CYCLE NO.  5  OF THE SUBPATTERN WITH CYCLE  NO.  9  OF THE NETWORK
CYCLE NO.  6  OF THE SUBPATTERN WITH CYCLE  NO. 16  OF THE NETWORK
CYCLE NO.  7  OF THE SUBPATTERN WITH CYCLE  NO. 14  OF THE NETWORK
CYCLE NO.  3  OF THE SUBPATTERN WITH CYCLE  NO. 13  OF THE NETWORK
EDGE  NO. 10  OF THE SUBPATTERN WITH EDGE   NO. 61  OF THE NETWORK
EDGE  NO. 15  OF THE SUBPATTERN WITH EDGE   NO. 28  OF THE NETWORK
EDGE  NO. 16  OF THE SUBPATTERN WITH EDGE   NO. 29  OF THE NETWORK
EDGE  NO. 18  OF THE SUBPATTERN WITH EDGE   NO. 34  OF THE NETWORK
EDGE  NO.  9  OF THE SUBPATTERN WITH EDGE   NO. 60  OF THE NETWORK
EDGE  NO. 19  OF THE SUBPATTERN WITH EDGE   NO. 35  OF THE NETWORK
EDGE  NO. 20  OF THE SUBPATTERN WITH EDGE   NO. 36  OF THE NETWORK
EDGE  NO. 22  OF THE SUBPATTERN WITH EDGE   NO. 63  OF THE NETWORK
EDGE  NO.  8  OF THE SUBPATTERN WITH EDGE   NO. 59  OF THE NETWORK
EDGE  NO. 23  OF THE SUBPATTERN WITH EDGE   NO. 64  OF THE NETWORK
EDGE  NO. 24  OF THE SUBPATTERN WITH EDGE   NO. 65  OF THE NETWORK
EDGE  NO. 26  OF THE SUBPATTERN WITH EDGE   NO. 56  OF THE NETWORK
EDGE  NO. 27  OF THE SUBPATTERN WITH EDGE   NO. 57  OF THE NETWORK
EDGE  NO. 13  OF THE SUBPATTERN WITH EDGE   NO. 51  OF THE NETWORK
EDGE  NO. 14  OF THE SUBPATTERN WITH EDGE   NO. 52  OF THE NETWORK
EDGE  NO.  7  OF THE SUBPATTERN WITH EDGE   NO. 58  OF THE NETWORK
```