TOWARD COMPUTER-AIDED INDUCTION: A BRIEF REVIEW OF CURRENTLY IMPLEMENTED AQVAL PROGRAMS

by

Ryszard S. Michalski

Report No. 874, Department of Computer Science, University of Illinois,
Urbana, Illinois, May 1977.

UIUCDCS-R-77-874

UILU-ENG 77 1726

Toward Computer-Aided Induction: A Brief Review of Currently Implemented AQVAL Programs

by

Ryszard S. Michalski

May 1977

Toward Computer-Aided Induction:

A Brief Review of Currently Implemented AQVAL Programs

by

Ryszard S. Michalski

Department of Computer Science
University of Illinois
Urbana, Illinois

May 1977

This work was supported in part by the National Science Foundation, Washington, DC, under grant no. NSF MCS 74-03514.

TOWARD COMPUTER-AIDED INDUCTION:

A brief review of currently implemented AQVAL programs

Ryszard S. Michalski
Department of Computer Science
University of Illinois, Urbana, Ill. 61801

ABSTRACT

The paper reports on a set of computer programs developed at the University of Illinois, Champaign-Urbana, whose purpose is to aid in solving certain classes of inductive tasks. The programs can be used, in particular, for determining most economical (according to a user specified criterion), generalized and discriminant description of given sets of data. The inferred (and initial) descriptions are constructed of various logical and set-theoretic operators (such as "not", "and", "or", set membership, "climbing a generalization tree", quantifiers, equivalence), multiple-valued variables (originally defined or new ones generated by the program), k-place predicates and k-place functions. The underlying formalism for expressing descriptions are variable valued logic systems VL_{21} (a form of a first order predicate logic with additional operators) and VL_1 (a form of a multiple-valued propositional calculus).

Among described programs there is also a program for selecting "most significant" variables from a large number of potentially useful variables (for a given decision problem), a program for selecting "most representative" events from a large set of learning events and a program for testing VL_1 hypotheses and calculating detailed performance statistics.

Descriptive terms : inductive inference, computer-aided induction, machine learning, generalization techniques, variable-valued logic, knowledge-based systems, production systems.

INTRODUCTION

Recently an interest in implementing computer programs which can do various forms of inductive inference has been rapidly growing. This seems to be due in part to a growing need for such programs, and in part to better understanding of how to build them.

One of the important potential applications of inductive programs is in knowledge-based systems. The success of first built knowledge-based systems indicates that they have a potential for a wide practical application in the future. As a consequence, there is a need for developing new more efficient ways of introducing knowledge into machines. Inductive programs could be useful for this purpose in a number of ways. They could, for example, determine production rules from specific examples of decisions or transformations , optimize a given body of rules (by joining a few specific rules into one more general rule or by detecting unneccessary conditions), to create "rule models" which compactly describe a given body of rules (and can be useful for identifying missing information or errors in new rules (Davis 76)), to automatically correct rules in view of new contradictive information, etc.

Another potential application of inductive programs is to aid specialists working in applied sciences, e.g., biology, plant pathology, physiology, medicine, etc., in formulating hypotheses explaining a body data, in detecting patterns in complex numerical or non-numerical data, in suggesting alternative hypotheses, in selecting most relevant variables describing data, in partitioning data into meaningful clusters (which employ more complex relationships between data items than traditionally used distances), etc.

In recent years a number of papers have been published which contributed many significant ideas about how to build computer programs which can do certain specific inductive tasks or how to formulate a theoretical basis for induction (e.g., Winston 70, Simon and Lea 73, Buchanan 74, Michalski 74, Waterman 74, Zagoruiko 74, Hayes-Roth and McDermott 75, Kochen 75, Morgan 75, Solomonoff 75, Vere 75, Larson 76, Lenat 76). There also have been studies on how people do induction (e.g., Kotovsky and Simon 73).

Relatively little, however, has been done on building efficient and simple to use computer programs, which have sufficient generality and flexibility that could aid a non-computer specialist in solving certain classes of practical inductive tasks.

This paper briefly describes a few computer programs which represent our current results toward such a goal . The programs can be used as tools in determining the simplest (in some sense) or most economical descriptions of sets of data. The descriptions (original or inferred) can involve various logical or set theoretic operations (such as "not" , "and", "or", set membership, "climbing a generalization tree", quantifiers, equivalence) , and multiple-valued variables (both, originally given or new ones which program generates), k-place predicates and k-place functions. An important advantage of the descriptions inferred by the programs is that they have a very simple, direct conceptual interpretation, and also display high computational efficiency when used for deductive inference (particularly, VL_1 descriptions) .

Since the set of operators used in programs is quite limited, the forms of descriptions or hypotheses which programs can infer are also quite limited. On the other hand, due to the use of such very basic operators (which theoretically can describe every discrete transformation) the programs display a significant generality and flexibility. Consequentely, if the type of descriptions the programs can infer are interesting for some practical task, then the programs can be quite useful, since they can combat vast combinatorial complexities often occuring in inductive tasks and difficult for a human. Thus, their role can be compared to a role of an efficient linear programming program which is of practical value for someone whose problem can be formulated in terms of linear programming.

For the lack of space, we will give here only a brief description of the major function performed by each program. An interested reader or potential user is referred to the appropriate sources for details.

PROGRAMS

1. AQVAL/2 -v1 (briefly, VL2) (Larson, Michalski 77)

- a program for generalization and optimization of VL_{21} production rules.

The function of the program is to transform a given set of VL_{21} production rules into a new set of rules which are more general and optimal (or quasi-optimal) with regard to a user specified criterion of optimality or "simplicity" and which satisfy various restrictions and properties characteristic of a given task domain. The VL_{21} production rules (briefly VL rules) are rules in a form:

CONDITION --> DECISION

where CONDITION is a formula in variable-valued logic system VL_{21} and DECISION denotes a decision or an action to be assigned to an object (or situation) which satisfies the CONDITION.

Since there is not enough space for defining the VL_{21} system here, we will only provide an example a VL production rule, just to give a basic idea of the formalism to reader who is unfamiliar with the system (for a description of the system see Michalski 74, Larson and Michalski 77). An example of a VL rule:

$$\exists x_2, x_3 ([q(x_1,x_2,x_3)=0,2][x_1=2..6] \vee \forall x_4, x_6 [r(x_1,c) > 2][s(x_4) \; \& \; t(x_6)= \text{same}]$$
$$==> [\text{decision}= d]$$

The rule is interpreted : if in the given situation (e.g., in a data base or in a description of an object) there exist values of variables x_2 and x_3 such that function q for given value of x_1 , and the values of x_2 and x_3 takes value 0 or 2 and x_1 has value between 2 and 6, inclusively, or the function r for given value of x_1 and the constant c takes value greater than 2 and functions s and t are equal for every value of x_4 and x_6, then assign to the given situation decision d.

The VL_{21} system is a subset of the VL_2 system which is a form of a many valued logic system, developed to combine the precision of a logic system with a facility to express compactly and in an almost self-explanatory way descriptions of significant logical complexity and varied degree of generality. VL descriptions can be easily used both for induction and deduction (a reader is advised to compare the readability of the above VL rule and of an equivalent expression in the first order predicate logic).

The input to the program consists of:

a) data rules, which is a set of VL production rules describing a given situation or an object (for example, a set of rules which relate a description of a group of patients to a diagnosis assigned to them),

b) a problem environment description, which contains VL rules describing properties of predicates or functions involved in data rules (e.g., that a given predicate is transitive),

c) definitions of the generalization structures of structured descriptors, i.e., definitions of hierarchies of concepts related by different degree of generality (e.g., that a triangle or rectangle is a special case of a polygon),.

d) a criterion of optimality which specifies what properties of the goal rules are most desired by the user (e.g., minimum number of rules, minimum cost of descriptors used in the rules , etc.).

The input data can be specified to the program either in an interactive or a batch mode. The goal of the program is to produce a new set of data rules which are more general than initial ones and optimal (or quasi-optimal) with regard to a user defined criterion (where the optimality criterion may require a computational efficiency of the rules, or economy of measuring the information needed for rule evaluation, conceptual simplicity, etc.). Also, the new rules will satisfy various restrictions and take advantage of properties characterizing descriptors, which are defined in the input data as specific to the given problem. In generalizing the initial rules, the program uses certain generalization rules which take into consideration the types of descriptors or variables which are used to describe objects. The program distinguishes between nominal, ordinal and structured descriptors (interval and ratio variables can also be used, but they will be treated as ordinal variables). The program is written in PASCAL, and therefore is easly transportable to other computer instalations.

To illustrate the program's function we will use a very simple inductive task. Given are three classes of objects O_1, O_2 and O_3 (fig.1). Find a possibly simple description of each class of objects, such that each description describes all the objects in the given class and none of the objects in other classes.

It may be interesting for a reader to try to solve this problem him/her-self before looking for the solution found by the program (APPENDIX). The program was given descriptions of all objects in terms of such concepts as shape of a part of an object, texture of a part, its size, relations between parts, e.g., that one part is on top of another , etc. Although this problem is quite simple, the author observed that some people have difficulties with finding descriptions of classes O_1 and O_2 if they first found a description of class O_3. This seems to be due to the psychology of human problem solving, namely, when people find some concept useful for one problem (in this case, texture), then they try to use it again for solving another similar problem, but in this case such a heuristic plays against them . A trap of this kind can be avoided, if one would use the program for solving the problem. This is yet another argument for possible usefulness of inductive programs.

A potential for practical applicability of the program can be better appreciated if one observes that the program can handle in a resonable time (say, 1 to 3 minutes on a Cyber 175) a problem with a few dozen classes, with a dozen or so objects in each class and a few dozens descriptors (variables, predicates or functions) which constitute the initial vocabulary of the program. Concluding, it may be worthwhile to notice that the production rules used in various knowlegde-based systems , e.g., MYCIN (Shortliffe 74), can be easily represented as VL rules, and therefore the program could find an application in designing such systems .

2. AQVAL/1 -v7 (briefly, AQ7) (Larson, Michalski 75)

- a program for determining optimal or simplest (according to a certain criterion of optimality or simplicity) VL_1 descriptions of sets of VL_1 events.

VL_1 descriptions are expressions in the VL_1 logic system (Michalski 74) which is a subset of VL_2 (there are no quantifiers, no k-place functions or predicates, certain forms are not allowed). If a VL_1 expression is used both as a condition and a decision part of a production rule, then the rule is called a VL_1 production rule. Here is example of a VL_1 production rule:

$$[x_1 = 1,2][x_4 \neq 2][x_5 = 2..5][x_6 = 0] ==> [x_{10} = 2]$$

(if x_1 equals 1 or 2, x_4 not equal 2, x_5 has value between 2 and 5 inclusive, x_6 equals 0, then x_{10} is assigned value 2)

A restricted version of this program (written in PASCAL) is used as a subroutine in VL_2 program, and can be used directely within the VL_2 program. The input to the program consists of:

A. data sets, which are sets of VL_1 events, each set is associated with a certain decision. VL_1 events are descriptions of objects in the form of a list of values of certain descriptors (variables). Each descriptor can take an arbitrary number of discrete values.

B. a criterion of optimality or "simplicity" for VL_1 descriptions to be inferred. A user defines the most preferable criterion (for a given problem) in the form of a sequence of subcriteria ordered according to their priority (the subcriteria are selected from a list of available subcriteria).

C. control parameters, which tell the system what kind of rules are desirable , what should be the scope of search (in order to control the memory and computational time to be spent on the problem), and the definitions of the domains of input variables.

The program can produce 3 kinds of rules:

1) linearly ordered rules, which have a property that an event can satisfy a rule i, only if it does not satisfy any of the previous rules i-1, i-2, ..., 1.

2) unordered disjoint rules, which can be satisfied by an event in any order and no event can satisfy more than one rule.

3) unordered intersecting rules, which also can be satisfied in any order but there may exist events which satisfy more that one rule.

The goal of the program is to produce optimal (or quasi-optimal) with regard to the given criterion, VL_1 descriptions of the input event sets.

The program was written in PL/1 and can handle in a few minutes of computation on an IBM 360/75 a few dozens of event classes, each containing a hundred or so events involving a few dozen multiple-valued variables. There is also available an IBM 360/75 assembly version of this program which runs an order or so faster than the decribed obove PL/1 version (Yalow 76).

3. AQVAL/1 - v11 (briefly AQ11)   (Larson 76)

- a program for multi-step formation of VL_1 hypotheses.

The program accepts as input data VL_1 decision rules and a set of VL_1 events with an associated class name. If any of the events are classified by the rules incorrectely, the program modifies the rules to make them consistent with these events. If now some new events contradict the current rules, a new iteration of the process begins.

The following simple example illustrates one iteration of the program. Suppose DR_1 and DR_2 are the initial hypotheses:

$$DR_1: \quad [x_1 = 0,1] ==> [d=1]$$
$$DR_2: \quad [x_2=2][x_3=1] ==> [d=2]$$

Suppose we are now given 2 events associated with decision d=2:

$$e_1: (0,0,1) ==> [d=2]$$
$$e_2: (1,3,1) ==> [d=2]$$

These events satisfy, however, the first rule instead of the second. The program will modify the rules to make them consistent with the new events (and also with any old events which it has in memory). A result of such modification might be:

$$DR_{11}: \quad [x_1=0,1][x_3=0] ==> [d=1]$$
$$DR_{21}: \quad [x_3=1] ==> [d=2]$$

The program was written in PL/1 (for IBM 360/75) and can handle in a few minutes of machine time a few dozen of original hypotheses and a few hundred events, which can involve a few dozen of variables, each able to take up to 8 different values.

The program also includes a procedure (which can be used as a separate program) for testing VL_1 hypotheses (formulas) on a given set of events and for calculating detailed performance statistics.

4. AQVAL/1 - v9 (briefly, AQ9) (Cuneo 75)

   - a program for optimization of a set of VL_1 formulas.

This program (written in PL/1) can accept as input a set of VL_1 formulas (rather than only VL_1 events as AQ7 and UNICLASS). Each formula is associated with certain decision class. An important feature of this program is that input formulas associated with different classes may intersect. The program optimizes the input formulas with regard to a user defined criterion. Similarly to AQ7, the program can infer ordered rules, unordered intersecting rules and unordered disjoint rules. The program can also be used for determining an optimal logically equivalent (i.e., not generalized) description of a single class of objects. The price for having the above features is that this program is less efficient than AQ7.

5. UNICLASS - RS (Stepp 76)

   - a program for determining an optimal (or quasi-optimal) and generalized (or logically equivalent) description of a single set of VL_1 events.

This program differs from the previous programs in that there is only one set of events to be described, and consequently , a degree of generalization has to be limited by other means that the condition of non-intersection with other event sets (i.e., there are no "negative" examples). The degree of generalization is limited by a special input parameter, called a "density threshold".

6. AQPLUS (Forsburg 75)

- a program for an interactive determination of a descriptor set (a "feature selection" program).

This program serves as an aid to select most relevant variables (features) for a given classification problem. Suppose there are 150 potentially relevant variables. Running any of the above AQVAL programs with such a large number of variables may require too much time or memory. Standard feature selection techniques developed in pattern recognition could be used here, but they are not very satisfactory, because they select variables based on a measure of usefulness of each variable taken separately.

This program initially selects a random subset of original variables . An AQVAL/1 program is run using these variables as descriptors for a given problem (a version of program used here can handle continuous variables) . Depending on a role each variable plays in the obtained VL_1 descriptions (whether it is present or not, in how many terms it occurs, etc.), an appropriate "usefulness" value is assigned to it. In the next iteration a new random subset of variables is selected , but the probability with which any given variable can be selected depends now on the "usefulness value" assigned to it. The subset of variables selected in each iteration tends to stabilize after a certain number of iterations. Such a subset is then taken as the representation space for the problem.

The program (written in FORTRAN) is interactive and at each iteration a user can change the "usefulness value" of any variable or add or remove a variable from a selected by the program subset.

7. ESEL (Michalski 75, Larson, Michalski 77)

- a program for selecting most representative events from a large learning set.

This program serves as an aid to select from a very large number of VL_1 learning events (i.e., events for which the class they belong to is known) a subset of the "most representative events". For example, the initial set of events may be of size of a few thousand, and the selected subset will be of size, say, a few dozen. Such a small set of events can now be easilly handled by the AQVAL/1 program.

APPENDIX

The VL2 program found the following descriptions of the classes from fig.1:

O_1: $\exists$ part([is-top(part)][shape(part)= polygon][texture(part)=blank]
(in objects of class O_1 the top part is an unshaded polygon)

O_2: $\exists$ part([is-top(part)][shape(part)=oval]
(in objects of class O_2 the top part is an oval)

O_3: $\forall$ part[texture(part)= shaded]
(texture of every part is shaded)

or, alternatively,

$\forall$ part[texture(part)= same]
(every part has the same texture)

ACKNOWLEDGMENTS

The research described in this paper was supported in part by National Science Foundation, Grant NSF MCS 74-03514.

The author expresses sincere thanks to the following people who contributed to the development of the programs described in this paper : R. Ph. Cuneo, S. Forsburg, J. Larson, , R. Stepp, Val Tareski and Ed Yalow.

Thanks are also due to James Larson for many discussions, suggestions, and a proofreading of the paper.

The author wrote this paper during his sabbatical at the Department of Computer Science at Carnegie-Mellon University. The hospitality of the Department and a permission to use its exellent editing facilities are gratefully acknowledged.

REFERENCES

Buchanan B. G., Scientific Theory Formation by Computer, NATO Advance Study Institute on Computer Oriented Learning Processes, Bonas, France 1974 (in the book: Computer Oriented Learning Processes, Noordhoff-Leyden 1976)

Cuneo R. Ph., Selected problems of minimization of variable-valued logic formulas, Master's Thesis, Department of Computer Science, University of Illinois, Urbana, 1975

Davis R., Applications of meta level knowledge to the construction maintenance and use of large knowledge bases, Computer Science Department, Report no.STAN-CS-76-552, Stanford University, July 1976

Hayes-Roth F. and McDermott J., Knowledge Acquisition from Structural Descriptions. Departamental Report, Department of Computer Science, Carnegie-Mellon University, Pittsburgh 1976

Forsburg S., A user's guide for AQPLUS, an internal report, Department of Computer Science, University of Illinois, Urbana, 1975

Kochen M., An Algorithm for forming hypotheses about simple functions, Third Milwaukee Symposium on Automatic Computation and Control, Miwaukee, Wisconsin 1975

Kolovsky K. and Simon H. A., Empirical Tests of a Theory of Human Acquisition of concepts fo sequential patterns, Cognitive Psychology 4, 1973

Larson J., A multi-step formation of variable-valued logic hypotheses, Sixth International Symposium on Multiple-Valued Logic, Logan ,Utah, May 1976 (User's guide of AQ11 is described in: AQ11: a program for multi-step formation of VL_1 hypotheses and evaluation of VL_1 formulas, Department of Computer Science, University of Illinois, Urbana 1977).

Larson J., Michalski R.S., AQVAL/1 (AQ7): User's guide and program description, Report No. 731, Department of Computer Science, University of Illinois, Urbana, Illinois, June 1975

Larson J., Michalski R.S., Inductive inference of VL decision rules, Workshop on Pattern Directed Inference Systems, Hawaii, May 1977.

Larson J., Michalski R.S., A user's guide for ESEL: A VL_1 event selection program, internal report, Department of Computer Science, University of Illinois, Urbana, Illinois, February 1977

Lenat D., AM: An artificial intelligence approach to discovery of mathematics as heuristic search, Computer Science Department, Report No. STAN-CS-76-570, Stanford University, July 1976

Michalski R.S., Learning by inductive inference, NATO Advanced Study Institute on Computer Oriented Learning Processes, Bonas, France, 1974 (also in the book: Computer Oriented Learning Processes, edt. J.C. Simon, Noordhoff-Leyden 1976)

Michalski R.S., VARIABLE-VALUED LOGIC: System VL_1, 1974 symposium on multiple-valued logic, West Virginia University, Morgantown, West Virginia, May 1974

Michalski R.S., On the selection of representative samples from large relational tables for inductive inference, M.D.C. 1.1.9., Department of Information Engineering, University of Illinois at Chicago Circle, Chicago, Illinois 60680

Morgan C.G., Automated Hypothesis generation using extended inductive resolution, Advance Papers of the 4th I.J.Conf. on Artificial Intelligence, vol.1, Tbilisi, Georgia, USSR, September 1975

Shortliffe, MYCIN: a rule based computer program for advisin physicians regarding antimicrobial therapy selection, Ph.D. Thesis, Computer Science Department, Stanford Universisty, October 1974

Simon H.A. and Lea G., Problem solving and rule induction: a unified way, Carnegie-Mellon Complex Information Processig Working Paper 227, 1973

Solomonoff R.J., Some recent work in artificial intelligence, Proceedings of IEEE, vol.54, no.12, December 1966

Stepp R., User's guide for UNICLASS program, an internal report, Department of Computer Science, University of Illinois, Urbana, Illinois, 61801

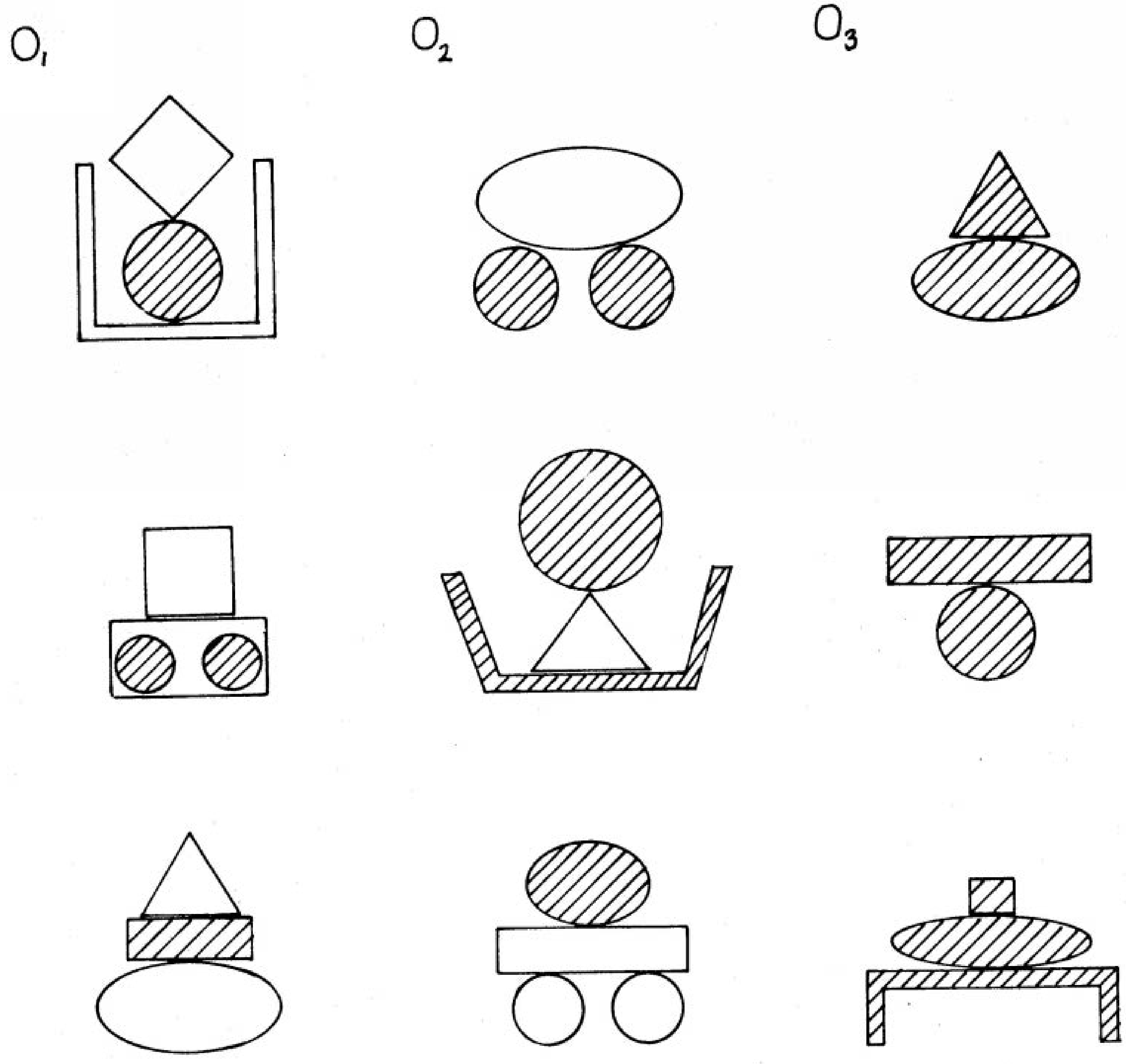
Vere S., Induction of concepts in the predicate calculus, Advance Papers of the 4th I.J. Conf. on Artificial Intelligence vol.1, Tbilisi, Georgia, USSR, September 1975

Waterman D.A., Adaptive production systems, Complex Information Processing Working Paper 285, Carnegie-Mellon University, December 1974

Winston P.M., Learning Structural Descriptions from examples, Tech. Report AI TR-231, MIT AI Lab., Cambridge, Mass.,1970

Yalow E., AQVAL/1 - v12 (briefly, YAL), internal report, Department of Computer Science, University of Illinois, Urbana, 61801

Zagoruiko N.G., Empirical Prediction Algorithm, NATO Advance Insitute on Computer-Oriented Learning Processes Bonas, France 1974 (also in the book: Computer-Oriented Learning Processes , edl. J.C.Simon, Noordhoff-Leyden, 1976)



Find a unique, possibly simple description of each class of objects.

Fig. 1.

BIBLIOGRAPHIC DATA SHEET	1. Report No. UIUCDCS-R-77-874	2.	3. Recipient's Accession No.

4. Title and Subtitle
Toward Computer-Aided Induction: A Brief Review of Currently Implemented AQVAL Programs

5. Report Date
May 19, 1977

6.

7. Author(s)
Ryszard S. Michalski

8. Performing Organization Rept. No.

9. Performing Organization Name and Address
Department of Computer Science
University of Illinois
Urbana, IL 61801

10. Project/Task/Work Unit No.

11. Contract/Grant No.
NSF MCS 74-03514

12. Sponsoring Organization Name and Address
National Science Foundation
Washington, DC

13. Type of Report & Period Covered

14.

15. Supplementary Notes

16. Abstracts

The paper reports on a set of computer programs developed at the University of Illinois whose purpose is to aid in solving certain clasees of inductive tasks. The programs can be used, in particular, for determining most economical (according to a user specified criterion), generalized and discriminant description of given sets of data. The inferred (and initial) descriptions are constructed of various logical and set-theoretic operators (such as "not," "and," "or," set membership, "climbing a generalization tre," quantifiers, equivalence), multiple-valued variables (originally defined or new ones generated by the program), k-place predicates and k-place functions. The underlying formalism for expressing descriptions are variable valued logic systems VL_{21} (a form of a first order predicate logic with additional operators) and VL_1 (a form of a multiple-valued propositional calculus).

Among described programs there is also a program for selecting "most significant" variables from a large number of potentially useful variables (for a given decision problem), a program for selecting "most representative" events from a large set of learning events and a program for testing VL_1 hypotheses and calculating detailed performance statistics.

17. Key Words and Document Analysis. 17a. Descriptors

inductive inference
computer-aided induction
machine learning
generalization techniques
variable-valued logic
knowledge-based systems
production systems

17b. Identifiers/Open-Ended Terms

17c. COSATI Field/Group

18. Availability Statement	19. Security Class (This Report) UNCLASSIFIED	21. No. of Pages
	20. Security Class (This Page) UNCLASSIFIED	22. Price