

LEARNING BY INDUCTIVE INFERENCE

by

Ryszard S. Michalski

Chapter in the book, *Computer Oriented Learning Processes*, J. C. Simon (Editor), NATO
Advanced Study Institutes Series, Series E, No. 14, pp. 321-337, 1976.

OFFPRINT FROM

COMPUTER ORIENTED LEARNING PROCESSES

edited by

J.C. SIMON

Professeur à l'Institut de Programmation
Université Pierre et Marie Curie
Paris VI, France

NATO ADVANCED STUDY INSTITUTES SERIES

Series E: Applied Science - No. 14

NOORDHOFF - LEYDEN - 1976

LEARNING BY INDUCTIVE INFERENCE

R. S. Michalski

University of Illinois
Urbana, Illinois 61801

SUMMARY. The paper is addressed to learning processes which employ inductive inference. A system of variable-valued logic, called VL_2 , is briefly described and its application to implementing inductive learning processes is discussed. The VL_2 can be characterized as a 'multi-valued first order predicate logic'. An example is given of how a computer program learns the simplest relative description of two classes of objects.

INTRODUCTION

Learning processes can be generally viewed as the processes of determining and representing the relationships existing among objects of the universe of discourse (whereby 'objects' are meant physical objects or their parts, abstract concepts, situations: e.g. positions in games, goals: e.g. 'win' in games, etc.). These relationships are determined and represented within the system which learns ('STUDENT') using a source of information about the objects ('TEACHER'). It has been observed (e.g., Bongard¹), that the smaller the degree of STUDENT-oriented organization of information which the TEACHER provides, the greater must be the complexity of the STUDENT. Consequently, the learning processes can be classified according to the degree of STUDENT-oriented organization of information provided by the TEACHER. Thus, we can distinguish, e.g., learning 'by being born' (innate capabilities) or being designed (the greatest organization on the part of the TEACHER), learning by being programmed, learning from examples, from observation ('without teacher'), learning by 'inspiration'. In this paper we are concerned with problems which belong to the area of 'learning from examples'.

Like physical processes which are governed by a law of minimum energy, it seems that information processes, thus also learning processes, are governed by a corresponding law of 'minimum-complexity' (or 'maximum-simplicity'). In other words, information processes seem to have an overall tendency to achieve given information processing goals by the simplest means (which, in special cases, just means the minimum number of operations). An evidence of the existence of such a tendency in the area of human literary expression is the Zipf's law.² It seems that all human information processing activities, in particular scientific activities, are oriented toward determining adequate and, at the same time, simple descriptions or explanations of surrounding environment and phenomena. The ability to create the simplest descriptions, which use only the 'most significant' concepts, and disregard the 'irrelevant details', is highly regarded and considered an evidence of intelligence. But how can we formally define such concepts as the 'simplest description'. How can we create machines which have the ability of determining such descriptions?

As Banerji³ pertinently observed, a simple concept for one person may not be simple for another. His explanation of it is that 'there is something in the human mind which, given constant exposure to a concept, however complicated, makes it simple'. This explanation can be deepened by saying that a seemingly complex concept becomes simple if it is well understood, which, in turn, means that its relationship to the well-known concepts has been clearly established. Therefore, in order to be able to define a measure of simplicity of descriptions, two requirements have to be first satisfied:

- (1) A language in which descriptions are expressed has to be assumed.
- (2) A measure of 'semantic equivalence' of descriptions has to be established. This condition is necessary because for determining the 'simplest description' of whatever we describe, we want to compare only descriptions which convey the same information (i.e., which are semantically equivalent).

Having satisfied (1) and (2), a measure of simplicity of descriptions can be easily formalized. It can be, e.g., a monotonically decreasing function of the length of a description (measured, e.g., by the number of certain assumed constructs of the language which occur in the description). If there is given a 'simplicity function' over the individual constructs, then one can consider a weighted sum of constructs. If only a preference order of constructs is assumed, then one could use the lexicographic functional defined by Michalski.⁴

In this paper we present some recent results from our work on the theory and computer implementations of systems which can

learn the 'simplest descriptions' by executing an inductive inference process ('inductive learning').

LANGUAGE FOR EXPRESSING DESCRIPTIONS: SYSTEM VL_2

The formal system which we are currently developing as a tool for expressing descriptions and implementing inductive learning is a variable-valued logic system VL_2 . This system is an extension of the system VL_1 described by Michalski.^{4,5,6} The VL_2 system gives a sound formal basis for developing an 'algebra of descriptions' which would enable one, for example, to build descriptions, to simplify them, generalize to various degree, to compare descriptions of individual objects or classes of objects, to infer a description of a class of objects from examples of objects of this class, etc.

The full definition of the system VL_2 is not yet available. For the purpose of this paper we will briefly and informally describe some* of the concepts of the system, most relevant to our subject.

To do it simply, we will relate our description of the system to the presently widely used first order predicate logic (FOPL):

1. In FOPL, the atomic formulas (k-ary predicate symbols followed by k occurrences of variables, function forms and/or constants) are assumed to be binary valued (true or false). In the VL_2 , these formulas (called atomic forms) are treated as functions which, as well as their arguments, range over independent domains. These domains are determined as most appropriate for the interpretation of the atomic forms and their arguments, or the problem at hand.
2. The atomic forms occur in a wff of VL_2 (a VL_2 formula) as parts of a broader concept of a selector, and are not, generally, the VL_2 formulas when standing alone (except for the case when a VL_2 formula reduces to a FOPL formula).
3. VL_2 formulas range over an output domain, denoted D, which is a linearly ordered set having the smallest and the largest element.

*In the full definition of VL_2 there are more operations than those described here and the concept of selector has a broader meaning.

4. The selector is defined as a selector statement, SS, enclosed in brackets:

$$[SS] \quad (1)$$

The selector statement is either a conditional statement:

$$L \# R \quad (2)$$

or a quantified statement

$$Q(L \# R) \quad (3)$$

where

L - called the left part of the conditional statement or the referee, is either a VL_2 formula (see p. 5) or an arithmetic sum of atomic forms, or a quantifier-free FOPL formula over atomic forms. It will be assumed for the purpose of this paper that this FOPL formula is in a disjunctive normal form, and that 'or' is denoted by ',', 'and' by '.' and 'not' by a bar over the predicate symbol. For example, a FOPL formula

$$p_1(x, f(y)) \wedge \bar{p}_2(y) \vee p_3(x, y, c) \quad (4)$$

where

$p_1(x, f(y)), p_2(y), p_3(x, y, c)$ -- atomic forms

x, y -- variables, $f(y)$ - a function of y

c -- a constant

is written as

$$p_1(x, f(y)) \cdot \bar{p}_2(y) \cdot p_3(x, y, c) \quad (5)$$

denotes '=' or '≠'

R - called the right part of the conditional statement or reference, is a subset of the union of the domains of atomic forms in L, or a VL_2 formula.

Q - a sequence of existential, $\exists x_i$, and/or universal, $\forall x_i$, quantifier forms, where x_i are variables in atomic forms of L.

Examples of a selector:

$$[p(x, y) = 3] \quad (6)$$

$$[p_1(x, a) \cdot p_2(y, z) = 2, 4] \quad (7)$$

$$[\exists x, \forall y(p_1(x,y,b), \bar{p}_2(y,c) = 0,2)] \quad (8)$$

The selector in which SS is a conditional statement is called a conditional selector (e.g., (6) and (7)), or else it is called a quantified selector (e.g., (8)). A conditional selector $[L \# R]$ in which the referee L is a single atomic form P_1 and the reference R is a subset of its domain, is called a simple selector.

A simple selector $[P_1 = R]$ ($[P_1 \neq R]$) is said to be satisfied, iff the value of the atomic form P_1 is (is not) an element of R. If P, P_1 and P_2 are atomic forms then:

$[\bar{P} = R]$ is satisfied, iff $[P \neq R]$ is satisfied

$[\bar{P} \neq R]$ is satisfied, iff $[P = R]$ is satisfied

$[P_1 \cdot P_2 \# R]$ is satisfied iff $[P_1 \# R]$ and $[P_2 \# R]$ are satisfied

$[P_1, P_2 \# R]$ is satisfied iff $[P_1 \# R]$ or $[P_2 \# R]$ is satisfied

$[(\exists x)(P \# R)]$ is satisfied iff, for given values of all free variables in P (i.e., variables other than x), there exists a value of x which satisfies the selector $[P \# R]$

$[(\forall x)(P \# R)]$ is satisfied iff, for given values of free variables, the selector $[P \# R]$ is satisfied for all values of x.

A VL_2 formula is defined by the following rules:

- (i) an element of the output domain D or a selector standing alone is a VL_2 formula,
- (ii) if V, V_1 and V_2 are VL_2 formulas then so are:
 - (V) formula in parentheses
 - $\neg V$ called the inverse of V
 - $V_1 \wedge V_2$ (written also $V_1 V_2$) called the conjunction or the minimum of V_1 and V_2
 - $V_1 \vee V_2$ called the disjunction or the maximum of V_1 and V_2 .

A VL_2 formula in the form of a disjunction of terms, where term is a conjunction of selectors and an element of D, is called a disjunctive simple VL_2 formula and denoted as DVL_2 .

A VL_2 formula which includes only conditional selectors is called a conditional or quantifier-free formula. In what follows we will discuss only conditional VL_2 formulas.

6. Each VL_2 formula V is assigned a value $v(V) \in D$ depending on the values of atomic forms in it:
- (i) The value of an element of D standing alone is this element itself.
 - (ii) The value of a selector is the largest element of D , if the selector is satisfied, otherwise the smallest element of D .
 - (iii) If the value V is the k -th smallest element of D , then the value of the inverse $\neg(V)$ is the k -th largest element of D .
 $V_1 V_2$ is assigned the smaller of the values of V_1 and V_2
 $V_1 \vee V_2$ is assigned the larger of the values of V_1 and V_2 .

For illustration, below is an example of a VL_2 formula and its interpretation:

$$4[p_1(x_1, x_2) \cdot p_2(x_2, x_3) \neq \text{medium}][p_3 = \text{true}] \vee 3[p_3 = \text{unknown}] \vee 1[p_4(x_2, x_4) = \text{yellow, red}] \quad (9)$$

Suppose that the domains of atomic forms $p_1(x_1, x_2)$, $p_2(x_2, x_3)$, $p_3, p_4(x_2, x_4)$ are, respectively, $D_1 = D_2 = \{\text{small, medium, large}\}$, $D_3 = \{\text{unknown, false, true}\}$ and $D_4 = \{\text{white, yellow, blue, red, black}\}$. And that the output domain of the formula (9) is $D = \{0, 1, 2, 3, 4\}$, ordered as indicated by numbers.

The formula (9) is assigned value (has value) 4, iff atomic forms $p_1(x_1, x_2)$ and $p_2(x_2, x_3)$, for given values of x_1, x_2 and x_3 take value not equal 'medium', and p_3 takes value 'true'. If the above condition is not satisfied, and p_3 takes value 'unknown', then (9) has value 3. If both of the above conditions do not hold and $p_4(x_2, x_4)$, for given values x_2, x_4 , takes value 'yellow' or 'red', then (9) has value 1. If none of the above conditions hold, (9) has value 0.

BASIC CONCEPTS UNDERLYING INDUCTIVE INFERENCE BY MEANS OF VL_2

The subject of inductive inference by means of the VL_2 system is very broad. For the limitation of space, we will only delineate some of its major concepts.

Suppose that the domains of all atomic forms in a VL_2 formula are D_1, D_2, \dots, D_n . The set of all possible sequences of values of atomic forms, that is set $D_1 \times D_2 \times \dots \times D_n$, is called an event space of the formula, and its elements are called events. The event space of a formula V is denoted by $E(V)$. If the output domain of V is set D , then V expresses a function

$$f: E(V) \rightarrow D \quad (10)$$

The atomic forms in a VL_2 formula denote functions of the similar type, namely an atomic form $p_i(x_1, x_2)$ denotes a function

$$p_i: D_{i1} \times D_{i2} \rightarrow D_i \quad (11)$$

where D_i is the domain of $p_i(x_1, x_2)$ and D_{i1} and D_{i2} domains of x_1 and x_2 , respectively.

The atomic forms, however, do not express the functions (11), they only denote their names and arguments. For further considerations we will make a simplifying assumption, that these functions are fixed and can be computed for any given values of their input variables.

Let V_1 and V_2 be two VL_2 formulas having comparable sets of atomic forms* (i.e., one set includes or is equal to another set). And let E be a subset of the event space E , specified by domains of the larger of the two sets of atomic forms. Formulas V_1 and V_2 are called semantically E-equivalent, which we write:

$$V_1 \stackrel{E}{=} V_2 \quad (12)$$

iff for every $e \in E$

$$v(V_1) = v(V_2) \quad (13)$$

If $E = E$, then V_1 and V_2 are called semantically equivalent and we write $V_1 \equiv V_2$. A rule which transforms one formula into another, semantically equivalent formula, is called an equivalence-preserving transformation rule. Below are given examples of such rules (read ' \equiv ' as: 'the formula on the left side may be replaced by the formula on the right side'). Assume that V is an arbitrary VL_2 formula; P P_1, P_2 are atomic forms; $R_1, R_2 \subseteq D_1$ (D_1 is the domain of P), and $R \subseteq D_1 = D_2$ (D_1, D_2 are domains of P_1 and P_2 , respectively).

* The atomic formulas are here considered equal if they represent functions which differ only in that some of their arguments are substituted by a value from the domains of the arguments.

$$V[P_1 = R_1] \vee V[P_1 = R_2] \equiv V[P_1 = R_1 \cup R_2] \quad (14)$$

$$V[P_1 \neq R_1] \vee V[P_1 \neq R_2] \equiv V[P_1 \neq R_1 \cap R_2] \quad (15)$$

If $R_1 \cup R_2 = D_i$ and $R_1 \cap R_2 = \emptyset$ (empty set) then (14) and (15) reduce to (16) and (17):

$$V[P_1 = R_1] \vee V[P_1 = R_2] \equiv V \quad (16)$$

$$V[P_1 \neq R_1] \vee V[P_1 \neq R_2] \equiv V \quad (17)$$

$$V[P_1 = R] \vee V[P_2 = R] \equiv V[P_1, P_2 = R] \quad (18)$$

$$V[P_1 = R][P_2 = R] \equiv V[P_1 \cdot P_2 = R] \quad (19)$$

$$V[P_1 = R][P_2 \neq R] \equiv V[P_1 \cdot \bar{P}_2 = R] \quad (20)$$

Suppose now that the output domain of a DVL_2 formula V is a set D whose smallest element is $*$. Suppose further that all elements of D , except $*$, denote certain 'specified decisions' about events, and element $*$ denotes an 'unspecified decision'. Let E^+ and E^* denote subsets of $E(V)$ for which V takes specified and unspecified decisions, respectively.

Events of E^+ are those which satisfy at least one term in V (i.e., satisfy all selectors in the term), while E^* are the remaining events in E , i.e., $E^* = E(V) \setminus E^+$. We will call the set E^+ a set of recognizable events of V and E^* a set of not-recognizable events of V . Elements of E^* will be called *-events.

Let V_1 be a VL_2 formula and E_1^+ its set of recognizable events.

A rule which transforms the formula V_1 into a new formula V_2 with a set, E_2^+ , of recognizable events, is called a deductive inference rule (DR) if

$$V_1 \stackrel{E_2^+}{\equiv} V_2 \quad \text{and} \quad E_2^+ \subseteq E_1^+ \quad (21)$$

and is called an inductive inference rule (IR) if

$$V_1 \stackrel{E_1^+}{\equiv} V_2 \quad \text{and} \quad E_2^+ \supseteq E_1^+ \quad (22)$$

According to (22), a rule is an IR, iff V_2 makes the same specified decisions as V_1 for events of E_1^+ , but, also, makes specified decisions for some other events than E_1^+ . A question arises of how these 'other' events should be selected and what decisions should be made about them. To answer this question, a

criterion governing an inductive rule is needed. We accept a criterion which can be characterized as a 'criterion of simplicity'. That is, we design a 'simplicity functional' for VL_2 formulas (which can be modified according to application) and employ inductive rules which maximize the assumed functional.

An important inductive rule of this type is the one which assigns to *-events such decisions which permit one to apply to a given formula rules (14)-(20) whenever it could lead to the simplification of the formula according to the accepted measure of simplicity (it can easily be seen that for intuitively acceptable measures of simplicity a simpler formula will also be more general).

An inductive program, called AQVAL/1, which operates on such principles, has been developed at the University of Illinois and already experimentally applied to selected learning and recognition problems from the area of medicine⁵ and plant pathology (the current version of AQVAL/1 implements a subset of VL_2 called VL_1). It should be mentioned that problems of inductive learning by means of variable-valued logic have a strong relationship to the problems of grammatical inference.⁷

DESCRIBING OBJECTS IN TERMS OF VL_2

In the application of VL_2 to describing objects, atomic forms are used to represent certain functions called descriptors. Descriptors are functions which a learning system uses to describe objects.

Let p_i denote a descriptor:

$$p_i: \prod_{j \in J} D_{ij} \rightarrow D_i \quad (23)$$

where

\prod denotes the cartesian product
 $J = \{1, 2, \dots, k\}$
 D_{ij} - input domains of the descriptor
 D_i - the output domain of the descriptor

Special cases of a descriptor:

1. $J = \{1\}$, i.e., p_i is a unary function. If D_{i1} denotes a set of objects, and D_i a set of the values of a specific characteristic of the objects, then p_i is called a feature.
2. $J = \{1, 2, \dots, k\}$, $k=2, 3, \dots$, $D_{i1}=D_{i2}=\dots=D_{ik}$, $D_i=\{\text{true}, \text{false}\}$
 If D_{ij} denotes a set of objects, then p_i can be interpreted as a k -ary relation among these objects. If $p_i(O_{i1}, O_{i2}, \dots,$

O_{i_k})=true, then we say that the relation among $O_{i_1}, O_{i_2}, \dots, O_{i_k}$ holds, otherwise does not hold. If D_i is not a binary-valued set, but has a finite number of values, then we will say that p_i is a multi-valued k-ary relation.

As we can see a descriptor has a very broad meaning.

Example

Suppose $D_{i_1} = D_{i_2}$ denote a set of parts of a certain physical object. To express a fact that, e.g., a relation 'above' holds between certain parts of the object, we can use a function:

$$\text{ABOVE: } D_{i_1} \times D_{i_2} \rightarrow \{\text{true, false}\} \quad (24)$$

If the relation 'above' holds between O_1 and O_2 we write $[\text{ABOVE}(O_1, O_2) = \text{true}]$, or, since the output domain is just binary, simply $\text{ABOVE}(O_1, O_2)$. Suppose, however, that we want to distinguish between 3 possibilities: not above, little above, much above. In this case we assume that

$$D_i = \{\text{not, little, much}\} \quad (25)$$

To express the fact that O_1 is much above O_2 , we use a selector

$$[\text{ABOVE}(O_1, O_2) = \text{much}] \quad (26)$$

If in describing a class of objects we observe that the part O_1 is either much above or not above the part O_2 , we would write:

$$\text{or } \begin{cases} [\text{ABOVE}(O_1, O_2) = \text{not, much}] \\ [\text{ABOVE}(O_1, O_2) \neq \text{little}] \end{cases} \quad (27)$$

In describing individual objects we can distinguish the following classes of descriptors:





1. Global, 0-level, descriptors.
These are features which characterize objects as a whole (e.g., color, size, texture, length, etc.)
2. Local 1-level descriptors which characterize basic (1-level) parts and k-ary, $k=2,3,\dots$ relationships among them.
3. Local P-level, $P=2,3,\dots$, descriptors which characterize P-level parts and relationships among parts of the P-1 level parts.

AN EXAMPLE OF LEARNING THE SIMPLEST DESCRIPTION OF THE
DIFFERENCE BETWEEN TWO CLASSES OF OBJECTS

Suppose we want to develop a machine which, given examples of objects from certain classes, could learn the simplest (according to some defined criteria) description of these classes of objects or the differences between classes. Let us assume that the machine has already built-in certain elementary abilities, such as the ability to recognize a triangle or rectangle, to measure their size and orientation, to determine various relationships between the recognized objects, e.g., a relation 'on top of', 'in between', etc. The problem of implementing the abilities of this type is quite difficult by itself. Though, there have already been developed computer programs which can, to a limited degree, measure the descriptors of the kind described above (see, e.g., Winston⁸). It is important to observe, however, that the number of such elementary descriptors which potentially may be needed is not very large, and therefore each of them could be implemented by a specially designed software or hardware device. On the other hand, the number of potential combinations of these descriptors, which may occur in descriptions of real objects, is prohibitively large. Therefore, an important problem, to which we are addressing ourselves is how to implement very efficient inference and learning processes which create goal oriented descriptions of objects or object classes, assuming that these elementary descriptors are available. A problem of this type is illustrated by the following example.

Fig. 1 presents two classes of 'TABLES'. The objective is to implement a learning process which would produce the simplest (with regard to an assumed simplicity functional) relative description of these two classes of TABLES.

Suppose that the following descriptors and their domains are used to describe the TABLES:

1. global descriptors: length; domain = {short, long}
parts; domain = {3, 4}
2. a) features of individual parts P_i , $i=1,2,3,4$, (table-top, left leg, right leg, and bar, respectively):
 part-type(P_i); { \emptyset , \square , ∇ , ∇ , \equiv }
 part-length(P_i); { \emptyset , short, long}
 part-texture(P_i); { \emptyset , , , , }
 (\emptyset means 'not relevant' - when a part does not exist)
- b) binary relations among parts on-top:
 on-top(P_i, P_j); {above-middle, above-left, above-right}

- c) ternary relations among parts:
 in-between(P_i, P_j, P_k); (low, high) (part P_i is between P_j
 and P_k , and located low or high)

Using these descriptors, the machine describes each object in terms of the VL₂ system, as a conjunction of selectors. For example, object 1 in class 1 would be described as:

$$\begin{aligned} &[\text{length}=\text{short}][\#\text{parts}=4][\text{part-type}(P_1)=\square][\text{part-type}(P_2)=\nabla] \wedge \\ &[\text{part-type}(P_3)=\nabla][\text{part-type}(P_4)=\ominus][\text{part-length}(P_1)=\text{short}] \wedge \\ &\dots[\text{on-top}(P_1, P_2)=\text{above-right}][\text{in-between}(P_4, P_2, P_3)=\text{high}] \quad (28) \end{aligned}$$

Suppose that T_{i1}, T_{i2}, T_{i3} and T_{i4} denote the descriptions of objects 1, 2, 3, 4 in class i , $i=1, 2$, respectively. A description of the class 1 (which is the 'least general') could then be:

$$\text{CLASS1}(T_{11} \vee T_{12} \vee T_{13} \vee T_{14}) \quad (29)$$

and of class 2:

$$\text{CLASS2}(T_{21} \vee T_{22} \vee T_{23} \vee T_{24}) \quad (30)$$

where $\{*, \text{CLASS1}\}$ and $\{*, \text{CLASS2}\}$ are the output domains of formulas (29) and (30), respectively.




(Events which satisfy none of these formulas are *-events.) Suppose now that as a simplicity criterion we accept a criterion demanding that a formula has the minimum number of terms, and, with the secondary priority, the minimum number of selectors.

A way to attain the simplest, in the above sense, relative description of the two classes, is to maximally simplify and generalize the formulas (29) and (30) under the restriction that the resulting formulas will have the empty intersection. (The 'empty intersection' means that there will be no events which satisfy both formulas.) This is done by assigning to *-events such decisions which lead to the maximal simplification and generalization of formulas (29) and (30) by using rules (14)-(20) (without violating the above-mentioned restriction). Such an inductive process can be very efficiently executed by the previously mentioned computer program AQVAL/1. The simplest formulas, according to our criterion, obtained from the program AQVAL/1 were:

$$\text{CLASS1}[\text{length}=\text{short}][\text{part-texture}(P_4)=\text{⦶}, \text{⦿}] \quad (31)$$

$$\text{CLASS2}[\text{length}=\text{long}] \vee [\text{part-texture}(P_4)=\text{⊖}, \text{⦿}] \quad (32)$$

(the execution time was less than 3 sec. on the IBM 360/75; AQVAL/1 is written in PL/1).

These formulas state that TABLES of class 1 are 'short' and the texture of the bar is  or , and that TABLES of class 2 are either long or the texture of the bar is  or there is no bar.

This description of the classes seem to agree with what a human might accept as a 'most simple' relative description of the two classes.

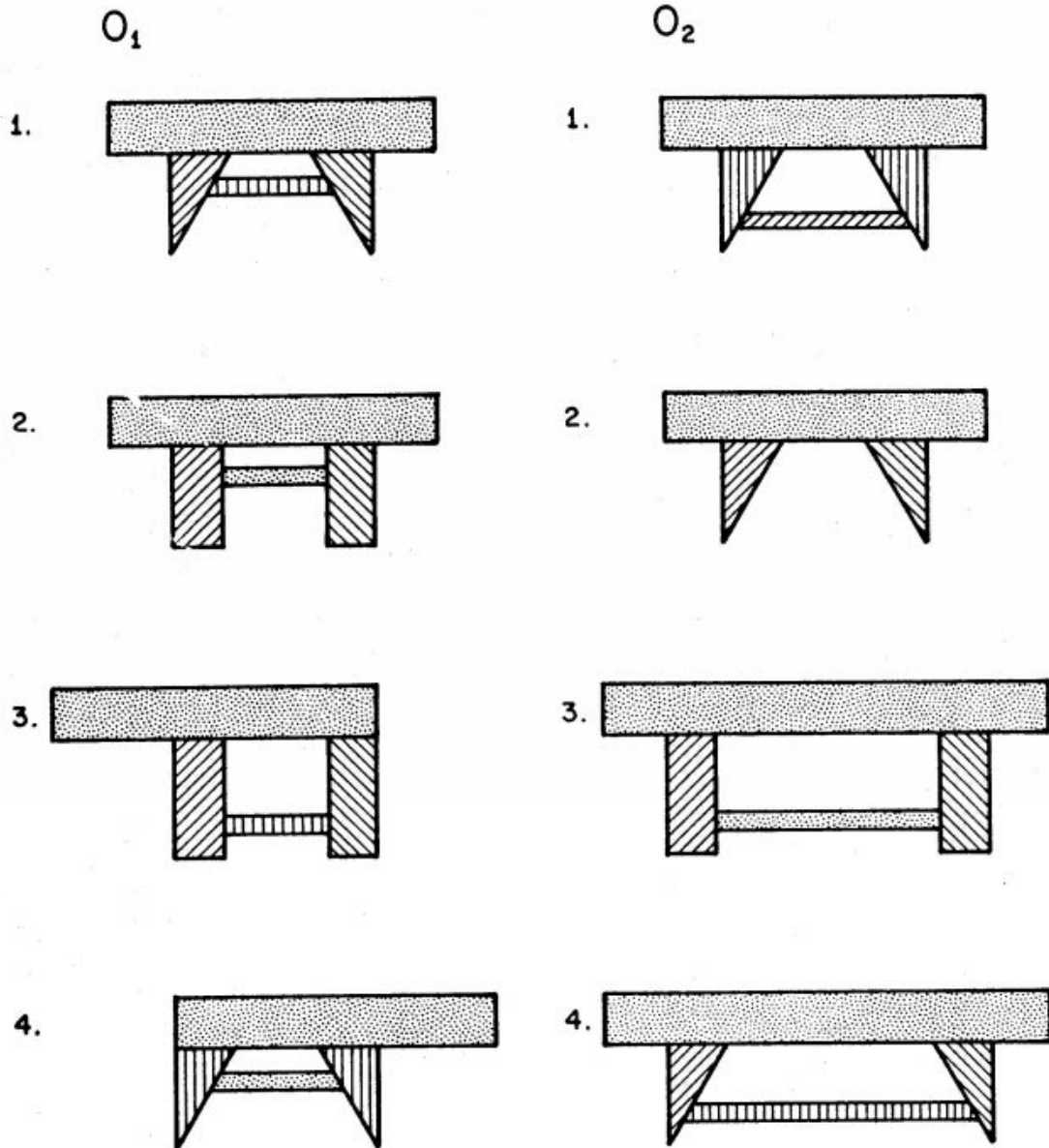
ACKNOWLEDGMENT

The author gratefully acknowledges the financial support he obtained from the Department of Computer Science of the University of Illinois at Urbana-Champaign for conducting the research reported in this paper. It is also his pleasant duty to express thanks to Mr. A.B. Baskin for the fruitful discussions, criticism and proofreading of the paper.

REFERENCES

1. Bongard, M. M., Problema uznawania, izd. Nauka, Moscow 1967. (English trans.: Pattern recognition, New York Spartan Books, 1970).
2. Cherry, C., On Human Communication, the M.I.T. Press, Cambridge, Mass., 1964.
3. Banerji, R. B., Simplicity of concepts, training and the real world, Artificial and Human Thinking, edit. A. Elithorn, D. Jones, Jossey-Bass Inc., Publishers, 1973.
4. Michalski, R. S., A Variable-Valued Logic System as Applied to Picture Description and Recognition, GRAPHIC LANGUAGES, edit. F. Nake, A. Rosenfeld, North-Holland Publishing Company Proceedings of the IFIP Working Conference on Graphic Languages, Vancouver, Canada, May 1972).
5. Michalski, R. S., AQVAL/1--Computer Implementation of a Variable-Valued Logic System and the Application to Pattern Recognition, Proceedings of the First International Joint Conference on Pattern Recognition, Washington, D.C., October 30-November 1, 1973.
6. Michalski, R. S., VARIABLE-VALUED LOGIC: System VL₁, Proceedings of the International Symposium on Multiple-Valued Logic, West Virginia University, Morgantown, West Virginia, May 29-31, 1974.

7. Baskin, A. B., A comparative discussion of variable-valued logic and grammatical inference, Report No. 663, of the Department of Computer Science, University of Illinois, Urbana, July 1974.
8. Winston, P. J., Learning structural descriptions from examples, Ph.D. thesis, MAC-TR-76, Artif. Intell. Lab., M.I.T., 1970.



Two classes of 'TABLES'
Fig. 1.

MICHALSKI : Discussion

SIKLOSSY

Most good languages (natural or programming languages) are highly redundant. And you take an extreme step in your presentation saying that you look for a maximum of non-redundancy.

MICHALSKI

I am looking for goal oriented descriptions which disregard what is not important for this goal, descriptions that are just sufficient, e.g., to recognize an object among other objects without measuring or checking unnecessary or irrelevant information.

SIKLOSSY

I have two more remarks. You did not mention studies by Hunt. Then in the example you give in your paper, Pitrat just proposed another description which I find more natural : O_1 is defined by : table not centered or bar is high.

MICHALSKI

Hunt uses only elementary selectors, i.e., selectors in which left part is a single variable and right part is a single element of the variable domain. Thus, he considers only a simple special case of the system VL_2 (also of the earlier system VL_1). Yes, the description you mentioned is an alternative description. The program could also discover it i.e., would give a description :

$$\text{CLASS1 } ([\text{on-top}(P_1, P_2) \neq \text{above-middle}] \\ \vee [\text{in-between}(P_4, P_2, P_3) = \text{high}])$$

if the criterion of the simplicity, specified as input data to the program, would be such that the above two single-selector conjunctions are preferable over one two-selector conjunction chosen by the program (the latter was preferable according to the criterion used by the program).

COULON

You have a simplicity criterion. ROSENFELD did use also a discrepancy criterion as well. Don't you need it ?

MICHALSKI

Yes I have a condition of computational efficiency or simplicity. Discrepancy is taken care of by the covering algorithm which finds a cover of the given set against another set or sets, e.g., it covers all "positive" examples but doesn't cover any "negative" examples.

HUET

What are the relations with classical algorithms to minimize boolean functions ?

MICHALSKI

There is a relation : the classical algorithms are not useful, they are much too inefficient to be used for problems we are interested in, e.g., when we have, say, 50 or 80 multivalued variables. In the algorithm A^q which we use, not all complexes are generated (i.e., all prime implicants in the binary case) but only very small subsets of them in each step of the algorithm. We also have certain parameters which control the search space, reducing it more or less depending on the difficulty of the problem. The price for reducing the search space is that the chance of obtaining an optimal or very-close-to-optimal solution is decreasing, though we still may obtain an optimal solution.

HUET

The language you propose seems to be a first order language.

MICHALSKI

Yes, it can be characterized as a "multivalued first order predicate calculus" with some additional operators which seem to be useful for our purpose.

ADRIEN

Your principle to select your criteria (the minimum number of terms) is not semantic. So you have no proper basis for generalization, which is however a final goal of the classification.

MICHALSKI

Yes, we look for the "simplest" explanation of the known facts. Thus, the generalization the system is making is governed by the "simplicity or efficiency" criterion. Therefore, obviously, the system may make errors in explaining "new facts". If this happens, the formulas should be properly corrected. Similarly as we correct our hypotheses when they do not explain new facts. This is not difficult to do in our "VL algebra".