Representations and Algorithms for Cognitive Learning

Manfred Kochen

Mental Health Research Institute, University of Michigan, Ann Arbor, Mich. 48104, U.S.A.

Recommended by S. Amarel

ABSTRACT

This is a report summarizing our progress towards a theory of cognitive learning. It is concerned with an algorithm that recognizes, selects and formulates in an internal language problems that arise in an external environment. This algorithm revises its representation of the environment and uses it to cope with self-selected problems.

The algorithm depends on the formation of hypotheses and their use to select actions. The key ideas of this project are major new additions to a theory of representation of knowledge built on an inductive predicate logic.

1. Introduction

Algorithms for recognizing problems and formulating corresponding problem-statements are of basic interest in both artificial intelligence and theoretical behavioral science. How, for example, could a program do anything like what Homer Atkins and his Sarkhanese partner Jeepo in *The Ugly American* [18] did when they picked, formulated, and solved the problem of developing a man-powered water pump to raise water from one terraced rice paddy to another under the constraints they imposed. Proven techniques for educating people so that they can shift representations—in the above example to conceive of rider + bicycle as a pumping motor when it is not needed for transportation—can have significant social consequences.

In this paper we report progress toward the development of a research tool to investigate algorithms for learning to recognize and cope with problems by the formation and use of representations that are built of hypotheses. We call this "cognitive learning". We regard this as a process for utilizing experiential inputs to shift representations so that an increasing variety of problems can be recognized, formulated, coped with.

Artificial Intelligence 5 (1974), 199–216

The mainstream of activity in artificial intelligence has been directed towards making computers more powerful aids to human problem-solving, by enabling a computer user to state his problem in programming languages at ever higher levels and expecting the computing system to find a solution from merely the problem-statement. But the user has to present a well-defined problem-*statement*. Such a statement must specify three sets: (1) all possible "solutions"; (2) properties a solution must have; (3) methods or knowledge necessary to find a solution. For example, "Find a pair of integers (x, y) such that $7x + 5y = 31$ using Euclid's algorithm" is a well-defined statement; the three sets are (1) all integer pairs, (2) the given equation, (3) the name of a method, known to work.

For a computer to solve the problem in the above example if the problem-statement in quotes is presented as input, there must be programs to analyze this input sentence both syntactically and semantically. The result of this analysis is another program. Running that program generates a solution. All this is within the state of the art of advanced computer programming for a rather narrow and well defined class of problem statements. To enable a computer to deal with any of a large and diverse class of problem-statements without considerable readjustment of the stored programs for each problem-type is quite another matter.

Problem-solving in the context of well-defined problem-statements is equivalent to linguistic analysis, path-reduction and efficient search. By changing the specifications of the 3 sets in the problem-statement while still representing the same problem, the search effort to find a solution can be greatly reduced [2, 3]. Specifying a reasonably general algorithm that produces such shifts of representation is as much of a challenge as it was when the "representation problem" [20] was first recognized as central [11]. Learning, in its deeper sense—what we call cognitive learning—involves shifts in representation.

To conceptualize and advance towards truly general-purpose problem-solving algorithms, it is customary to develop techniques which are specific to each problem-class and then attempt to synthesize these special techniques or to generalize from them. This does not appear to be the only or the most attractive way to proceed. Algorithms that enable a computer to learn how to analyze and solve a diversity of problem-statements, developed by a combination of insight and luck, may be an equally fruitful approach. It may even pay to risk the boldest approach of searching for algorithms that enable the computer to formulate its own problem-statements as the need arises.

There has been little concern with problems that do not meet these 3 specifications of well-definedness. Problems that the world creates for people do not come as problem-*statements*, much less as well-defined ones. In that

respect, real problems differ from those presented verbally by an experimenter in a psychological laboratory or by a computer user. We would expect a computer to recognize, select and pose problems by and for itself only if it had to function as an autonomous robot.

The study of such autonomous robots is significant for the theoretical foundations of artificial intelligence and cognitive psychology. The general goal of our research is to use such self-instructed robots as a vehicle for fundamental theoretical investigations towards conceptualizing and explicating cognition and the organization of knowledge.

In previous works [12, 13, 14] we distinguished, with logical precision, between "information" (in the sense of communication theory) and "knowledge". Knowledge differs from information in that it is a property of the knower, interpreted by him through an internal representation system, preparing him for action. Cognitive learning involves the acquisition and use of knowledge for improved coping performance, for taking increasingly effective actions.

In this paper, we present our present conceptualization of cognitive learning. This takes the form of an algorithm which forms and uses representations. Two aspects of this algorithm, in greatly simplified form, were implemented as operational computer programs to illustrate and to serve as a vehicle for research. They are described elsewhere. We first define the kind of problem-generating environment in which cognitive learning is to take place. We then explicate our notion of a cognitive learner. Both models of the environment and the learner are presented in two stages of increasing complexity.

Our primary concern is to show how knowledge can be represented for use in recognizing and coping with problems. The value of the ideas presented here is in their possible stimulation of fellow theorists in artificial intelligence, epistemology, and cognitive psychology to discuss and to further develop this line of modeling.

2. Simple Environments

We start with the concept of a simple environment E. This is a prerequisite to formalizing the notion of a learner L, and, subsequently, complex environments. Then we state a central problem about a learning algorithm and a result about its incompleteness. The formal definition of E that follows is intended to capture the intuitive notion of a law-governed environment over which L has some control, such as a satellite in orbit.

An informal statement describing the notion of a simple environment follows. To specify a simple environment is to specify how the current state of the relevant world of some actor co-determines, together with his action, the next state of his world. It is also necessary to specify whether this change

of state was a step in the direction of making the actor more viable and increasing the "quality of his life", whether it was a step in the opposite direction, or whether it was neither. By viability and quality of life we do not mean personal values, but the analogue of criteria for biological and cultural survival. For example, a change of state in which the pH of the actor's blood increases by a large amount may be in the direction of lower viability, and a change of state in which the actor's income is reduced is likely to decrease rather than increase the quality of his life in the sense of decreasing his options for enjoying the benefits of his culture.

We shall use the words "learner" and "actor" interchangeably. Both are intended to refer to human beings in their role as information-processors in acquiring, organizing, and utilizing knowledge by interaction with their environment. Our primary concern is to better understand how nature could have evolved such information processors with a view towards helping them improve. If we attempt to identify learners and actors as robots or as computer programs, it is only to gain conceptual clarity and deductive power, as well as to explore the conceptual limits of automata as models for human cognitive learning.

The formal definitions that follow are steps in this more precise explication. They are the basis for operational computer programs that generate a general class of environments.

DEFINITION 1. A *simple environment* E is a six-tuple, $\{S, R, A, F, s(0), O\}$, where:

S is a set (of states), e.g. $\{1, 2, 3, 4\}$;

R is a set of relations on S, e.g. $\{\{(1, 2), (2, 3), (3, 4), (1, 3), (1, 4), (2, 4)\}\}$;[1]

A is a finite set (of inputs to E), e.g. $\{0, 1\}$;

F is a mapping of $S \times A$ into S, e.g.

		1	2	3	4	current state
action	0	3	4	3	4	next state
	1	2	1	1	1	

$s(0)$ is a particular element of S (initial state), e.g. 1;

O is a partial ordering (or value-function v) on S or $S \times A$ (preference, utility, survival value), e.g. $4 > 3 > 2 > 1$, or $v(s) = s$, $s = 1, 2, 3, 4$.

For a physical example, consider the motion of a billiard ball. The set of states is the set of all possible positions and speeds of a ball on the surface of the billiards table. Any state can be represented as a four-component vector, (x, y, v_x, v_y), where x and y vary from 0 to the length of the table and the v's vary up to plus or minus the largest speed a billiard ball can attain. If there were two balls, the state can be represented by an 8-component vector, 4 for

[1] (S, R) is a relational structure (Bell and Slomson [5]).

each ball. Relations between the two balls in any state (i.e. at one time) are represented by such predicate-names as "is to the left of". Such a relation is, however, an infinite set of ordered pairs of corresponding x-coordinates.

The set of possible actions in this example is the set of possible forces that could be exerted by the billiards player through his cue in hitting a ball. The transition function F represents the laws of motion of the ball. Such a law has the form: "If at time t a ball is in a certain position and moving at a certain speed, and the player applies a specified force, then, a small time increment later, the ball will be at a specified new position and speed." More complex versions of such a statement take into account collisions with the walls and other balls, and slowing down due to friction. The set of possible actions is part of the specification of a simple environment, because it is part of the domain of the transition function F.

What makes such a lawful world an *environment* is the reference to an actor who can select actions and to whom the various states and actions make a difference. The partial order on the states of an actor's environment represents the varying degrees of viability and quality of the various states. If one state is more viable than another, then the actor "ought" to prefer it, if we assume that he is driven by a motivation to survive. Under well-known, plausible conditions regarding the actor's choices, this partial order can be replaced by a real-valued utility function. We call this the actor's value function; its value represents the utility or value of a state to that actor. No two actors need have the same value function, though certain states will have greater survival value than specified other states for all actors who share the same environment.

The value function may depend not only on the state, but on the learner's action as well, or on just his action. Feedback from the environment consists of information about the change of state, and the ordering of the new and the old state; the utility of the change of state may be interpreted as reinforcement. If the value function depends on the learner's action, we can interpret this as a way to reinforce the *way* that a problem is solved and to attach value to elegance.

The actor is part of a larger world that includes both him and his environment. Well-known paradoxes concerning the consistency of choices occur when more than one actor participates in this larger world. Such larger worlds are no longer the *simple* environments we are starting with here.

The two essential features of a simple environment are the transition function F and the partial order O. The sets S and A are necessary to specify the domain of F and O, and the set of relations R is useful in describing the structure of a simple environment which is to be left unchanged under shifts of representation.

Had we allowed t to be a real variable, then the transition function could

be replaced by a linear differential operator, such as $\sum_{k=0}^{n} a_k \mathrm{d}^k/\mathrm{d}t^k$, to be applied to the state function $s(t)$ and set equal to 0, or more generally, by $F(s(t), \mathrm{d}s/\mathrm{d}t, \mathrm{d}^2s/\mathrm{d}t^2, \ldots, t) = 0$. In that sense E is any environment whose "laws of motion" can be described by differential equations. $s(0)$ corresponds to initial conditions. Because we have not yet explored E–L coupling for even simple environments, we introduce more complex environments in very small steps. This is necessary if we are to conceptualize non-trivial cognitive learning.

In the more realistic, non-physical example of having to raise water from one terraced rice paddy to another, the states can be interpreted in one way as the presence or absence of water at different levels, and the relations are denoted by such terms as "above". The actions correspond to pumping so as to effect changes of state, i.e. a flow of water from one level to the next. The ordering or value function is such as to favor conditions in which each rice paddy has an adequate amount of water to produce a good yield. The set of actions must, however, also specify how the pumping is to be accomplished. It must allow for the representation of such alternatives as electric motors, hand-powered motors, bicycle-powered motors, etc.

Selecting the appropriate state-variables is a most critical determinant of success in recognizing and coping with problems. In this example, it is essential to broaden the notion of state to include an indication of whether a given bicycle is used for transportation or to drive a pump.

The transition function or the partial order (value-function) or both must also specify the constraints on feasible solutions. In this example this excluded electric motors or important devices manufactured outside Sarkhan. The ideal pump was something that was already in widespread use, and which could be readily maintained.

DEFINITION 2. Suppose there exists a value-function $v(s)$, $s \in S$, v a real number, such that L is totally ordered. Let $\boldsymbol{a}(0, T) = (a(0), a(1), \ldots, a(T))$ and let $V(\boldsymbol{a}(0, T), s(0)) = \sum_{t=0}^{T} v(s(t))$, where $s(t+1) = F(s(t), a(t))$. An *optimal path* of length $T' - T$ is a sequence $\boldsymbol{a}(T+1, T')$ for which

$$V(\boldsymbol{a}(T+1, T'), s(0)) = \sum_{t=0}^{T'} v(s(t)) - \sum_{t=0}^{T} v(s(t)) = \max_{\boldsymbol{a}(T+1, T'), s(0)} V(\boldsymbol{a}(T+1, T'), s(0)).$$

A path is *limit*-optimal if there is a T_0 such that for all $T > T_0$, and, for all T', $\theta(T+1, \ldots, T')$ is optimal.

An optimal path is a finite sequence of successive actions that leads to an accumulation of utilities to the actor, which is the greatest he could possibly attain. A path is optimal from only the actor's point of view. If there are several actors, there may be several optimal paths. Alternatively, there may

be one sequence of successive actions by each of the actors cooperatively which brings a higher accumulation of utilities to each of them than would paths that are individually optimal.

Optimal paths are properties of the simple environment. They represent upper limits to which any actor could attain if he acts independently of other actors. For an action sequence of finite length, such an upper limit always exists. It is illustrated in Fig. 1 of Example 1.

If we let the sequence of actions continue indefinitely, we can define upper limits to the accumulation of utilities, but these do not always exist. Under reasonable conditions on both the transition function and the value function, they usually exist. Examples of where limits do not exist correspond to unusual simple environments.

EXAMPLE 1. In the illustrations placed next to the above definition of E, the optimal path is 1, 0 0 0 0 . . . with the resulting state-sequence 1, 2, 4, 4, 4 . . . and $V(t) = 1 + 2 + 4(t - 1) = 4t - 1$. The path 0 0 0 0 . . . is plausible. If the learner tried to pick the action at $t = 1$ which maximized the value of the next state, this would be chosen. The resulting state sequence is 1, 3, 3, 3, . . . with $V(t) = 1 + 3 + 3(t - 1) = 3t - 1$. This path can switch to the optimal path at any time by continuing with action 1 (3 → 1), then 1 (1 → 2), then 0, 0, 0, . . . (2 → 4)(4 → 4) . . . . That is a limit-optimal path.

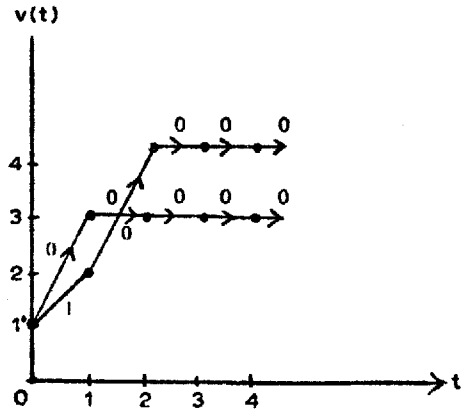

FIG. 1

EXAMPLE 2. $S = \{(x, y, z),\ x, y, z \in \{1, 2, 3, \ldots\}\}$; $R = \phi$; $A = S$; $F = F(x, y, z, a) = (x', y', z') = (x + 2, (\sqrt{y} + 1)^2, z + 2)$; $s(0) = (2, 1, 1)$. The partial ordering O is specified by the following value function:

$$v(s, a) = \begin{cases} 1 & \text{if } a = (x', y', z'), \\ 0 & \text{otherwise.} \end{cases}$$

In this example, F does not depend on a, but only on s. The utility to the actor depends on his current action, which is a prediction of the next state.

Thus, if the current state is (4, 4, 3), the next state will be (6, 9, 5). If the actor selects as his action the triple (6, 9, 5), the value of his value function is 1. For any other action, it is 0.

In this example the actor is assumed to behave in such a way that if he predicts correctly, then this increases his chances of survival or of improving his lot. The utility function which governs such behavior is as much part of the simple environment as is the transition function, even though it might seem as if the utility function specifies the actor. We regard the utility function as similar to the transition function for the "internal", as distinct from the external, environment of the actor.

This simple environment generates a sequence of states such as: 2, 1, 1; 4, 4, 3; 6, 9, 5; 8, 16, 7; 10, 25, 9; . . . . Every third number beginning with the first, second, third is, respectively, even, a square, odd.

Many of the tasks used to test AI programs are readily formulated as special cases of E. This is also the case for many of the tasks used in psychological experiments on problem-solving, such as recognizing the regularity in a sequence.

One advantage of presenting such problems in this formalism is the opportunity to train and test the general learning algorithms on them. Another advantage is in describing examples of applying the algorithm. The main algorithm is specified in the same formalism. The chief advantage, however, is the possibility of demonstrating generality, by showing that the main algorithm can be applied to diverse problems without major readjustments to fit each special case.

Thinking about various specific simple environments has led to a number of interesting new tasks for both artificial intelligence and experiments on human problem-formulation and recognition. More important, this conceptualization of a simple environment captures an interesting class of real problem-generating environments.[2]

The precise description of E is presented as a program in a formal metalanguage L_0 used by the observer-analyst-programmer. In practice this was SNOBOL4. Special assignment-statements can both describe the state-transition and value functions for E and, when executed, generate the successive

[2] The notion of state space has for long been fruitful in theoretical physics, electrical engineering and more recently mathematical economics and biology. Success in solving problems, such as the n-body problem for planetary motion by applying global analysis, depends critically, however, on the choice of the coordinates of state-space (e.g. total energy and angular momentum) (Smale [21, 22]). It is equally easy to replace state-space notions by "Petri nets". There, the nodes of a graph which in state-space denote states, denote moments of change, as a car moving from one section of the highway to another. An edge, which conventionally denotes state-transition, denotes a state, such as a section of highway being occupied or free. In artificial intelligence, it has also been used by several authors (e.g. Doran [8]).

states of E. Several programs to generate specific E's exist; there is one rather general program to generate an E with specified properties.

We view a learner L as a program coupled with E. Its outputs, interpreted as actions, correspond to A. They are inputs to E. For simplicity we assume that every L and E are all synchronized with a clock which measures time as $t = 0, 1, 2, \ldots$ and with states and actions specified for each t. The inputs to L are two successive states of E, plus an indication of whether or not one is not preferred by L to the other.

DEFINITION 3. A *simple actor* L is a seven-tuple $\{S, O, L, A, M, \phi, s^*(0)\}$, where:

S is a finite set (of subjective states), with $s^*(0) \in S$ (initial state), e.g. hunger, thirst, . . .;

O is a partial order on S (personal preferences), e.g. euphoria > hunger, etc.;

L is an internal formal language, e.g. group theory[3] may be part of it;

A is a finite set (of possible actions, as seen by L), e.g. may be in 1–1 correspondence with A;

M is a memory, partitioned into 5 finite sets of cells to be read and written on, reserved for storing inputs,[4] actions and hypotheses;

ϕ is an algorithm which, for each t, (1) selects $a \in A$, (2) forms hypotheses and stores them in M, (3) stores inputs and actions in M, (4) selects, tests, revises hypotheses in M, (5) determines the next internal state.

Subjective states resemble states of consciousness more than they resemble internal physiological or biochemical states. An actor is motivated by preferences among his subjective states. He prefers a state of satiation to one of hunger. The internal formal language is what enables him to represent his environment to himself. Sentences of that language may refer to external as well as to internal and subjective states. The set of actions as seen by the actor need not coincide with the set of actions that are the inputs to his environment. Our previous specification of a simple environment must be expressed in some representation, and if that is not the actor's representation, then A and A^* could differ.

L has a dual nature. It is an object which is part of L's environment. It is also a program, which steers the object.

Only ϕ is the program part of L. The other 6 items are sets of symbols

[3] Formally, group theory is specified by: (a) a vocabulary consisting of: $0, x, x_2, x_3, =, +, (,), \sim, \wedge, \vee, E, \supset$; (b) 6 formation rules generating an infinite set of well-formed or syntactically correct sentences; (c) 5 logical axioms, of any predicate calculus; (d) rules of inference for establishing some sentences as logical consequences of others; (e) 4 axioms for equality; and (f) 3 proper axioms for groups (associativity, identity, inverse).

[4] Each input consists of 3 things. First, there is a received signal corresponding to the current state of E (or to a state-transition). Second, there is a received signal indicating whether states are in some relation determined by an n-tuple of R. Third, there is a received signal indicating whether or not $(s, s') \in O$ (or a value $v(s)$ assigned to state s).

initially stored for ϕ to use. M is a "scratch-pad" store. For simplicity of exposition, assume henceforth that A = A and M is initially empty or cleared. Assume also that L is rich enough to include sentences capable of describing E. We can interpret S, O, L, A as the content of a read-only memory. We view M as a read–write random-access memory with push-down features.

We borrowed the logician's phrase "formal theory" for L. But the specification of L contains no sentences other than the axioms. L itself is the infinite set of possible well-formed sentences. This does not capture the notion of a developing theory as intended in science and as needed here.

At any time t, memory M contains 5 data bases. They store past state-descriptions, instances of relations, utilities, the past action, and hypotheses. We shall denote them by $B(t)$, $R(t)$, $U(t)$, $a(t)$, and $H(t)$, respectively.

The first, $B(t)$, is a store of past state-descriptions which were registered by L as inputs from E, and which are still saved by time t. Not all recorded state-descriptions are saved for all time. When an hypothesis refers to a state, and that hypothesis has been strongly confirmed, then the record of the confirming instances may be erased at some risk. Thus, if in the billiard ball example, a ball was in position 5, 3 at time 4, then this may be stored in $B(t)$, for $t > 4$, as long as a record of that state may be useful.

To explain $R(t)$, consider the example of two billiard balls, with only the position coordinates constituting a 4-component state-description. Suppose the two balls are at 1, 1 and at 1, 2 at time 4. The same two balls move to positions 1, 1 and 2, 2 at time 5. That is, the first ball stays put, and the second moves to the right one step. Suppose that the simple environment consists of just one relation, $<$, which is such that $(1, 1, 1, 2) < (1, 1, 2, 2)$. The relation $<$ is an infinite set of ordered pairs of such quadruples, and $((1, 1, 1, 2), (1, 1, 2, 2))$ is an *instance*, which is recorded in $R(t)$ for $t > 5$.

In this instance, we may name another relation, "is to the left of", and apply it to the first and the second billiard balls in that order, which is defined by $(x, y, u, v) < (x, y, u', v')$. Thus, the instance "Ball 1 is to the left of Ball 2 at time 5" is what may be stored in $R(t)$. The third data base, $U(t)$, is the past record of utilities. The fourth one is the last action that was taken. The fifth is the most important; it consists of saved hypotheses.

A typical hypothesis is illustrated by the following sentence: "For all t, the next state $(1, 1, s_3(t + 1), 2)$ is equal to $(1, 1, s_3(t) + 1, 2)$." The algorithm ϕ attempts to verify selected hypotheses at certain times. It does this by first interpreting the hypothesis, that is, by assigning specific state-references in $B(t)$, relations in $R(t)$, $a(t)$, or utilities to the variables in the hypothesis. For example, data base $B(5)$ might contain an indication that $s(2) = (1, 1, 1, 2)$ and $s(3) = (1, 1, 2, 2)$. This record of two successive states would confirm the above hypothesis. This notion of ϕ interpreting an hypothesis corresponds to the concept of "interpretation" in the predicate calculus. An hypothesis,

interpreted by ϕ, and verified by the "data" in M is a representation of a unit of knowledge.

Based on a count of the number of such confirmations and refutations, each sentence in $H(t)$ is assigned a weight. This varies with t. This weight function has the properties of a measure of plausibility or credibility [10, 19]. It also takes into account the utility of sentences for selecting actions that can drive E into preferred states.

In discussing the data bases in M, we treated the internal representations of states, relations, and actions as if they corresponded exactly to representations of E by the analyst. This need not be the case. It is particularly important to maintain a distinction between the subjective states of which the actor may be aware and of his personal preferences among these states on the one hand, and, on the other hand, of states of the environment and the environmentally determined utility ordering among them. In a normal, healthy actor, the states of the environment are aggregated into large classes, e.g. all states in which nutrition is below a certain level, and these correspond to subjective states, e.g. hunger, such that the preference ordering of the subjective states coincides with the utility ordering of the environmental states. The distinction between the subjective and environmental states, and between information stored in an internal representation and some other representation becomes important when we want to apply the main algorithm ϕ to simulate and study certain abnormal conditions.

For simplicity, we now assume that there is a many-one (homomorphic) mapping of S into S such that the ordering O on S corresponds to the ordering O on S. Thus, ϕ tries to shift L into internal states of S preferred according to O. That is, ϕ searches for hypotheses of high weight. These are likely to imply the selection of actions that drive E into states of S preferred according to L's preference ordering.

DEFINITION 4. An *hypothesis* is a statement in $H(t)$ together with an associated weight $w(t)$.

We consider an hypothesis to be revised if its weight is changed by ϕ. An hypothesis is *true* if it can never be refuted. (It may or may not be a theorem; we should call (L, $H(t)$) a good theory if, along with many true hypotheses in $H(t)$ there are proofs within L.)

DEFINITION 5. A simple actor L is a *simple learner* if (a) its actions are a limit-optimal path, (b) $H(t)$ contains true hypotheses for all t after the optimal path is attained, and (c) the weights for true hypotheses are greater than the weights assigned to other hypotheses.

Psychologists have defined learning very broadly as any behavior in which the probability of a "response" increases. This includes habituation. It could not exclude such effects as fatigue or increased entropy in non-living systems.

In one implementation of the algorithm ϕ, an action is chosen with a probability proportional to the weight of the hypothesis in $H(t)$ which is relevant to the current state. Thus, the notion of a simple learner fits into the psychological conception.

3. Complex Problem-Generating Environments

We now resume our formulation of an environment as one which includes simple learners. This is necessary if we are to get beyond simple environments to deal with environments that require of L the ability to form and use representations of knowledge. Our aim is to formalize the notion of problem-recognition and coping in such environments.

Let $\geqslant$ denote the partial ordering O, interpreted for $s \geqslant s'$ as: state s' is not preferred to state s. By a path from s to s', we mean a finite sequence of actions, $a(1), a(2), \ldots, a(n)$ such that, for some n, $F(s(i), a(i)) = s(t + 1)$, $s(0) = s$, $s(n) = s'$, $i = 1, \ldots, n - 1$.

DEFINITION 6. A state s is a *problem* (or task) for L if (1) L can select at least two a's, $a, a' \in A$, such that $F(s, a) \neq F(s, a')$, (2) there exist states s' and s'' and a path from s to s' and one from s' to s'', and (3) either (a) $s' > s$ and $s'' > s'$, or (b) $s > s'$ and $s' > s''$. Here, $s' > s$, if $s' \geqslant s$ and $s' \neq s$.

This is to capture the idea that L has a problem if there is some future state of E into which L could drive E which L would either strive to reach or to avoid because L "ought" greatly to prefer it to the current state or vice versa. That is, a problem for L is either a hidden opportunity or a hidden trap, an approach or an avoidance task.

State 1 in Example 1 is a problem state, because:

(1) L can select action and action 1. The transition function $F(s, a)$ is such that $F(1, 0) = 3$ and $F(1, 1) = 2$.

(2) There are two other states, namely 2 and 4, and there is a path from state 1 to state 4 via state 2, namely action 1 followed by action 0.

(3) The value of state 4, which is 4, exceeds the value of state 2, which is 2, and this, in turn, is greater than the value of state 1, which is 1.

The purpose of (3) is to distinguish problem-states from ordinary states from which there could be a one-step transition to a state of higher value. A problem-state, interpreted as an opportunity, is to offer the possibility of change to a *considerably* more valuable state. We tried to capture this by requiring that there be at least two increases in value for a two step path. (2) is to assure that such a considerably more valuable state is attainable. (1) is to make sure the actor is faced with a non-trivial choice.

So far, we have considered E's in which the state transition function either does not depend on L—e.g. $F(s) = s$—or where L can control E completely.

If a program such as L exists, we can certainly conceive of N "copies" of

L. Call them $L_1, \ldots, L_N$. To specify L_i is to specify $\{S_i, O_i, L_i, A_i, M_i, \phi, s_i^*(0)\}$ as in Definition 3. Only ϕ is, in general, the same for all i. If they are truly identical copies, then the other 6 objects in the 7-tuples are the same also. By copies we mean only that ϕ is the same, and at least $s^*(0)$ is different. We still make the above assumptions about S_i, O_i and $A_i = A$; also that L_i and M_i are the same for all i. By a simple environment E we now mean a simple environment in which the set of possible inputs is A^N; each input is an N-tuple corresponding to N simultaneous outputs of $L_1, \ldots, L_N$. No longer can one learner control E if F depends on $a = (a_1, \ldots, a_N)$. Also the input to L_1 is now $a_2, a_3, \ldots, a_N$ as well as the new state and its utility to L_1. To make it more interesting, however, we can specify which components of the state-vector L_i receives as input. For example, if $N = 2$ and $s = (s_1, s_2, s_3)$, L_1 might receive a_2 and s_1, s_2 while L_2 receives a_1, s_2 and s_3. That part of L_i's memory M which stores inputs stores not only its own past actions but those of the other L's as well.

Definition 6 of a "problem for L_i" still applies here, with a, a' replaced by a_i, a_i', except that a path from s to s' is now a sequence of acts by all N learners, $(a_1(t), \ldots, a_N(t))$, such that for some t_0 and t_1, $F(s(t), a_1(t), \ldots, a_N(t)) = s(t+1)$, $s(t_0) = s$, $s(t_1) = s'$, $t = t_0, t_0 + 1, \ldots, t_1 - 1$. There are problems for L_i which L_i can recognize and solve only by the simultaneous, coordinated actions of $L_1, \ldots, L_N$. A *problem recognized by* L_i is an internal state of L_i which satisfies the conditions of Definition 6 (with $\geqslant$ applying to the ordering O rather than O, which we had assumed isomorphic, for simplicity). To "solve" such problems, the N learners must either be controlled by a higher-level coordinator capable of forming the solution or by communicating with one another in an external (public, conventionalized) language L_e.

DEFINITION 7. The *complex environment of* L_1 in a universe consisting of $L_1, \ldots, L_N, E$ is $(L_2, L_3, \ldots, L_N, E)$.

In the complex environment of L_1, L_2 has a dual existence. On the one hand, it is a program, formally a 7-tuple as in Definition 3. On the other hand, it may have attributes which appear as components of the state-vector of E. Like a robot, L_2 appears to L_1 as an object specified by position coordinates, size, etc. In addition, however, L_2 produces an action a_2 and sentences of L_e which are input to L_1.

DEFINITION 8. L_i *copes* with a problem in a complex environment if (1) L_i recognizes it, (2) selects a future state s'' to be reached or avoided, (3) selects actions which are part of some path from the present state to s'', (4) generates messages to and utilizes messages from L_j, $j \neq i$, whose coordinated actions may be necessary, and (5) succeeds in reaching or avoiding s''.

4. Representations and Higher Learners

We now revise Definition 3 as follows. First, A, the set of possible actions or outputs of L is infinite and consists of (a) the set L_e of possible sentences (requests, questions, answers) that can be generated by rewrite rules that are part of the specification of A, and (b) actions that are generated by ϕ as compounds of a set of primitive actions. Secondly, ϕ can modify L by adding new symbols to its vocabulary, new rules of formation, new axioms and postulates and new special rules of inference. Thirdly, ϕ can modify S and O. If, for example, ϕ causes the subjective state of S "pain" to be replaced by several states like "headache", toothache", etc, and if according to O, "pleasure" is preferred to "pain", then ϕ causes O to be revised so that "pleasure" is preferred to "headache" and to "toothache".

We now specialize all the above sets. S contains only three elements, corresponding to a high reward state, a tranquil state and a disturbed state.[5] O is obvious. L's vocabulary contains a set PFL of primitive predicate—and function-names such as EQ(X, Y), PLUS(X, Y) corresponding to = and + in the previous example for group theory, (see footnote 3, p. 207) and EVEN(X), interpreted as "x is even", etc. The formation rules of L include, for example, the following:

(a) VHYP → If PREDS, then B;
(b) PREDS → EVEN(X);
(c) B → VAL = FUNCT;
(d) FUNCT → PLUS(X, Y);
(e) X, Y → | 1 | 2 | 3 | $s\langle 1\rangle$ | $s\langle 2\rangle$ | t | FUNCT | ACT |.

An example of an hypothesis formed from just these rules is "If EVEN ($s\langle 1\rangle$), then VAL = PLUS(ACT, PLUS($S\langle 1\rangle$, 2))", interpreted as follows. If the first component s_1 of the state-vector of E is even, then the utility or value of that state is a number equal to $s_1 + 2 +$ the numerical input representing L's act. This is an example of a "value-hypothesis" (without the associated weight).

Similar rules allow for the formation of an action-hypothesis, which is a formal sentence interpreted as "If the present state is such that $s_1 = s_2^2$ and s_3 is even and the action selected is 1, then the next state is such that $s_1' = s_1 + 1$, $s_2' = s_2!$, $s_3' = s_1^2$". A companion hypothesis states[6] that, otherwise, $(s_1', s_2', s_3') = (s_1, s_2, s_3)$. This hypothesis without its companion is only partially relevant because it does not account for all possible conditions; together with its companion, it describes a function f that assigns to each

[5] We often take S to be a vector space. In this example of a 4-dimensional space, the x–y coordinates of two objects, A and B, are given as (x_A, y_A; x_B, y_B). Only states in which A and B have the same x-coordinate—are vertically aligned—are reward states.

[6] Here the next-state components are denoted by primes.

current state and action a unique next state. The hypothesis is true if f is equivalent to F.

DEFINITION 9. The 5-tuple (L(t), $H(t)$, $B(t)$, $R(t)$, $U(t)$) is L's *representation* of E at time t. It is an *accurate* representation if $H(t)$ contains a true hypothesis of higher weight than the other hypotheses.

Recall that $B(t)$, $R(t)$, $U(t)$ are the data bases against which action hypotheses and value-hypotheses are tested. If an hypothesis is sufficiently confirmed and not falsified, the confirming data could be erased, at a risk. Moreover, more general hypotheses can replace several others that are special cases. In this way, optimum use can be made of the limited memory storing $H(t)$, $B(t)$, $R(t)$, $U(t)$.

Shifts of representation can occur at several levels. At the lowest level, ϕ may effect a significant change in the distribution of weights over an unchanged set of hypotheses in $H(t)$. At a higher level, new hypotheses, possibly contradicting existing ones are added. More radical is a new set of entirely different hypotheses replacing $H(t)$, but all still within the same L(t). At the highest level is a shift in L(t), for example the addition of new functions and predicates to PFL.

DEFINITION 10. L is a *higher learner* if the number of problems in its complex environment with which it copes varies with time at a rate no smaller than that at which such problems arise.

Can a higher learner cope with any complex environment? Gold [23] showed that under certain conditions such a learner will commit the only possible error. He also conjectured that there exist trap states which prevent the maximum reward rate from being attained.

If the repertoire, PFL, of primitive predicates and functions is too small or weak, then there exist environmental transition functions F that no hypothesis in L can represent. Nor could combinations of these primitives or even adding a finite number of new predicates ensure that there is no F that can then be represented by some hypothesis f.

But a great variety [7] of interesting environments can be represented. To illustrate, reconsider formation rules (d) and (e) introduced at the beginning of this section, and replace (d) by: FUNCT → | ID(X) | SQ(X) | FC(X) | SC(X) | NG(X) | PLUS(X, Y) | MULT(X, Y) | EXP(X, Y), interpreted as x, x^2, $x!$, $x + 1$, $-x$, $x + y$, xy, and x^y. Repeated application of

[7] A most remarkable and fundamental result of Kolmogorov and Arnold states that if $f(x_1, \ldots, x_n)$ is continuous in $x_1, \ldots, x_n$, it can be expressed as a repeated composition of functions of just one or two variables. For 2-valued functions of n 2-valued functions a similar result follows from Boolean algebra. This makes it reasonable to assume that a large class of interesting functions can be obtained by repeated composition of functions in a set of rather simple primitives.

both rules, especially the rule $X \rightarrow$ FUNCT leads to an infinite set of functions obtainable by repeated composition, e.g. SQ(SC(SQ(SC(SQ (. . . (*X*) . . .))). Some functions, for example, $(x + 1)^2$ or SQ(SUC), are equivalent to others, e.g. PLUS(SQ(*X*), PLUS(MULT(2, *X*), SUC(*X*))). For certain sets PFL of primitive functions, the set of all functions obtained by composing them can be a group. Composing a pair of composed functions, such as SUC(*X*) and PLUS(*X*, NG(1)), can be made equal to ID(*X*). Such a statement of equality can be viewed as a transformation rule. This is part of L. Determining whether or not two functions in such a group are equivalent is the word problem for groups. This is known to be undecidable in general (except for special groups).

Suppose ϕ can add new predicates and functions to PFL. Then the set of possible environments that can be represented is greatly enriched. Insofar as the universal learning algorithm is a finite string of words describing how to select the next action, how to form a new hypothesis, how to test, select, and revise hypotheses, and how to select the next internal state—as well as how to form new predicates—, it can be represented by another algorithm capable of generating such descriptions. The complex environment is characterized in part by this universal learning algorithm because it includes other learners. The question then arises whether a universal learning algorithm could eventually learn to represent itself. A partial answer is given by Myhill's interpretation of the results of Church and Gödel as "our creativity outruns our capacity for anticipating the outcome of that creativity" [24]. This seems to indicate that there are logical limits to the existence of a universal learning algorithm that can learn to represent any complex environment.

More advanced versions of a higher learner incorporate the ability to form hypotheses about his own abilities, particularly his ability to form hypotheses. This is our approach to an explication of consciousness. An actor is aware of being hungry if he not only has an hypothesis to that effect, but if he has a second-order hypothesis about his ability to form, process, abandon, revise the hypothesis that he is hungry.

5. Conclusions

The conceptualization of cognitive learning we proposed has been useful as a research tool. It has helped at least us to formalize and explicate, with increasing depth and precision, basic notions such as "problem", "coping", "hypotheses", "representation". It led (and was aided by) the design of several computer programs which are of great value for testing and stimulating ideas. It has led to theorems about certain learning algorithms [25]. It stimulated psychological experiments on problem-recognition and shifting of representation in children and adults [4, 15, 16].

The ideas and steps towards demonstrating the existence of a general learning algorithm provide new ways of looking at various real phenomena. Consider two instances: the skid-row effect and the maintenance of a delusion.

By the skid-row effect we mean a trapping phenomenon such as encountered by an alcoholic whose self-esteem is low. He behaves consistently with this hypothesis. His actions—compulsive drinking—have consequences that confirm his hypothesis. Our conceptualization enables us to specify a variety of conditions that can bring about and maintain this effect. Failure for the ordering on the states of the external environment to correspond with the ordering on the internal states is an example.

A delusion may be viewed as an hypothesis that is used despite its being falsified. This may happen if input data relevant to such an hypothesis is screened out, or if it is never stored in $B(t)$, or stored but never attended to [1, 7].

REFERENCES

1. Abelson, R., and Carroll, J. D. Computer simulation of individual belief structures. *Am. Behavioral Sci.* **9** (1965), 24–30.
2. Amarel, S. On the automatic formation of a computer program which represents a theory. *Self-Organizing Systems*, Spartan Press, Washington, D.C., 1964.
3. Amarel, S. Representations and modeling of problems of program formation. *Machine Intelligence* **6**, Edinburgh Univ. Press, Edinburgh, 1970.
4. Badre, A. N. On hypotheses and representational shifting in ill-defined problem-situations. Ph.D. Thesis, Univ. of Michigan, Ann Arbor, Mich., July 1973.
5. Bell, J. L., and Slomson, A. B. *Models and Ultraproducts*. North-Holland, Amsterdam, 1969.
6. Buchanan, B. G., Sutherland, G. L., and Feigenbaum, E. A. Heuristic DENDRAL: A program for generating explanatory hypotheses in organic chemistry. *Machine Intelligence* **5**, Edinburgh Univ. Press, Edinburgh, 1969.
7. Colby, K. Simulation and change in personal belief systems. *Behavioral Sci.* **12** (1967).
8. Doran, J. Experiments with a pleasure-seeking automaton. *Machine Intelligence* **3**, Edinburgh Univ. Press, Edinburgh, 1968.
9. Feigenbaum, E. A., Buchanan, B. G., and Lederberg, J. On generality and problem-solving. *Machine Intelligence* **6**, Edinburgh Univ. Press, Edinburgh, 1970.
10. Kochen, M. Experimental study of strategies in "hypothesis-formation" by computers. *Information Theory*, Butterworth, London, 1960.
11. Kochen, M. Cognitive mechanisms. IBM Rept. RAP-16, IBM Corporation, Yorktown Heights, N.Y., April 1969.
12. Kochen, M. Stability in the growth of knowledge. *J. Am. Soc. Inform. Sci.* **20** (1969), 186–197.
13. Kochen, M. Cognitive learning processes: an explication. *Artificial Intelligence and Heuristic Programming*, Edinburgh Univ. Press, Edinburgh, 1971.
14. Kochen, M. *Information Retrieval Systems Theory*. Wiley, New York, forthcoming.
15. Kochen, M., and Badre, A. N. Question-asking and shifts of representation in problem-solving. *Am. J. Psych.*, to appear.

16. Kochen, M., Badre, A. N., and Badre, B. The process of formulating mathematical problems: assessment and improvement. Fourth Annual Interdisciplinary Structural Learning Meeting, April 6–7, 1973; *Instructional Sci.*, to appear.
17. Lederberg, J., and Feigenbaum, E. A. Mechanization of inductive inference in organic chemistry. *Formal Representation of Human Judgment*, Wiley, New York, 1968.
18. Lederer, W. J., and Burdick, E. *The Ugly American.* Norton, New York, 1958.
19. Pólya, G. *Patterns of Plausible Inference.* Princeton Univ. Press. Princeton, N.J., 1954.
20. Simon, H. A., and Siklossy, L. (eds.) *Representation and Meaning.* Prentice-Hall, Englewood Cliffs, N.J., 1972.
21. Smale, S. Topology and mechanics. *Invent. Math.* **10** (1970), 305–331; **11** (1970), 45–67.
22. Smale, S. Applications of global analysis to biology, economics, electrical circuits and celestial mechanics. Lectures at Am. Math. Soc. Meetings, Univ. of California, Berkeley, Calif.
23. Gold, E. M. Universal goal-seekers. *Information and Control* **18**(5) (1971), 395–403.
24. Myhill, J. Some philosophical implications of mathematical logic. *Rev. Metaphysics* **6** (1952), 105–198.
25. Hantler, S., and Kochen, M. ASP: a program using stored hypotheses to select actions. *J. Cybernetics* (1974), to appear.

Received June 1973; revised version received March 1974