Genetic Psychology Monographs, 1980, **102**, 61-116.

A MODEL OF DEVELOPMENTAL AND PSYCHOLOGICAL PROCESSES*

Department of Educational Psychology, University of Texas College of Education, Austin, Texas 78712

MARK H. BICKHARD

* Received in the Editorial Office, Provincetown, Massachusetts, on April 30, 1979.

SUMMARY

This work is a formal exploration of the relationship between biology and knowledge. Part One elaborates a mathematically formal explication of knowing and knowledge, and relates this formalism to the biological characteristics of adaptation and adaptability in living systems. This formalism and its biological interpretation serve as the basis upon which ensuing discussions are grounded. Part Two argues that living systems encounter a certain increasingly adaptive path of potential evolution, a path definable in terms of particular elaborations of the formal model of knowing. Steps along this path are heuristically identified with the evolution of learning, emotions, and consciousness, respectively. In Part Three, such biological considerations are temporarily set aside, and the formal logical properties of the knowing model are examined. It is shown that the knowing model implies the existence of a hierarchical constraint on knowing, and it is argued that this knowing hierarchy underlies the Piagetian developmental stages. Part Four reunites the bilogical and the logical considerations. It is pointed out that biological consciousness constitutes level two in the logical knowing hierarchy, and then argued that the biological evolution of consciousness is a prerequisite to the cultural evolution of any higher levels in the knowing hierarchy. The form of the argument is that such cultural evolution requires symbolic language, and that symbolic language will not evolve except in a species that is already biologically capable of consciousness. A final section discusses these results as they exemplify a general approach toward modeling in psychology.

I. INTRODUCTION: KNOWING, CONSCIOUSNESS, AND COGNITION

The fundamental thesis explored in this monograph is that the Piagetian cognitive developmental stages are naturally emergent from the formal nature of knowing. The general argument is that the logical character of knowing constrains any knowing system to a specific hierarchically organized sequence of (classes of) potential objects of knowing, and this formal hierarchical sequence can be identified in the cognitive stage sequence described by Piaget (32).

A second theme in the discussion is the plausibility of the evolution, that is, the natural existence, of such hierarchically structured knowing systems. It is argued that such an evolution would occur in two basic steps: first, the physical evolution of a species with a particular system structure, followed by the cultural evolution within that species of higher levels of the knowing hierarchy. The biological considerations of the evolutionary argument serve to complement the logical considerations of the argument for the knowing hierarchy, and together they contribute to an understanding of the existence of thought in physical systems—of the relationship between biology and knowledge (31).

The presentation of the argument begins with a general discussion of the characteristics of knowing processes, and of the constraints that these characteristics impose on any model of knowing. A formal model of a general knowing system is then defined within abstract automata theory, and shown to meet these constraints. Analysis of one of the model constraints leads to an explication of biological stability and adaptability within the automata model, and it is argued that knowing in at least a trivial form is a characteristic of any living system.

This initial development of the knowing model thus introduces the two basic themes of the monograph—the logical and the biological characteristics of knowing. The next several sections pursue the biological considerations with an analysis of a potential evolutionary path beginning with the basic knowing model. The path consists of a sequence of abstract systems or machines beginning with the automata knowing model, each one a modification of the preceding, and each one more adaptable than its predecessor. The sequence is thus presumed to identify a path of potential system evolution of monotonically increasing adaptability.

The sequence of machines ends with a two-layered knowing model, with the first layer knowing the environment and the second layer knowing the

first. This will in a later section turn out to be the critical structure for the cultural evolution of higher levels in the knowing hierarchy, and the evolutionary plausibility of this machine sequence will thus constitute the plausibility of the initial physical step in the evolution of a hierarchically structured knowing system.

Processes within the two-layered knowing model are heuristically identified with reflexive consciousness, and processes in other machines in the sequence are identified with learning and emotions. These identifications require their own explications and defenses. For the purposes of this monograph, however, the function of the discussion is to argue the plausible evolution of a particular system structure, independent of the relationship of that structure to standard psychological terms. The identifications are therefore left at the suggestive level, with their elaborations and defenses postponed.

In the next several sections, the biological considerations of the evolutionary argument are set aside and consideration of the formal logical characteristics of knowing is begun. The first step is to transfer the basic knowing model from automata theory to Turing machine theory—a more convenient formalism for these purposes.

Within the Turing machine model, it is shown that the objects of knowing are necessarily arranged in a certain hierarchical sequential structure of indefinite extent. Several properties of this hierarchy are derived which are identical to properties attributed to the Piagetian cognitive stage structure, and the identification of particular cognitive stages with particular levels in the hierarchy is addressed. The identification arrived at suggests several differences in emphasis and conceptualization from standard versions of Piaget's model (10). It is emphasized that the formal hierarchy is derived from an independent conceptualization of knowing, and thus can serve as both an explanation of the stage structure and an independent source of hypotheses about that structure.

A second theme addressed in these sections is the adequacy of the formal knowing model to human knowing—can the Turing machine model of knowing be taken as an adequate representation of human capabilities? The argument occurs in two basic parts: (*a*) a discussion of the relationship between the formal knowing hierarchy and formal logic taken as proxy for human thought, and (*b*) a discussion of some of the inadequacies of formal logic to human thought, and of the likelihood that the Turing machine model escapes these limitations of formal logic. The general point receives strong support from Turing's thesis in the mathematical literature (39), a thesis that the general Turing machine model is adequate to any realizable process,

and, in turn, the knowing model can be viewed as an exploration of Turing's thesis relative to knowing processes.

The Turing machine model establishes the existence of an abstract knowing hierarchy; it does not establish the likelihood of a physical system which in fact exhibits interesting parts of that hierarchy. If the identifications of the formal hierarchy with the Piagetian stages are accepted, then human beings constitute *prima facie* proof of the existence of such systems. They do not, however, establish whether or not such systems are expectable on general grounds—they establish the fact, but not the plausibility, of such existence. The evolutionary sequence of machines began to address this issue, and it is now taken up again in conjunction with the formal knowing hierarchy developed in the intervening sections. Essentially, the discussion of the machine sequence argued the plausibility of the biological evolution of a certain system structure, and it is now argued that a species with such a structure, but not with any simpler structure, could be plausibly expected to develop further levels of the knowing hierarchy in a formal logical sense rather than in a physical or biological sense. It is argued that the critical new capability obtained with this system structure is symbolic language.

This discussion thus reunites the logical and the biological themes introduced in the early sections defining the basic knowing model: logical considerations argue for a hierarchical constraint on any knowing system, and biological considerations argue for the plausible existence of systems actually manifesting those constraints. The separate discussions of the biological and the logical in the middle sections are thus seen to complement each other in the explanation of the existence of hierarchical knowing systems. A final section presents a general discussion of these results, and of the general strategy of modeling that they exemplify.

II. KNOWING: THE BASIC MODEL

A. Interactive Knowing: Characteristics of the Model

There is a sense of epiphenomenality about knowing in psychology. It seems to be associated with mentalism in a pejorative sense of the word, and to be accordingly avoided and considered unsuited for scientific study. And yet, whatever we want to make of knowing from a phenomenological perspective, it is clear that when we refer to an organism *knowing* something, we are referring to something going on in ordinary physical reality—something that might be worth investigating.

There is an active sense of "knowing" in which we wish to indicate that an organism is engaged in knowing something right now. Presumably, in this sense "knowing" is referring to some kind of relationship between the knower and whatever is being known. I will tend to reserve "knowing" for this active process sense, and to use "knowledge" to refer to a capability of knowing. There seem to be many kinds of knowing, and a conceptual analysis is clearly in order if we are to keep the meanings straight, but these two distinctions will be sufficient for now.

Knowing as physical process must be some special kind of process—not everything is counted as knowing—and at least broadly characterizing that kind of process would be a reasonable next task. Such a characterization is always part definition and part explication; its correctness can never be proven, only its usefulness and appropriateness defended. I will propose a characterization that, in one way or another, underlies most of the rest of the model. I will not at this time give much explicit attention to the defense of the explication, though its apparent usefulness in later discussions constitutes an important kind of defense, but propose it explicitly because it serves as one of the primary integrating concepts for, and thus helps to make sense out of, much of what follows.

Consider a system in an environment. If the system engages in some process which is at least in part determined by the environment, and if the system can in some sense differentiate among the internal outcomes of that process, then those different outcomes will correspond to differing influences from the environment, and, finally, those different influences will correspond to differences in the environment. Thus the different outcomes impose a structure on the environment, grouping together those parts and characteristics that yield the same outcome, and differentiating among those that don't. Furthermore, if the system engages in the process and arrives at a particular outcome, then it *knows* that it is in one of the environmental

circumstances that yield that outcome. This I take to be the basic sense of knowing.

Any living system is engaged in processes that depend on the environment, and these processes eventuate in life or death for the system. Furthermore, life and death are differentiated by the differences in processes that ensue from each. Thus, in a broad sense, knowing as explicated above is an intrinsic characteristic of any living system. What is known may not seem particularly interesting to some—perhaps "existence"?—and is certainly not structured or differentiated, but the point is that the basic relationship is there in the simplest life forms, that "common" examples of knowing are highly sophisticated versions of it, and that the development of these sophisticated versions is intrinsic to the evolution of sophisticated organisms.

The model to be presented is a model of knowing, but that is only one of its characteristics. It is also a model of biological adaptiveness, of consciousness, and others, and it has the ambition to have the potential for saying interesting things about many parts of psychology. Correspondingly, it could have been approached from any of several directions—general evolution, problem solving, adaptive regulation, etc.—like any other thing, it shows different profiles and different details from different perspectives. It is being developed from the perspective of knowing because that was how it originated, and that is the perspective most related to the application of the model to be made at this time. Other perspectives, especially evolutionary adaptiveness, will remain explicit themes, but most will be unexplored.

The general characteristics of the model, then, will depend greatly on the general characteristics of knowing. I have proposed an explication of knowing, but that constitutes, if anything, only the minimal characteristics for a model to be of knowing at all, and doesn't begin to explore the possible characteristics of knowing—that is, the maximal constraints on an adequate model of knowing. I will develop a few more characteristics of knowing, and argue the constraints they impose on any adequate model of knowing (and, by extension, on any sufficient model in psychology).

It is clear that certain passive processes can be knowing processes by the above explication—for example, a system can engage in a recording process whose outcome depends only on an initial input, and thus serves to classify that input. But it is also clear that most interesting knowing processes are interactive in the sense that parts of the process are conditional on the environmental results, the environmental feedback, from earlier portions of the same process. A clear example is the interactive dance by which the stickleback recognizes a mate (20): no simple input suffices; it is rather the

responses to earlier outputs that are critical. The same is true, however, for almost any complex activity, from eye-hand coordination to stalking prey to playing chess.

The model, therefore, must be adequate to such conditional interactions. The primary constraint that this imposes on the model is that it contain some minimal model of the environment within which the conditionals, the environmental process from system output to subsequent input, can be represented. Without such a minimal environment, such as in S-R models, conditional interactions cannot be formally defined or analyzed.

Many conditional interactions are not simply sequential, but are goal directed in the sense that the system's processes tend toward a goal condition, and tend to counter or avoid deviations and obstacles on the way. This familiar fact has myriads of examples—from posting a letter to checking for a tautology in logic—and is hardly in need of demonstration. Furthermore, as with conditional interactions, goal directed processes are not, in general, reducible to or replaceable by simpler processes. All possible situations have to have been "anticipated" for a strictly sequential interaction to proceed, but a goal directed feedback system is capable of generating appropriate responses to situations that have never before been encountered. Also, much of what we know about an environment depends not so much on indicators the environment has presented to us but on indicators that we have "deliberately" (goal orientedly) induced from the environment. That is, we detect, test, and accomplish, as well as merely receive. Thus the model must be adequate to goal directed processes.

There is an additional important reason for the necessity of goal oriented processes in the model: only with respect to a goal condition can a general learning process be determinate. Given a system that can "learn"—that is, alter its internal processes—the general logic is that those alterations will be random and arbitrary unless they are with respect to some standard or rule applied to the processes; and approaching that standard or following that rule thus becomes the goal around which the learning is oriented. In other words, if learning is to involve any kind of trial and error correction, however simple or sophisticated, it must involve some standard or rule, a goal, with respect to which "error" can be defined. Furthermore, if the learning is to be functional for the system, the learning "goal" must in some sense be equivalent to the improvement of the ability of the system processes to reach their process goals.

The issue can be illustrated with the concept of a pattern recognizer. A pattern recognizer is a device that receives inputs (often thought of as or-

ganized into a grid or retina), engages in internal processes, and emits an output. The output is taken to classify the input, and a pattern is said to be recognized if all of its instances are classified as a distinct group; similarly, an instance is recognized if it is classified in the "appropriate" group. It is also relevant to consider how the recognition capabilities of the device vary with differences in learning processes. (Strictly, the device can be thought of as a responder, and the issues are what kind of response rules can be manifested and learned. Treating the outputs as classifications of the inputs is incidental to the basic process.)

The earlier discussion about goal oriented processes amounts, in this context, to an argument that some things cannot be *recognized*, or "perceived," without interactive goal directed processes. The issue does not seem to have been addressed in exactly that form, but related discussions and proofs can be found in Minsky and Papert (26), Hopcroft and Ullman (15), and MacKay (19). It is the dependence of *learning* on goal directedness, however, that is made especially clear by the pattern recognition perspective. The critical question is, how can the recognizer know which of the infinity of possible recognition rules (classification rules, response rules) is the "correct" one—the one it is "supposed" to learn; how is it to know when its learning is successful and when it has still more to do? Clearly, any particular recognition rule is completely arbitrary from the perspective of the recognizer, and therefore the "correct" rule can only be determined with respect to some external standard. Consequently, the recognizer can know if a particular classification is right or wrong, and therefore if its current classification rule is working or not working, only if some comparison with the external standard is made and the result is fed back to it (unless some direct comparison with the "correct" rule itself is possible, in which case we have some kind of copying or duplication process, not a learning process). Furthermore, if the learning is to be effective, it must in some sense seek "correct" feedbacks to the recognition processes as a goal. In other words, only with respect to a goal condition can a general learning process "know what to learn."

For the formal recognizer, the rule to be learned is predetermined by the E, and the learning process to "correct" and "incorrect" feedbacks is defined with respect to that rule. In the equivalent biological case, on the other hand, the learning goal condition is the prior element—for example, hunger satisfaction—and it is with respect to that goal that response rules are to be defined as "correct" or "incorrect." "Correct" is whatever is necessary or at least sufficient to get the goal "response" from the environment. Note that, even in the strict recognition version, the rule learned need not be identical

with the external standard; it need only be functionally equivalent to it (that is, produce the same outputs for the same inputs). For example (44), for N an integer, $y = N^2$ is equivalent as a function to $y = \sum_{i=1}^{|N|} (2i - 1)$.

The necessity for the model to be adequate to goal directed processes imposes a seemingly obvious constraint: the model must include a sophisticated language for defining and analyzing processes internal to the system. Earlier it was argued that the conditionally interactive structure of behavior required a minimal attention to the environment in order to be adequately modeled—enough "attention" to at least define an environmental feedback. It might conceivably be thought that, as well as being necessary, an environmental perspective was sufficient to model the complexities of behavior; this, of course, would ignore the processes underlying that behavior, but those processes might be considered a separable problem, perhaps interesting in itself, but not necessary to the analysis of the behavior *per se*. Some radical behaviorists seem to have at times made exactly this claim.

The environmental, or behaviorist, restriction encounters difficulties even for non goal directed conditional interactions if the conditions start becoming complex. The difficulty is that a single condition test in the system might be met by a large number of possible environmental "conditions," and these might not have apparent (intuitive) commonalities when viewed strictly environmentally. In such a case, the environmentalist can at best somehow discover the set of corresponding environmental conditions and simply list them. The situation is identical in the case of goal conditions, though the consequences can be more striking if an "unexpected" environmental version of the goal is encountered. It is in modeling goal directed behavior itself, however, the behavior "on the way" to the goal, that the internal process language becomes necessary. Very simply, feedback and servomechanism goal seeking processes can correspond literally to infinite (and almost always "very large") sequences of possible behaviors in approaching those goals; no simple listing strategy will suffice. If "process" and "behavior" languages were really equivalent, then nothing would be gained by avoiding the conceptual and notational efficiencies of process languages. If they are not equivalent, then the behaviorist must show that process languages are (*a*) unnecessary and (*b*) undesirable. The feedback example indicates their necessity; and their power in expressing and analyzing complex conditional strategies and logics indicates their desirability.

I have argued that knowing processes can be conditionally interactive and goal directed, and that these characteristics impose the constraints of in-

cluding the environment and providing sophisticated process languages on any adequate model of knowing. The characteristic of goal orientedness imposes still another constraint, though of a different kind. The goal definition in a goal directed process must come from somewhere. If it is "innate" to the structure of the system, then the problem of where it originates gets transferred to biology and evolution, but, clearly, many goal definitions are determined in an ongoing fashion by other processes. Thus we can have some processes receiving goal definitions from others, these in turn receiving goals from another "layer," and so on. But clearly such a hierarchy must stop: there must be some highest level goal(s). Furthermore, that highest level goal must in some sense be "innate" or "wired in"; there is no higher level process to provide it dynamically. The necessity for accounting for such a highest "integrating" goal (all others can be viewed as subgoals) is the additional constraint imposed on the knowing model by goal directedness.

In addition, the integrating goal must, in the case of biological knowing, have some close relationship with survival and adaptability. To the extent that it is able to maintain its goal condition of "alive," a living system has knowledge of its environment in the sense explicated earlier. The greater the scope, power, flexibility, and differentiation of that survival ability, the greater the knowledge—and conversely. A model of biological knowing must somehow account for the intimate relationship between knowledge and adaptability (31).

The characteristics of knowing that I have developed have been conditional interactiveness and goal orientedness. The model constraints that I have argued have been inclusion of the environment, an internal process language, and a biologically meaningful integrating goal. I find these views to be similar to those expressed by a number of others: for example, Bruner *et al.* (6), Pribram (35), and Taylor (45). The general viewpoint of analysis by synthesis in cognitive and linguistic psychology (14, 28) seems to me to be particularly compatible with the above conceptions of knowing, but it is developed primarily with respect to problems in perception. The most closely related discussions have been by MacKay (e.g., 18, 19) and Piaget (e.g., 31). MacKay has concerned himself with interactive models of knowing and meaning, and has generally used precise and powerful languages for his models, but seems to have paid relatively little attention to the biological integration of his general conception. Also, I find his interactive analysis incomplete—he analyzes the "knowing" or "meaning" of a set of inputs in terms of the system's matching responses to those inputs—certainly interactive in spirit—but seems to leave the concept and process of "matching" completely unanalyzed. Piaget, on the other hand, has been greatly con-

cerned with the biological and evolutionary foundations and contexts of his model, and they are explicitly interactive, but they have been stated in such imprecise and ambiguous informal language as to sometimes obscure even the general points, let alone the details. Nevertheless, I find the intent and spirit of Piaget's model to be closely related to my own, and at least the epistemological part of the model to be presented can be viewed in part as a potential formalization of Piaget's model of cognitive epistemology.

B. Automata Theory

The process language used in the initial presentation of the model is that of automata theory. The model is not a study of automata theory, but rather makes use of concepts and vocabularies from automata theory in its own development. This section introduces these basic conceptual tools. Automata theory results and concepts are taken primarily from Minsky (25) and Ginzburg (12).

An automaton is a type of abstract model of a machine or system. It considers that a machine receives inputs, engages in internal processes in some determinate way, and emits outputs; and it attempts to represent these processes formally. In a physical machine or system, at any particular time there may be many situations and conditions in the machine which, together with future inputs to the machine, will determine the future behavior of the machine. An automaton model of a machine will abstract all of these simultaneous situations and conditions into a single summary or index symbol, called a *state* of the machine, so that a single state of an automaton will, together with its inputs, determine the machine's behavior.

An automaton in a particular state will receive an input, engage in various processes, and arrive in a few moments at a new set of situations and conditions indexed by its own state symbol. Since the state symbol summarizes all relevant conditions in the machine, the current state plus the current input must completely determine what the next state of the machine will be. Such a movement from one state to another is called a *state transition,* and the determinancy of such transitions is expressed in *transition rules* of the form "if the current state is X, and the current input is Y, then the next state is Z." Thus an automaton is conceived of as moving around in the set of its possible internal states, in accordance with its transition rules and the inputs it receives; an automaton, in fact, is taken to be defined by its set of states and its transition rules.

A useful way of conceptualizing an automaton is in terms of what is called a state transition diagram. A *state transition diagram* is a diagram of points

connected by arrows in which the points represent states of the automaton and the arrows represent the transition rules. The points are commonly labeled with the appropriate state symbols, and the arrows with the input symbol that induces that particular transition.

Many interesting properties of an automaton can be studied without regard for its outputs—considering the automaton as a strictly passive system. An automaton that does have outputs, and for which those outputs are determined by the current state, is called a *Moore machine*. The concept of a Moore machine will be central to the model.

Many other useful concepts, such as subroutine, servomechanism, etc., can be defined in terms of an automata model, and I will sketch a few of these definitions. In the state transition diagram for an automaton, there may be many subdiagrams of the overall diagram that are internally identical and differ only in the transitions into them and out of them. These subdiagrams thus engage in identical processes, but are entered from and exit to different places in the main diagram; a set of such identical subdiagrams is called a *subroutine*.

The exact description of a goal seeking process depends on the goal being sought, and the environment within which it is to be sought, but the general structure will be a goal condition test attached to some procedures for attaining or maintaining the goal condition. These procedures can involve subsidiary tests, other goal seeking processes, etc., and, for that matter, so can the original goal condition test. Note that a goal seeking process may or may not be a subroutine: that is, may or may not have multiple entry and exit paths.

If there is a set of goal seeking subroutines which differ among themselves only in particular characteristics of the goal definitions, then that set can be considered a *servomechanism:* that is, a goal seeking subroutine for which the goal definition can be determined external to the subroutine itself—determined, at least, within the set of available goal definitions. If the available goal definitions in a servomechanism exhaust some space of possible goals—e.g., positions on a plane—then the servomechanism is of special usefulness in that it absorbs all concerns about reaching such goals (if it works), and leaves other processes only the problem of selecting the goal, not attaining it. The goal attaining procedures may, of course, be of varying sophistication and power.

As an automaton receives a string of input symbols, it will shift from state to state in accordance with its transition rules, with the particular path of transitions being determined by the particular string of symbols being re-

ceived as input. If an automaton is started in some particular initial state, some input strings will induce various differing transition paths, but will leave the automaton in the same final state when the strings are finished. Other input strings will leave it in a different final state when the input is finished, and so on. If a particular state of an automaton is chosen as a starting state, then the automaton will differentiate all its possible input strings into subsets depending on which final state the strings will leave the automaton in—strings leaving it in the same final state grouped together, and strings leaving it in different final states grouped separately. The state that an automaton is in thus carries information about the string of inputs that brought it there.

An automaton in which a particular initial state and some set of final states has been selected is said to "accept" or "recognize" any input strings that move it from the initial state to one of the specified final states. Such an automaton with its initial and final state selections is called a *Rabin-Scott automaton* or a *recognizer*, and one of the main theorems in automata theory characterizes the kinds of input strings that can be "recognized" in this sense by an automaton.

This sense of recognition is essentially a passive version of knowing as explicated earlier, and the first step of the model is to extend it to an interactive and potentially goal seeking mode.

C. An Interactive Machine

Consider two Moore machines arranged so that the outputs of each one serve as the inputs of the other. Consider one of the Moore machines as a system and the other as its environment, and let the system have the initial and final state selections that make it a recognizer.

The system can thus recognize input strings in the standard sense in automata theory. In this interactive configuration, however, an input string corresponds to—is generated by—a state transition sequence in the environment. The set of recognizable input strings thus corresponds to the particular set of state sequences in the environment that could generate them. The recognition, or knowing, relationship is thus extended from inputs to situations and conditions in the environment.

Furthermore, during an interaction, the environment is receiving outputs from the system, and it is these outputs from the system that induce the environmental state transitions that generate the inputs to the system that the system either recognizes or doesn't. Thus the "recognition" process is no longer strictly passive; the "recognized" strings are induced from the envi-

ronment by the system's own outputs. In fact, the interaction doesn't need to be viewed as a recognition process at all. It is equally as much a construction or transformation process (constructing the situations and conditions corresponding to a "recognizable" environmental state sequence) or a test or detection process (detecting an initial state of a "recognizable" environmental state sequence) and so on. (Note that the end states of a set of recognizable sequences could be a subset of the initial states—so that a test process, for example, could leave unchanged the conditions it tests for.)

The system need not be thought of as a single undifferentiated recognizer. It could be, for example, a collection of recognizers connected to each other, say, with the final states of one attached to the initial state of another. Such connections could induce functional relationships among the recognizers, such as one recognizer testing for the appropriate conditions for another to begin, or a servomechanism being used to create a subcondition for another process to proceed, etc. In discussing such interconnected systems, an interaction or process that reaches a final state will be said to be *complete* or *successful*, otherwise *incomplete* or *unsuccessful*; and, once an interaction has begun, strings and "environments" that will complete it are said to be *expected*, otherwise *unexpected*. Thus, for example, a goal oriented process expects to reach its goal, and is complete or successful when it has done so. "Construct," "test," "transform," and others will be used with the general heuristic meanings suggested above, and, in accordance with standard usage, a transition to a subroutine will be a *call* to the subroutine.

The model so far constitutes a minimal logical model of knowing: it is explicitly interactive and easily goal directed and it uses a powerful formal process language. It doesn't, however, propose an integrating highest level goal, and certainly doesn't relate such a goal to biological processes. These issues are addressed in the next section.

D. Stability in an Interactive Machine

In this section, I will discuss biological stability and adaptability in the interactive configuration, and show that it constitutes a special goal directed case of a recognizer. In this configuration, stability in an environment will turn out to be equivalent to knowledge of the environment.

Biological stability and adaptability correspond roughly to the ability of an organism to persist and survive in an environment—to maintain itself in interaction with that environment. In the interactive automata model, it is not immediately apparent what corresponds to such an interactive stability. The system is constituted in the model by an automaton—a state transition

diagram. So, as long as the diagram remains the same, the machine is the same: that is, stable. But interactions with the environment are presumed to occur in *accordance* with the transition diagram, not as operations on it, so it is not clear how the system could ever change, how it could ever be unstable. It might seem that the interactive model is stable by definition.

Suppose, however, that the machine goes off into some part of the diagram that it can't get out of—there are no transitions back. Then, in a functional sense, the machine is no longer the same because there are processes it was once capable of that it isn't any longer; it can't reach them anymore. Thus change in the formal model is constituted by absorption in some part of the transition diagram, and stability is, conversely, the maintenance of the ability to reach in the diagram whatever could originally be reached.

Whatever could "originally" be reached, was reachable from some "original" state; the machine is always in some state or another. Consequently, as long as that original state is reachable, anything once reachable from it is still reachable. Thus, reachability of the original state implies maintenance of original diagram reachability. Conversely, if the original state is not reachable, then at least one state that was once reachable, the original state, is no longer. Thus, maintenance of diagram reachability implies maintenance of reachability of the original state, and diagram reachability and original state reachability are equivalent characteristics. In other words, machine stability is equivalent to the maintenance of reachability of some initial state in the machine.

If we assume that the processes of the machine tend to maintain this stability in a goal directed sense, tending to "avoid" dangerous (absorbing) parts of the diagram, and to "approach" safe parts of the diagram, then machine stability becomes the integrating goal for other goal oriented subprocesses in the system: successful completion of a subprocess will constitute, for that process at least, successful maintenance of stability as reachability.

Successful maintenance of stability means successful maintenance of reachability of some "original" or "central" state and, perhaps, a return to this central state from time to time. But this makes the central state into both an initial and a final state in the "recognizer" sense, and thus the whole system into a kind of "cyclic" recognizer. Maintenance of stability thus corresponds to the receipt of input strings that are still "potentially" recognizable (that is, that could still lead back to the central state) and thus corresponds to a successful interactive knowing of the envorinment in the

sense originally explicated. Stability in an environment is thus equivalent to knowledge of that environment, the capability of knowing that environment.

Furthermore, as pointed out before, such knowing is constituted as much by the induction of appropriate inputs as by the receipt of them—the machine must control the environment as much as recognize it. Knowing requires the ability to counter, or to anticipate and prevent, inputs from the environment that tend toward absorbing conditions, and to induce from the environment inputs that move the system along safe reachability paths in the diagram—knowing thus requires knowledge of what the environment is doing, might do next, and might be induced to do with appropriate inputs. [Knowing thus encounters Ashby's law of requisite variety (1).] All of these are inextricably present in the transition diagram of a successfully stable system.

We now have an interactive knowing model with an integrating goal of machine stability. The correspondence between machine stability and biological stability remains to be explicated. The link is essentially that the "central" state—or some set of equivalent mutually reachable "central" states—must be compatible with the physical and physiological well being of the organism. That is, maintenance of diagram reachability must accomplish the protection of biologically safe physiological tolerances. But this suggests an obvious identification: let the "central" states be those states which are themselves most "central" to the safe physiological tolerances; the automaton states, after all, are just abstractions for conditions in the system in the first place. Thus "central state" means "state of biological well being," and machine stability corresponds to biological stability. [This relates to Ashby's concept of stability as regulated homeostasis (1). Note that one way to enter a central state might be to induce "blank" inputs from the environment (that is, to quiet or terminate an input string); this would correspond to a deviance countering (drive reducing, noxious avoiding) homeostasis. Note also, however, that in this model this wouldn't necessarily be the only way to get back to a central state.]

At this point, we encounter the evolutionary perspective. Biological stability is the integrating goal of all life processes; it is perhaps definitive of it. The process of evolution, then, may be seen as the elaboration of systems that are stable in new environments, and increasingly stable in all environments: that is, of systems with new knowledge, and of systems with more knowledge. It is the systems with more knowledge that I will be concerned

with, for I will be interested in some of the general forms it might take, and in some of the processes that might be evolved for generating it.

Note in particular that the model to this point is either interactively stable in an environment or it is not. It has processes and "behaviors" for maintaining that stability in a dynamic sense, and it has available to it information about whether or not it is being successful—whether or not it reaches its central states and subgoals. But it has no way of changing or adding to any of those processes if that should become appropriate; it has no way of learning, of improving its own adaptability.

III. CONSCIOUSNESS: THE EVOLUTION OF KNOWING SYSTEMS

A. An Interactive Machine with Interrupt

Entering a central state indicates success in the maintenance of overall stability. Similarly, subgoals indicate an intermediate success, and failures to reach subgoals indicate potential disruption. The model as defined to this point, however, is unable to respond to such indications; success and failure information is available, but the machine has no way to use it.

The simplest use of such information would be to try to recover interactions that seem to be failing. The general idea would be to detect interactions that were not reaching their goals, to interrupt the machine processes that were controlling the interaction, and to transfer control to some other part of the state transition diagram which could then attempt to continue the interaction. Note that as long as failure detections were not in error, and as long as the transfers had any positive expectation of recovering the interaction, then the interrupt and transfer process would tend to improve the general stability of the machine. A machine which can effect such interrupts and transfers on the first machine will be called an *interrupt machine*.

The interrupt mechanism is essentially a second machine operating in parallel to the original machine, monitoring the first machine for errors and "doing things" to it to improve its stability. The parallel processing monitor constitutes a basic conceptual addition to the model. As its capabilities increase, as the things it can do to the first machine become increasingly powerful, it becomes in turn a learning machine, an inducer of feelings and emotions, and, finally, a machine which knows and operates on the first machine in the same sense that the first machine knows the environment. This latter process of internal or reflexive knowing will be identified with conscious knowing.

The succession of machines beginning with the interrupt mechanism will be presented as a possible path of machine evolution. That is, each machine arises as a "simple" modification of its predecessor in the sequence, and might reasonably be expected to have evolved from it, and, further, it is argued that each machine constitutes an increase in the adaptability of the overall system. The machine sequence thus constitutes a potential path of biological evolution both in the sense of being machine continuous—not involving unexplained qualitative jumps—and in the sense of involving a monotonically increasing gradient of biological adaptability.

B. An Interactive Machine with Adaptive Self-Organization: Learning

The recovery attempts of the interrupt machine will not always be successful: some interrupts and transfers will lead to processes which also fail and require still further recovery attempts. It would be advantageous if the interrupt machine could detect such recovery failures and change its interrupt and transfer rules so that the same transition under the same interrupt conditions would not be repeated; that is, it would be advantageous if the interrupt machine could avoid past mistakes. Clearly, it would also be useful if such rule changes did not unduly interfere with previously determined successful interruption transitions. Such an interrupt machine would tend to avoid unsuccessful interrupt transition rules and stabilize around successful ones.

Such stabilized successful interrupts would not really constitute interaction failures from the perspective of the general system, and it would be correspondingly advantageous if they ceased to count as recovery failures for earlier interrupt transitions leading to them—that is, if new interrupt transitions leading to stabilized interrupt transitions were counted as successful. Note that, with this addition, such a stabilized interrupt transition functions identically to a simple state transition: in both cases, when the appropriate state input pair is encountered, a determinate state transition occurs, and in neither case is the occurrence of that transition treated as an interaction or recovery failure.

In other words, a machine with these properties will tend to learn successful new state transition rules—and therby learn to know corresponding new processes in the environment. There are many issues involved in such learning processes, but the important characteristic for following sections of the model is that the original interrupt mechanism can now construct successful new state transition rules in the first level machine.

C. An Interactive Machine with Harm Avoidance: Pain

One characteristic of the learning machine is that it tends to avoid *interrupts;* transitions leading to (nonstabilized) interrupt conditions tend to be discarded as failures. Correspondingly, the machine tends to avoid *regions* of the state transition diagram that tend to yield interrupt conditions; transitions leading to such regions tend to be discarded. One example of such a region would be one in which there were many undefined state-input transi-

tion conditions, which would simply halt—producing an immediate interrupt—if such an undefined condition occurred. Note that such a sparsely defined region would be a poor representation of the environment. (Strictly, the critical characteristic is the probability of entering an interrupt condition, which may or may not be related to the density of undefined conditions in the area. For example, a region in which there was only a single defined transition path, but in which the outputs successfully forced the environmental inputs needed to stay on the path, would not tend to yield interrupts, and thus not tend to be avoided. Note that such a path would constitute a successful representation of its environment.)

Suppose there was a special class of inputs that always induces interrupts—i.e., for which no transitions are defined—then, similarly, any behaviors (regions) tending to induce such inputs will produce interrupts and will be avoided. Furthermore, if a particular input, once induced from the environment, tends to be repetitively persistent—that is, repetitive and difficult to stop—and if it in addition never induces transitions to subgoals by itself, then such an input will tend to interfere with normal processing, causing the machine processes to cycle around without being able to reach goals, and thereby tending eventually to induce interrupts. Behaviors that produce such an input will also tend to be avoided by the machine. An input always inducing immediate interrupts, as in the first case above, is just an extreme case of this. Such an input will be called a *pain input*. The interrupt characteristics of an input, thus, the pain characteristics of an input, are in relation to a particular machine or type of machine; they are not characteristics of the inputs *per se*.

The essential characteristic of pain inputs is that it is impossible (or difficult) to interact successfully with them, and they are correspondingly avoided. [Many "normal" inputs might be pain inputs by this definition if they had (or were artificially given) the characteristic of persistence. That is, many inputs are, perhaps, successfully interacted with by being changed to something else, and if the change didn't work, they might become painful.] Pain inputs would be especially useful to the machine if they originated in the physical entity that constituted the machine and were correlated with damage to that entity. The critical characteristic of pain inputs for following sections of the model is that they suggest the concept of learned approach-avoidance tendencies, or "valence" tendencies, within the system's behaviors and with respect to the system's inputs. This is in no way incompatible with a view of pain signals as having instinctive or reflex withdrawal responses; it simply focuses on the learning aspects.

D. An Interactive Machine with Internal Feedback: Emotion

I have shown that the learning characteristics induce approach-avoidance tendencies among behaviors depending on their interrupt consequences. In this section, I will argue that there are strictly internal characteristics of machine processes that contain viability information—that is, that indicate that the ongoing process should be sought again or avoided. Therefore, it would be advantageous if feedbacks of these characteristics could be input to the machine in such a way as to induce appropriate approach-avoidance tendencies.

The basic idea is that, sometimes inputs from the environment that the machine is not ready for (require interrupts) indicate that the current machine processes are highly—therefore dangerously—inappropriate to existing conditions. Such interrupts and associated processes should therefore be especially avoided. They are characterized by being "long" or "detour" interrupts to some new and "hopefully" appropriate processes.

On the other hand, sometimes inputs inducing interrupts indicate rather that the machine was awaiting more complexity or difficulty than it actually encountered. Such interrupts are not to new processes, but simply to later, more advanced, or closer-to-the-goal parts of the same process, and they indicate an overrepresentation of the appropriate environment, therefore an especially safe situation, and therefore should be approached. Such interrupts are characterized by being "short" or "short cut" interrupts in the processing.

How could such information about the "length" of interrupts be generated in the machine? Each change of state in the abstract machine corresponds to a change of physical condition in the physical machine. The physical change can be considered as a construction process—the construction of the physical conditions corresponding to the next state. Presumably, the physical conditions and processes will be related to the logical process structures—in the sense, for example, that conditions within a given subroutine are more similar than conditions between differing subroutines. But this means that the amount of construction involved in an interrupt corresponds to the "length" of the interrupt—long interrupts requiring a lot of construction, and short interrupts requiring little, or even a net decrease in construction when intermediate state transitions are "skipped over" and their detailed constructions thus avoided.

The interrupt-learning processes thus already contain information concerning the length of interrupts—the only addition (evolution) necessary is a

way of encoding it and feeding it back as pain-like inputs for long interrupts, and "anti-pain" (easy to interact with) or "pleasurable" inputs for short interrupts, so as to induce appropriate learning tendencies. Such inputs will be called *feeling inputs*, and considered *positive* or *negative* as they are "pleasurable" or "painful." Feeling inputs function like any other inputs in the sense that they induce machine processes attempting to interact with them. Interactions involving feeling inputs will be called *emotional* interactions.

The structure of feeling inputs is limited to positive or negative, though perhaps of varying degrees. Thus the multiplicity of differentiations and relationships among emotions must in this model be based on other structures of the emotional interaction. To some extent this additional structure is based on other accompanying inputs, and to some extent it is logically arbitrary: thus, for example, a particular negative feeling may be considered as arising from a block to an ongoing interaction (i.e., the original goal remains salient)—giving rise to anger—or it may be considered as the introduction of a new and possibly unsuccessful interaction—giving rise to fear. The logical indeterminancy of reactions to feelings allows a large scope to learning in determining the kinds and tendencies of emotional interactions, and it is also likely that certain basic emotions have direct neurophysiological foundations.

Emotions are closely related to motivations, which are not formally treated in this presentation of the model. Essentially, the model as defined is always active and always under the control of some set of goal definitions. Particular subgoal definitions are often affected by input conditions, and the interplay between the environment and such goal directed behavior constitutes the field for the analysis of motivation in this model. For example, a fall in blood sugar induces signals from the hypothalamus that can only be interacted with by eating (raising blood sugar)—thus the "motivation" of hunger (13).

E. A Reflexive Interactive Machine: Consciousness

With "small" additions, the interrupt-learning-emotion machine described in the preceding sections becomes a second machine engaged in knowing interactions with the first machine in the same sense that the first machine interacts with the environment. Such reflexive knowing will be called conscious knowing, or consciousness. The interrupt machine effects error-induced state transitions in the first machine. The learning machine effects new state transitions which stabilize: that is, function like all other state

transitions. In other words, stabilized learning transitions contain, or constitute, new state transition definitions (rules). If the machine could separate the transition *process* from the transition *definition*—e.g., by initiating a transition, then triggering on its own an interruption back to itself before the transition process is completed—it could, in effect, output state transition *definitions per se* to the first machine.

The interrupt machine is engaged in a constant monitoring of what the first machine is doing and what it is about to do. In particular, if it is about to do "nothing"—an undefined condition—it initiates an interrupt and attempts to recover the interaction. An extension of this monitoring information, from "on or off" to "amount of construction involved," was posited as a part of the development of feelings and emotions. If this monitoring information is now extended to the actual state transition rules being readied or realized in the construction process, then the second machine will be able to read and receive state transition rules as inputs from the first machine.

Thus the second machine would receive and emit as its input and output signals the state transition rules—the basic fundamental components—of the first machine. It would interact and operate on the underlying state transition diagram, taking the structures of and processes within that diagram as the objects of its knowing. The learning machine already outputs state transition rules; it must only develop the ability to dissociate those rules *per se* from the ongoing state transition processes. The emotion process already involves the encoding of the length of state transitions; it must only encode their form as well.

Along with its inputs and outputs, the internal processes of the second machine will presumably have changed as specializations of it developed the new forms of inputs and outputs just defined, and the second level machine would function according to its own particular internal rules and heuristics. It is reasonable to assume that such a second level machine would also benefit from learning and internal feedback (it is very unlikely, in fact, that the second machine layer would ever have existed as a simple switchboard structure—it evolved out of learning and emotion systems and would have retained those characteristics and structures during its evolution), and I will in fact assume that both machine layers have such auxiliary machines. Thus we have two machine layers, each with learning and emotions, one interacting with the external environment, and the second interacting with the first. (This raises a question of why the second machine layer doesn't (hasn't) develop a third machine layer, etc. Essentially this question, though from a ifferent perspective, is the focus of "consciousness and semantic ascent.")

The adaptive value of such a second machine layer is of many forms. Perhaps the most basic form is that the second machine layer can "look ahead" in the logic of the first machine to check on possible consequences of (*a*) current events in the environment, to see if they require anticipatory behavior, or (*b*) possible behaviors of the first machine, to see if they are likely to lead to desirable consequences before actually engaging in them. (Note the assumption that the first machine layer contains information, "knowledge," about the environment.) Eventually, such a second machine layer would be able to engage in deliberate (goal directed) learning and planning, and to deal with higher order logical considerations as discussed in later sections.

The first level machine engages in knowing interactions with the environment. These interactions will involve perceptions, behaviors, anticipations of environmental response, goal oriented manipulations, and so on. Other characteristics associated with awareness may also be identified. For example, to the extent that the environment is familiar to the machine, the state transition structures of the relevant subroutines and servomechanisms will implicitly represent—will constitute an apperception of—portions of the environment that are not currently being directly known (e.g., the "back" of a tree). If we identify such a basic knowing process of the first machine with awareness, then the knowing processes of the second machine level constitute a reflexive awareness or consciousness in the sense of G. H. Mead (23). This would also seem to be consistent with the reflective consciousness of the phenomenologist. It is in this sense that the processes of the second machine level will be called *conscious knowing*, or *consciousness*. (Not all uses of the word "consciousness" involve this reflexive or reflective element; often it is used synonymously with "awareness" or even "wakefulness.")

IV. COGNITIVE DEVELOPMENT: LOGICAL CONSTRAINTS AND POSSIBILITIES

At this point, the considerations of potential evolutionary sequence are temporarily set aside, and the discussion returns to the basic model of knowing. First, the knowing model is translated into a different mathematical framework known as Turing machine theory. Within that framework, it is then shown that the knowing model yields a necessary hierarchical constraint on levels of knowing that corresponds to the Piagetian developmental stages. Finally, in the next chapter, the developmental and evolutionary perspectives are united: it is argued that the biological evolution of consciousness is necessary to the evolution of language, which, in turn, is fundamental to the cultural evolution through still higher stages.

A. Turing Machines

A Turing machine is a Moore machine in interaction with a special environment—a paper or magnetic tape (or some equivalent). The tape is presumed to carry symbols from some alphabet of symbols, and the Moore machine interacts with the tape through a tape head that can read symbols from the tape, write symbols on the tape, and move the tape in either direction. The tape is considered to be indefinitely extendable in the sense that, if the tape head encounters an end of the tape, more tape is spliced on as needed. Turing machine results and concepts have been taken primarily from Minsky (25), Davis (7), Rogers (39), and Hopcroft and Ullman (15).

The Turing machine model with its tape environment and the knowing model with its automation environment can be viewed as special cases of each other, and therefore as functionally equivalent models. The system parts of both models are Moore machines—they differ only in their environments. Therefore, their equivalence must be shown through the equivalence of the environments: the general logic is to reduce the tape model to the automaton model by treating the various tape configurations as descriptions of automaton states, and to reduce the automaton model to the tape model by coding the automation's transition rules on the tape. Among the many details of such mutual reductions, only one seems particularly troublesome: an automation is generally considered to be finite, while a tape, and thus the number of potential tape configurations, is considered to be potentially infinite. This would correspond to a potentially infinite automaton, and potential trouble in the reduction. The automaton model of the environment, however, is not necessarily of some restricted size; it is intended to represent

all of the relevant environment, and is thus in principle the "size" of the universe—a "restriction" that should make little difference relative to a Turing machine (whose tape is also presumably "limited" by the universe.) Given their functional equivalence, one or the other versions of the model can be chosen depending on their convenience for particular purposes. For much of the discussion to follow, the Turing machine version is most convenient, and so I will begin with a discussion of some of the basic results and concepts from the general theory of Turing machines.

The original heurisitic for the formal definition of a Turing machine was a person engaged in some determinate symbolic activity. Such a person could, presumably, receive some symbol representation of a problem, operate on those symbols according some set of rules, using as many pieces of scratch paper as necessary, and, finally, produce some symbolic result or answer. The automaton part of the Turing machine is taken to represent the rules, and the tape to hold the input symbols, the intermediate scratch results, and the final results.

The point of the Turing machine definition was to try to explicate the general concept of an "effective procedure." A procedure is a specified way of doing something, and an effective procedure is a procedure, or a specification of a procedure, that "is workable" (a procedure that contains instructions that are ambiguous or impossible to carry out is hardly "effective.") A Turing machine, then, is a way of formally specifying a procedure.

Turing's thesis is a hypothesis that the concept of a Turing machine is equivalent to the concept of an effective procedure in the sense that any effective procedure can be represented (specified) by a Turing machine. In technical language, this would mean that a Turing machine could do anything that was doable. Turing's thesis can never be proven because "effective procedure" is a basically intuitive concept, while "Turing machine" is a formally definable concept: two things can be formally proven equivalent only if both are formalized, but "Turing machine" is precisely an attempt to formalize "effective procedure" and proving the adequacy of the formalization is precisely the problem. The thesis could be disproven, however, if a more powerful formal model were presented:

Turing's thesis is not without support. Several decades of potential explications of "effective procedure," some amounting to variations on the basic Turing machine definition, and some taking completely different forms, have all been proven either equivalent to or weaker than the Turing machine. That is, a great many potential falsifications of Turing's thesis

have failed to falsify it. Essentially, the basic issue is considered closed, and the interest is on applications and implications.

Turing's thesis, with its great support, claims that the Turing machine *can* do anything that can be done, including anything that a human being could do, and the identification with the earlier model of knowing puts special emphasis on any *knowing* that a human being could do; but the thesis says nothing at all about *how* it would do it, or what constraints would be encountered on the way. Current results, in fact, have relatively little to say about this problem, and what relevant few there are are specific to computer applications. The exploration of knowing from a Turing machine perspective that I will discuss later can in one sense be viewed as an exploration of the cognitive implications of Turing's thesis, or of the formal constraints encountered in realizing Turing's thesis with respect to knowing.

Note that a procedure is a set of steps or rules for doing things. Generally, in interesting cases, these rules will tend to accomplish some result or goal, but there is nothing in the concept of effective procedure as used here that guarantees that the goal will in fact be reached. "Effective" isn't used to mean "effective in reaching a goal," but rather "effective in defining what to do next." A procedure that does guarantee results—such as for multiplying two numbers—is called an algorithm, while one that doesn't—such as for playing chess—is called a heuristic. For examples and discussions of heuristic problem solvers, see Feigenbaum and Feldman (9), Newell and Simon (29), or Slagle (43). (An algorithm for winning in chess is definable—simply explore all possible game sequences and stay on those that win—but would require greater than the age of the universe to be computed.)

One of the most interesting and important positive results of Turing machine theory is the existence of what are called Universal Turing machines: essentially, Turing machines that can simulate any other Turing machines—including themselves—as long as they are given appropriate descriptions, or programs, for the machines to be simulated. One of the consequences of the existence of Universal Turing machines is that, if desired, machines can be studied in terms of their programs for some Universal machine, and this often is conceptually and notationally very convenient. This translatability between "physical" Turing machines and "formal" programs introduces an ambiguity into the nature of existence of Turing machines. This ambiguity does not affect the logical considerations addressed in the next several sections, but it is relevant to other considerations, and its resolution is addressed in a later section.

A Turing machine begins with an initial symbol string on its tape, engages in various processes (computations), and ends with some other symbol string. The Turing machine computations are stopped only when the Turing machine enters a special halt state, and the result of the computation is whatever is on the tape at that time. (Other halt conditions have been studied and proven equivalent to this one.) The procedure is taken to be undefined on the initial string if the computation never halts. The halt state of one Turing machine can be connected to the initial state of another, giving the possibility of interconnected machines, subroutines, etc., as discussed earlier in the knowing model, but the conceptual problems become difficult as we encounter subroutine class of machines, classes of subroutines, and so on. In the Universal Turing machine perspective, however, it is possible to represent not only single machines, but also subroutines, etc. by single programs, and to deal with the complexities of machine interconnections with flexible means of passing control among the programs. Essentially, we encounter all the power and flexibility of general programming languages.

The most interesting and important "negative" result in Turing machine theory is the existence of precisely definable, but effectively unsolvable problems. The paradigmatic unsolvable problem is called "the halting problem": it asks for an effective procedure for deciding if, given any particular Turing machine and any particular input string, the computation will ever halt. The assumption that such a procedure exists leads to a logical contradiction. The study of unsolvable or uncomputable problems is a major part of Turing machine theory, but the major effect of unsolvability for my purposes is to introduce the constraint that epistemological and knowing processes be computable—a process of knowing something is of restricted use if it can never arrive at a result. (This is a constraint that does not seem to have been considered among epistemologists.)

A Turing machine engages in knowing interactions with its tape symbol environment in the sense of the formal knowing model; its interactions are environmentally conditional, and its results can be distinguished. Knowing in this sense has many characteristics, and some of them have been especially emphasized in Turing machine theory. Perhaps the most intuitive characteristic of the Turing machine's processes is that it is computing a function on tape symbol strings, with the initial string as the argument and the final string as the value of the function. The results of such a process can be taken as being "about" the initial input to the process. Thus a machine that halts with a blank tape is said to have "recognized" the string it started with [again, there are many equivalent halt conditions; for a hierarchy of

string recognition capabilities, see Hopcroft and Ullman (15)]. Or the machine can compute the characteristic function of a set, where a result of "1" indicates that the initial string belongs to the set, and a result of "0" indicates that it doesn't belong. Or, finally, the machine can be taken to be computing predicates on the strings, with a result of "1" (say) indicating "true" and a result of "0" indicating "false." String recognition, characteristic function, and predicate computation are all variants of each other; for example, recognizing a string is identifying an element of a set, and computing a predicate is computing the characteristic function of its extension, and all are specializations of function computation. Actually, the reductions can be carried out in any direction: e.g., consider a function as a set of ordered pairs of argument strings and value strings, then computing that function is reducible to computing the characteristic function of that set of ordered pairs. All such computational properties are intrinsic characteristics of the general concept of knowing.

The possibility of uncomputability introduces a number of additional distinctions into these characteristics—in particular, there are functions, predicates, and characteristic functions that are not computable. Those that are called, simply, "computable." Another equivalent adjective is "recursive"; this derives from another attempted explication of effective procedure called "recursive function theory," and the equivalence of "recursive" to "computable" is a theorem, not a definition.

The essential meaning of "uncomputable" is that the computation may not necessarily ever halt, but there are various degrees of such recalcitrance. In particular, there is a class of "almost" computable procedures called "semicomputable" or "partial recursive." Intuitively, a semicomputable or partial recursive procedure is one whose computation will halt for some kinds of results, but not necessarily for others. Computable procedures, thus, are a subclass of semicomputable procedures. For example, a semicomputable function is a function whose values may or may not be defined on all strings, and whose computation will halt with the correct value when it is defined, but will compute in vain forever when it isn't. Similarly, a partially recursive set is one for which the computation of the "characteristic function" will halt and identify an element of the set, but may never halt for a nonelement; and the computation of a semicomputable predicate will halt if it's true, but not necessarily halt if it's false. Note that a value for which a semicomputable function is undefined may never be identified: it is distinguished by the computation never ending, but there's no way to determine that an ongoing computation will never end. It might halt

at the next step (recall the unsolvability of the halting problem)—similarly for nonelements of a partially recursive set and false instances of semicomputable predicates. Partially recursive sets have a special property from which their standard name is derived: the elements of such a set can be generated, or enumerated, one by one, by a recursive procedure—thus their standard name, "a recursively enumerable set." Such a generating or constructing possibility is another characteristic of knowing.

Semicomputable predicates also have a special characterization. In terms of formal logic, semicomputable predicates that are not computable turn out to be precisely those predicates that require exactly one existential quantifier. Thus, truth functional and sentential logic are recursive, but quantificational logic may only be partially recursive. From this perspective, partial recursiveness or semicomputability has an interesting intuitive interpretation: to establish an existential quantifier, the machine need only find an instance, but to falsify it, the machine has to search an infinite set, and that search may never end.

There are worse forms of noncomputability—functions, predicates, etc. whose computations are not guaranteed to halt under any circumstances—and the investigation of the properties and conceptual structures of noncomputability is a major focus of the theory of computation. The critical point for current purposes, however, is that noncomputability exists and must be taken into account. The identification of the general knowing model with Turing machines has thus encountered the general support of Turing's thesis and, through the existence of Universal Turing machines, the conceptual power of general programming concepts, but has at the same time encountered formal limits of computability, and thus, of knowability.

B. The Turing Machine Hierarchy

The discussion of knowing has been primarily in terms of the process of knowing, with little attention thus far to what can be known. Turing's thesis, however, and the "knowing" version of it, forces attention on what can be known by Turing machines, and its presumed equivalence to what can be known by human beings. It is in exploring this equivalence that we encounter a natural hierarchy of Turing machines.

In the general Turing machine model, the environment is presumed to be coded on the tape in the form of various symbol strings. This means that, in principle, rocks, chairs, automobiles, trees, etc. can all be adequately represented on the tape by appropriate strings, and the justification for such an assumption rests in a general mathematical-physical conception of reality.

Given such representations, a Turing machine can know rocks, trees, etc. within the limitations of the knowing model, and, in particular, can compute functions and predicates and recognize sets of them.

The complexity of the tape and of the tape head are directly related: the tape head must be presumed capable of reading and writing, of recognizing and constructing, whatever the elements of the tape "alphabet" are presumed to be. In a sense, the tape is the "unwelt" of the Turing machine determined by its tape head. Thus, in principle, the "tape head" can consist of whatever input-output (sensory-motor) processes we choose, (e.g., those of a human infant), and the "tape" or "environment" can become as directly intuitive as we wish. Such an intuitive modeling process, however, risks burying exactly the representational issues of importance in the assumptions about the "tape head" and the "reality" it encounters. In particular cases, the details of such representational codings are external to the Turing machine model itself, and become, in practice, engineering questions. For example, "How do we best hook up the master computer to the automated factory?" that is, "What is the best sensor and effector arrangement:" or, finally, "What is the best "tape head" design?" There is one special class of objects, however, for which the coding details are not external to, but intrinsic to, and thus constrained by, the Turing machine model itself: the class of Turing machines.

I have indicated how Turing machines can be coded on the tapes of appropriate Universal Turing machines in the sense that they can define and determine procedures and can transfer control among themselves. It is possible, however, that the fact of these programs being on the same tape as the "environment" is simply a notational convenience, and that they do not in fact constitute a part of the "environment" in the sense that they can have procedures executed *upon* them. It is possible, in other words, that Turing machines can determine knowing processes, but cannot themselves be objects of knowing due to constraints inherent in trying to represent them to themselves. The issue is critical because, clearly, human beings can know Turing machines, and, therefore, Turing machines must as well if their adequacy to human knowing is to be maintained.

The possibility is essentially that, in some sense, Turing machine representations are intrinsically "special": e.g., symbol strings requiring a special alphabet or special string sequencings, that other Turing machines can't operate on. The possibility is refuted in the general case by a process known as Gödel numbering. Gödel numbering is essentially an effective procedure for translating any symbol string in any alphabet, including any extended

alphabet or special symbol strings "needed" to represent Turing machines, into one very special "alphabet": the positive integers. Universal Turing machines can easily be defined with respect to this alphabet, and it follows that Turing machines (now "merely" integers "like anything else") can both execute procedures and be the objects of procedures: can both know and be known.

The thoroughness of the method, however, raises a subsidiary question—sort of the inverse of the first—can Turing machines recognize other Turing machines: i.e., can they distinguish programs (integers) from other symbol strings (integers)? Note that the question is critical only if it is assumed that human beings are capable in the general case of making such distinctions. It turns out, however, that the answer is yes—the recognition of programs is a computable problem (7).

At this point we have a general class, or first level, of Turing machines which operate on general symbol strings, and a second class, or second level, of Turing machines which operate on first level Turing machines. Similar questions about knowability can be asked about this second level as were asked about the first, and similar answers are available—inducing a third level. The general procedure is, in fact, indefinitely expendable, and we obtain an indefinite hierarchy of reflexively applicable Turing machines. [See the definitions and theorems concerning $K^{(N)}$ in Rogers (39). The issue of recognizability for higher levels is not computable.]

This Turing machine hierarchy, in which an element of the $N+1$st level operates on elements of the Nth level, is the central result of the Turing machine analysis of knowing. Its necessity as a hierarchy emerges from an analysis of the thesis that the knowing model is adequate to human knowing—in particular, from the necessity that Turing machines (knowing systems) can be potential objects of knowing to other Turing machines.

C. The Turing Machine Hierarchy and Formal Logic

The Turing machine hierarchy has emerged from an examination of the Turing machine "environment" and the requirement that it be potentially as inclusive as the human environment. Turing's thesis, however, would also claim that the Turing machine could know about that environment anything that a human being could. Such a claim encounters directly the impossibility of proving the thesis, for there is no formal representative of potential human knowledge available. The issue can be approximated, however, by posing a formal proxy for human thought, and examining the Turing

machine's relationship to that. In particular, I will consider the formal logic of predicate calculus and set theory as such a proxy.

A Turing machine represents a predicate in the sense that it determines a procedure for computing it—for deciding its truth or falsity for a given instance. Recall that truth functional and sentential logic are recursive. Thus all nonquantificational predicates are (algorithmically) computable. Quantifiers, however, introduce issues of noncomputability, and it might be asked how Turing machines are to represent uncomputable predicates. Uncomputability, however, does not imply that a corresponding program can't be defined, only that the computations of that program will or may never halt. The program (Turing machine) itself is finite as long as the predicate is.

Note that, considering Turing machines as representating predicates, a Turing machine which operates on Turing machines represents a predicate about predicates. Thus the Turing machine hierarchy corresponds to the other hierarchy of (nth order) predicate calculus. This gives a deep additional meaning to the Turing machine hierarchy, as well as to the sense in which a Turing machine represents a predicate, including an uncomputable predicate; it serves as the "object" of higher order predicates. (Essentially, we obtain uncomputability hierarchies at each level of the predicate hierarchy. Little is known about the general structure; only the first two levels have received any extensive attention.)

Turing machines represent sets in the sense of recognizing, or determining, their elements—again, even if uncomputably so. Taken as set representations, Turing machines which take other Turing machines as objects correspond to sets with other sets as elements. Thus, the reflexive relation in the Turing machine hierarchy corresponds to the elementhood relation among sets, and the hierarchy itself corresponds to the hierarchy of logical types.

The Turing machine hierarchy thus bears close correspondences to natural hierarchies in both predicate calculus and set theory, and the knowing model together with Turing's thesis suggests that these correspondences adequately model human logical capabilities.

D. Turing Machines and Human Thought

In arguing the adequacy of Turing machines to human thought, I have used formal logic as a proxy for human thought and explored the adequacy of Turing machines to that. The general proposition can never be proven, since no formal definition of human capabilities can be given. There are, nevertheless, still two objections I would like to anticipate.

The more fundamental of the two objections is against the adequacy of formal logic as a proxy for human thought: There are definite elements of natural language that have not been successfully formalized in logic, and this would seem to invalidate its adequacy as a proxy. My general response is to accept the objection, but to argue that in fact the Turing machine model is more powerful than formal logic, and, in particular, that it is able to cope with the characteristics that have proven resistant to standard logic.

I will illustrate the general approach with what are known as propositional attitudes. These are constructions of the form ". . . believes that . . .," ". . . wishes . . . ," and so on. The difficulty they pose for logic is that they violate the principle of extensionality—the principle that constructions with equal extensions (referents) are equivalent, and, in particular, substitutable for each other. For example, the sentence,

> Tom thinks that Tully wrote the *Ars Magna,*

may be true, and yet become false when "Cicero" is substituted for "Tully," even though Cicero = Tully (38).

I consider that such constructions require a distinction between intension and extension, and that the objects of such constructions properly include intensions. It seems to me, for example, that the above sentence is appropriately paraphrased as

> Tom thinks that the extension of the intension "Tully" is equal to the extension of the intension "wrote the *Ars Magna*."

Clearly, the substitution of Cicero for Tully would not necessarily maintain the truth value of the paraphrase unless Tom knows that Tully = Cicero extensionally. Obviously, this suggests a general insertion of propositions, attributes, etc. between names and their referents. Many constructions depend only on the extensions (e.g., "Cicero = Tully"), and the intensions may be dropped from consideration; it is precisely those constructions that do depend on intensions that have resisted logical formalization.

Such intensional paraphrasing requires that the concept of intension be given some referents, and I suggest that Turing machines (procedures) serve adequately. Thus, we can again paraphrase:

> Tom thinks that the satisfier of the procedure "Tully" is the satisfier of the procedure "wrote the *Ars Magna*."

If propositional attitudes take procedures as objects, then substituting one procedure for another may change the truth value of the sentence even if the satisfiers of the procedures (the extensions) are equivalent. In particular,

such a substitution will hold only if the subject of the propositional attitude (e.g., Tom) knows that the procedures have equivalent extensions, but this is not an algorithmically solvable problem and depends entirely on the details of Tom's past experience. Formal logic does not consider such issues of computability internal to its names and sentences. I suggest that the Turing machine model can.

The second objection to the adequacy of Turing machines to human thought that I want to consider is that the creativity of human thought is antithetical to the "sterile" determination of a Turing machine computation. This objection is more fundamental than the first in that it is directly against the Turing machine model of human thought, without reference to any intermediary proxies, and yet is less fundamental in that it rests essentially on a lack of familiarity with, or a lack of understanding of, the foundations of logic and mathematics.

Results of the last four decades have shown that the "classical" and common conception of logic and mathematics as closed and fixed systems is fundamentally in error. In particular, nonalgorithmic (heuristic) procedures are capable of indefinitely producing new and unanticipatable elements, structures, systems, etc., and such procedures appear to be as accessible to Turing machines as to human beings.

There are some relatively sophisticated—though, in my judgment, none the less in error—versions of the "sterility" arguments against Turing machines as sufficient models of human thought. For a review, see Turner (46). Such arguments proceed from Gödel's theorem: Gödel proved that in any consistent formal logic of sufficient complexity, there will exist an undecidable true theorem: that is, a theorem which cannot be proven either true or false within the formal logic, but which is nevertheless seen to be true from the perspective of a metalanguage for the system—a perspective which "looks at," "refers to," or "operates on" the given system as a semantic object (27). The arguments against Turing machines then generally equate the Turing machine to a formal logic and claim that the human being can find and prove the theorem (from Gödel) that is undecidable for the Turing machine—the human presumably has this advantage because he is able to consider the Turing machine system from a "metaperspective," as a semantic object.

This line of argument clearly assumes that a Turing machine is incapable of semantic ascent, and that assumption is clearly false. It may still be that a Turing machine cannot ascend the semantic hierarchy as powerfully or as completely as a human being—again we encounter the indeterminate and

indeterminable edges of Turing's thesis—but it is clear that a Turing machine can ascend to higher levels and can develop new systems of programs (logics) at those levels.

The last line of defense against the sufficiency of Turing machines appears to be one of claiming that a Turing machine that has ascended to a new semantic level is no longer the same Turing machine that it was before. The reasoning appears to be as follows: a Turing machine is a formal logic, so if it changes the logic, it's a different Turing machine. In the first place, this is internally contradictory: if a Turing machine were a formal logic, it couldn't ever change the logic—to be computationally as powerful as one or more logics is not to *be* those logics. Equating Turing machines to the provable theorems in a logic seems to involve the assumption that Turing machines are at best capable of partial recursive computations—again clearly false. Turing machines are as capable of heuristic computations as anything else. In the second place, it's semantically arbitrary: there is no more reason to deny continuity to a Turing machine that has just proven a theorem at or from a new semantic level than it is to deny continuity to a Turing machine that has just proven a theorem within an old semantic level, and, furthermore, if an arbitrary linguistic convention regarding continuity of identity is to be adopted with respect to some such kind of change, then consistency requires that the same convention be applied to human beings, and we will have discontinuous new human beings whenever, for example, someone understands Gödel's proof (see also the section concerning semantic ascent and stages of knowing). Linguistic conventions can, of course, be inconsistent, but they thereby lose philosophical and methodological interest.

Other objections to the model of knowing could be considered, but there seems to be little reason to believe that they would be any more successful than the many attempts to disprove Turing's thesis in its general procedural form. For current purposes, then, I will conclude that the knowing model is adequate to human knowing, and that a particular reflexive Turing machine hierarchy is a necessary structure in those knowing capabilities.

E. The Semantic Ascent Operator

I would like to explore one more question concerning the Turing machine model of knowing: Given the abstract existence of the Turing machine hierarchy developed in the preceding pages, could a particular Turing machine access or realize—at least in principle—that hierarchy in its sub-Turing machines?

A particular Turing machine must be finite, and therefore must contain only a finite number of sub-Turing machines. Thus, it must realize only a

finite number of levels of the Turing machine hierarchy. This gives rise to two subquestions: (*a*) Can a Turing machine construct (that is, can it have a procedure for constructing) new component Turing machines at higher levels in the hierarchy than any of its current components, and (*b*) given that it has a procedure for constructing Turing machines at higher levels, can that procedure "in principle" generate the entire class of Turing machines at these higher levels? The "in principle" is used as "without regard for issues of computability": It is clear that such a procedure will have to be heuristic—the levels will be neither recursive nor recursively enumerable.

Any procedure satisfying these two conditions will be called a *semantic ascent operator*. The rationale for this is that if the strings operated on by a procedure are considered to be the "referents" or "semantic objects" of the procedure, then a procedure with the above properties will "ascend" the semantic levels in the Turing machine hierarchy. [The term was originally motivated by Quine's (38) notion of semantic ascent, for example, though I'm not sure how much Quine would agree with the implied connection.]

Two further questions can now be asked: (*a*) Do semantic ascent operators exist, and (*b*) Can one (or more) of them be identified as operative in man?

It is easy to construct operators that move to a higher level. For a trivial example, given a procedure, construct a new one—at the next level—that "checks" procedures to see if they are identical to the given procedure. In other words, construct a procedure to compute the predicate "$x = A$" where "x" varies over procedures, "A" is a particular procedure, and "=" is a notational equality.

The condition that a semantic ascent operator access "all" of the higher levels, however, encounters directly some of the more difficult and controversial issues in the philosophy of logic and mathematics. The difficulty is not so much that there are no candidates for a semantic ascent operator [though not generally considered as procedures; for example, finitistic constructability, Gödel's constructability, definability (3,4)], but that there is no agreement on exactly what properties such an operator should have; for example, there is no agreement on exactly what mathematical entities are to be considered as existing, and therefore no agreement on what such entities should be accessible to such an operator. I will not pursue these issues—though the Turing machine perspective should have interesting things to say about them—but will stop with having followed them to the same frontiers as exist for human beings.

The question of whether or not some semantic ascent operator is identifiable as operative in man is clearly not answerable at this time. I ask it in order to point out some things: If the general hypothesis concerning the

equivalence of the Turing machine hierarchy with human cognitive capabilities is true, then an answer to this quest for a psychologically real semantic ascent operator would constitute an explication of the procedure(s) of human learning. This conclusion, in turn, would imply that human learning involves much deeper issues than are normally considered by learning theorists—e.g., issues of the computability of learning procedures, and issues of the range of access of learning procedures. Only recently has the probability begun to be taken seriously that there are different learning processes (procedures, heuristics) for different areas of learning (40). The epistemological consequences and constraints of learning are still left to philosophers.

F. Semantic Ascent and Stages of Knowing

If the initial knowing model is valid, then the Turing machine analysis shows that any knowing system will exhibit an intrinsic hierarchical structure in what it can know—in its knowledge. The simplest case, of course, and undoubtedly the most common, is a system whose knowledge is restricted to some portion of the first Turing machine level. Furthermore, the finiteness of any particular knowing system restricts its possible knowledge to a finite number of the levels of that hierarchy, and to a finite portion of each of those levels. Development of new knowledge thus consists of accessing more extensive portions of each level, and of ascending to new levels in the hierarchy. The possibility of ascending to new semantic levels of knowing introduces a stagelike structure into the development of knowledge, and it will be the intent of this section to examine some of the characteristics of that structure.

Most clearly, a stage structure constituted by ascents through the semantic Turing machine levels would exhibit a relationship of hierarchical inclusion between successive stages. In particular, the procedural elements of each stage are taken as objects by the procedures of the next stage.

One consequence of this structure is that the elements of the highest level attained by a particular knowing system cannot themselves be known by that system; they cannot be taken as objects of knowing until the system attains the next higher stage, until procedures at the next semantic level have been developed. Such highest level procedures can be manifested in the system's behavior; they can know lower level elements, but they cannot themselves be part of the system's known world. From the perspective of a particular system, such procedures do not exist. Correspondingly, they will be called *virtual* procedures, and those which can be taken as objects of

knowing will be called *real*. [This terminology is motivated by Quine's (37) distinction between virtual and real sets.] Development through the stage structure thus consists of making virtual procedures real and adding a new level of virtual procedures in their place.

Another perspective on this same relationship is to consider, not the highest semantic level procedure attained by a particular system, but rather the highest level procedure operative at a particular time. Such a highest level procedure will be taking other elements as objects and will be operating on, transforming, exploring the structure or implications of (in general) knowing them. If these objects happen to be other procedures, then the higher level procedure will in effect be taking these procedures as representatives or images of the elements and characteristics that they know. Correspondingly, procedures being taken as objects of knowing will be said to be functioning as *images*, and currently operative procedures will be said to be functioning as *plans*. Thus, all procedures can function as plans, but only real procedures can function as images.

Clearly these definitions are motivated by the Plan and Image of Miller, Galanter, and Pribram (24). This definition differs from theirs, however, in that they would seem to accept, for example, a subroutine return containing information about the current status of the environment as part of the image of that environment, while the above definition tends to accept only general time independent knowledge as part of an image. The distinction becomes a bit fuzzy, however, when we consider the possibility of a currently operative procedure, engaged in ongoing interaction with the environment, being simultaneously taken as an object of knowing by a higher level procedure.

Another major consequence of the general hierarchical structure of the stages is that development through them would necessarily realize an invariant order of their sequence. Very simply, no procedure can be operative without the lower order elements for it to operate on. Furthermore, with a constructive semantic ascent operator, no higher order procedure can exist in any sense without lower order procedures out of which it can be generated. Thus the semantic knowing levels generate a logically necessary ordinal scale of development for any knowing system.

Recall, in particular, that the knowing model was shown to be intrinsic to any biological system, and that it has been argued as adequate to human knowing. Thus we should, on the basis of these analyses alone, expect to find an invariant hierarchical stage sequence in the development of human knowing. (Note that the general argument applies both ontogenetically and phylogenetically.)

These characteristics are, of course, clearly and deliberately suggestive of the cognitive development stages empirically described and studied in the development of the child. The obvious next step is to further explore this relationship between stages as emergent from the nature of knowing and stages as empirically manifested in child development. In particular, I would argue that the formal model provides an adequate explication of the developmental model described and analyzed by Jean Piaget. Like any such task of explication, there is no determinate point of completion, and the argument must be halted by extrinsic considerations of time, space, and purpose. My current purpose is primarily to illustrate the plausibility of such an explication, and the discussion will correspondingly be limited to an exploration of the correspondence between the semantic level stage model and the major developmental stages of Piaget's model (32).

The general task is to determine the assignments of machine levels to cognitive stages, and this depends on the discovery of critical identifying characteristics of the *levels* in the empirical and theoretical definitions of the *stages*. There are essentially two characteristics of the machine levels that can be looked for: (*a*) A machine level is characterized semantically by the elements it can take as objects; and (*b*) a machine level is characterized computationally by having the potential capabilities of a Universal Turing machine. The two characteristics are not independent: clearly it would be possible to design a machine that contained procedures at a given semantic level, but did not have full computational potentialities at that level. The possibility of such a "partial" semantic level is thus an empirical issue for any particular system.

Universal Turing machine capability is thus the more general machine level characteristic. If it is accepted as an identifying characteristic, then it seems most likely that the preoperational stage appropriately corresponds to the first machine level. On the one hand, a preoperational child clearly has Universal Turing machine capabilities; some Universal Turing machines can be relatively simple systems [e.g., (25)] and would not seem to provide undue training difficulties for a conditioning psychologist and a preoperational child. On the other hand, a sensory motor child would not seem to be capable of Universal Turing machine capabilities. First, it is questionable if the sensory motor child has the necessary general learning capabilities (much of the learning is of specific kinds—e.g., eye-hand coordination, and might possibly be limited to specific neural systems that lack sufficient generality for a Universal Turing machine), and, even if the learning is granted, it is

further questionable whether the general perceptual and motor systems, the basic contacts with the environment, would be adequate. That is, they may be too labile and/or involve too small an alphabet. This is, of course, in principle an empirically testable issue. (Strictly, the learning and perceptual-motor questions are not independent: in the formal Turing machine model, a smaller alphabet generally corresponds to a larger Moore machine, and vice versa. The "size" of a Universal Turing machine is generally taken to be the number of symbols in the alphabet multiplied by the number of states in the Moore machine.)

If preoperational processes are identified with first machine level procedures, then the sensory motor stage presents some subsidiary special questions. It has already been suggested that sensory motor procedures are not equivalent to a Universal Turing machine, and this leaves two possibilities: sensory motor procedures are merely "partial" semantic level precursors of the "full" semantic level preoperational procedures—this implies that both "stages" are at the same semantic level and differ only in computational capabilities—or, preoperations are at a different semantic level from sensory motor procedures and can thus take sensory motor procedures as objects. The critical question, then, is whether or not sensory motor procedures can be objects for preoperational procedures.

There are many senses in which the sensory motor stage can be seen as the development of the basic input and output subroutines and servomechanisms for the latter stages. Eye-hand coordination, locomotion, object indexing, etc. constitute the basic foundational procedures through which (not "upon which") preoperations interact with the environment. Such subroutine relationships do not constitute a difference in semantic levels. If this is the extent of the relationship between sensory motor and preoperational procedures, then both are at the same semantic level, and sensory-motor developments constitute a "partial" level precursor to preoperations—analogous to the development of a complex interactive tape head in the formal model. Note that the relationship of "taking as object" from one semantic level to another does not preclude "transferring control to" among subroutine procedures across the differing levels. Thus, it is clear that sensory motor procedures are subroutines for preoperations; the critical question is if they *can also* be objects.

The primary indication that sensory motor procedures might be taken as objects by preoperational procedures is the existence of images in preoperations. Images might seem necessarily to involve some procedures being taken

as objects of knowing by others—some procedures functioning "as image" with respect to others. There is a critical sense in which even virtual procedures can constitute implicit images, however, and it is still possible that preoperational images are of this kind.

Any procedure represents the environment with which it interacts in the sense that it successfully "guides" or determines those interactions. Images would seem to be a special kind of representation that carry structural and contextual information about that environment, presumably derived from past experience. Yet this is precisely the information contained in *any* "integrated" procedure (e.g., a servomechanism) whether virtual or real. Such "contextual" information simply means that the procedure is able to handle many, or all, possible interaction sequences in its environment.

The critical question is functional: virtual procedures can only represent contextual information "for the sake of" or "in the guidance of" ongoing interactions; they cannot be used to deduce or anticipate the consequences of behaviors except in the execution of those behaviors. Such a characteristic of a virtual procedure will be called the characteristic of being a *virtual image*. Furthermore, a virtual image is a representation of its environment, and a set of rules for interaction with that environment; it is *not* a representation of *interactions with* that environment.

However, a real procedure, that is a procedure that can be taken as image by a higher level procedure, can correspondingly constitute a *real image* to that higher level procedure. In particular, the plan procedure can simulate the interactions of the real image in order to anticipate new consequences, or it can represent particular interaction sequences of the real image, thus representing behaviors within or transformations upon the corresponding environment.

If we now return to the issue of images in preoperations, we find that they are precisely the static nonanticipatory kind that would be expected of virtual images (33). Furthermore, anticipatory and transformational images, characteristics of real images, begin with concrete operations—which presumably corresponds to the development of a second machine level. Thus, preoperational images would not seem to entail a difference in semantic level from sensory-motor procedures to preoperations.

The sensory-motor stage can therefore be identified with a "zero" (or fractional, partial) semantic or machine level, and preoperations with the first semantic level. The objects of these procedures would be the perceptual complexes identified by sensory motor procedures, and the first level proce-

dures, would compute various functions and predicates on these complexes. In particular, certain servomechanism procedures, involving rotational and displacement transformations, would represent physical objects—the permanency of the object is implicit in the ability of the servomechanism to produce as a goal any of the perceptual "versions" of the object from any of its other versions, including hidden ones. However, this is a virtual representation during preoperations: the real representations are of the perceptual arrays.

With concrete operations, and the second machine level, these preoperational representations can become real. Thus, for example, physical objects can be taken as explicit entities, and various characteristics of the objects, as differentiated from their perceptual presentations, can in principle now be computed (for example, amount of substance, weight, or volume). An illustrative example of the effect of the second machine level involves the transitivity of an order relation. A preoperational child can learn to compute an order relation—e.g., match two sticks and pick the longer. The corresponding procedure thus represents that order relation, but only virtually. Thus the relation can be computed in any given case, but no characteristics of the relation itself can be deduced: that would involve taking the procedure as an object and computing predicates of it. With the higher machine level, however, the order procedure can be a real representation, and various predicates (e.g., transitivity) computed on it, perhaps using anticipatory images.

Thus concrete operational procedures can compute functions and predicates on objects. A predicate applied to an object constitutes a sentence, or proposition, about that object, and it is in the form of propositions that concrete operations are usually considered when taken as objects for third level, or formal operational, procedures. In particular, formal operational procedures can generate new, and perhaps hypothetical, propositions, can represent structures of propositions, and can compute deductive chains of propositions.

The first, second, and third semantic levels can be identified with preoperational, concrete operational, and formal operational thought, respectively, and these identifications seem at least consonant with the major features of those stages. The sensory-motor stage has a different status in this identification, being essentially a partial first semantic level. These identifications constitute the basic proposed correspondence between the formal stages and the Piaget stages. [There are an indefinite number of

formal semantic levels, and only three have been accounted for. This suggests the possibility of post formal operational stages, and Powell (34) has evidence that just such a stage may be found in a small percentage of the population.]

It was suggested earlier that the formal model can account for the hierarchical relations, the invariant sequence, and the biological emergence of the stages. There is one characteristic, however, for which the formal model suggests a difference from the Piaget model, or at least a difference in emphasis. This difference concerns the qualitative nature of the stages.

The Piaget stages are often identified with the mathematical structures that are manifested within them; especially, concrete operations is the stage of groupings, and formal operations is the stage of groups. It is not always clear just what is being emphasized here—the mathematical structures *per se*, or the objects out of which they are constructed—but the general emphasis seems to be on the structures. As if, for example, the abstract structure of a grouping were definitive of concrete operations. The formal mode suggests exactly the opposite emphasis; the stages are defined by what can be taken as objects of the structures, not by the structures themselves.

In order for this to be a nonvacuous distinction, it must be claimed that the objects are specific to particular stages—that much has been argued in the development of the formal-to-Piaget correspondence—and that the abstract structures are not similarly specific. This latter point is actually quite easy to make: for example, we find a group of displacements on arm position in the sensory motor stage, a group of locomotor displacements on body position in preoperations, a group of addition on the integers in concrete operations, and a group of logical operations on propositions in formal operations. In each case, the abstract structure is a group, but the elements involved are qualitatively very different. Piaget has posited differing structures for the same stage (e.g., groupings and groups in concrete operations, of groups and lattices in formal operations) and the same structure in differing stages (e.g., groups in all stages), but it does not seem to have always been explicitly realized that this implies that the structures *per se* cannot be the definitive characteristics—it must be the elements. The formal model suggests that it is the semantic level of the elements. This is in contrast, for example, to Flavell's (10) discussion of the "integrated character of stages." What does the "integrating" are the structures, not the elements. Strictly, particular structures with particular elements are usually specified, (e.g., the IRNC group on propositions), but the emphasis is still on the structures.

The discussion has focused on the basic stage structure and characteristics of the Piagetian model; clearly the major task of applying the formal model to cognitive development remains. The apparent success of this partial application suggests that such a general application might be both feasible and desirable. Note especially that the formal model has been derived independently of the general cognitive and developmental data, and thus offers an independent source of hypotheses and conceptual distinctions. In particular, if the knowing model is valid, then the formal stage model *must* hold in some form or another, even if not identically to already described stages, and, furthermore, it serves as an explanation of the characteristics and relationships that can be derived from it. Conversely, success of the model in explicating and suggesting empirical results constitutes empirical support for the validity of the knowing model.

V. THE POSSIBILITY AND PLAUSIBILITY OF CONSCIOUSNESS AND SEMANTIC ASCENT

There is an apparent contradiction in the fact that the semantic stage model contains an indefinite number of levels, while the conscious knowing model contains only two levels. To be sure, the semantic levels are permissive rather than prescriptive—there is nothing in the model to prevent a system from realizing a small number of levels—yet I have already identified three levels as operative in man in correspondence with the cognitive stages. How is the existence of more than two levels to be explained?

A preliminary response to the question consists in pointing out that in the Turing machine model the indefinite number of semantic levels were developed in the context of only one machine—a Universal Turing machine. That is, the semantic levels are logical in character, not physical, and even a single physical machine can suffice to the semantic hierarchy as long as it has Universal Turing machine capabilities.

But this only reconstructs the anomaly in the opposite direction: if one physical machine level is sufficient, why posit two of them? Or, if it is accepted that two do in fact exist in man, why should the second, and presumably superfluous, machine have evolved?

The answer to this is more involved. Consider a Universal Turing machine operating at only one semantic level. Such a machine can in principle have programs of other Turing machines on its tape, and can ascend semantic levels by constructing further programs that operate on the original ones. The first step, however, is the existence of Turing machine codes on the tape—that is, in the environment—and preliminary to that is the development of a language within which such codes can be written. Turing proved that machines are definable for which such programming languages are possible—they are now called Universal Turing machines—but this says nothing at all about how such a language might develop or evolve in a natural setting. I will argue essentially that such a language is unlikely to evolve except in machines (systems) that already have two physically realized semantic levels. Once evolved, however, the language would be available for ascents to higher semantic levels without corresponding physical levels.

In other words, the ascent to the second semantic level of a Universal Turing machine is logically possible either through a second physical machine level or by means of programs in an appropriate external language. In the development of the consciousness model, I argued the plausibility of

the biological evolution of a second physical machine level. I will now argue (*a*) the implausibility of the evolution of an appropriate language in (a species of) a one layer machine, (*b*) the plausibility of the evolution of such a language in a two layer machine, and (*c*) the likelihood that further semantic ascents would be in terms of the language rather than the physical machine, once the language was available (thus tending to limit the physical evolution to two levels). The critical issue, then, is what bearing the number of machine levels has on the evolution of a language.

A program on a tape—in the environment—can, as a program, be taken in only two possible senses: either as an initiator and controller of ongoing interactions, or as a static or "timeless" definition of a procedure (that is, as a plan to be executed or as an image to be examined). I have shown that this second sense is available only with the existence of higher level procedures. But this means that the programs for a single layer machine must always be enacted whenever the machine encounters them; they cannot be taken as real images, only as plans (virtual images).

The Universal Turing machine manages to avoid the potential difficulties of this restriction by carefully controlling when it is "ready" to respond to a program, and by carefully controlling where the tape head is on the tape when it enters such a condition of readiness. (A Turing machine interaction generally involves a lot of spacing to appropriate positions on the tape.) Thus a Universal Turing machine does always respond to its programs whenever it encounters them, but, through the restriction of the tape head to a single tape mark at a time, and with sometimes elaborate "bookkeeping" about where it's at, the Universal Turing machine manages to encounter programs as programs only when appropriate.

Thus, in a natural or ecological setting, the evolution of an adequate programming language among single layer machines involves the evolution of a language that is always "responded to" (enacted), whenever encountered, and either the simultaneous or the delayed evolution of an appropriate apparatus to restrict those encounters. Such an apparatus is not particularly trivial, and it seems clear that it would have to be designed, not evolved. It is perhaps conceivable that it could be designed with the aid of the language, once it had evolved, but I will argue that no such language is likely to evolve.

The primary restriction of such a language is that it must always be responded to—the programs it names or defines must always be enacted. This means that such a language is only appropriate to situations and conditions in the immediate environment. Anything outside the immediate envi-

ronment is not available to be interacted with, and thus any language usage relevant to such nonimmediate situations would induce inappropriate behavior in any systems (organisms) receiving the language. It is plausible that a few signals of fear, anger, pleasure, and perhaps a few social interaction signals might evolve under such constraints, but the evolution of much more than that, and, in particular, the evolution of a language adequate to general programming needs seems extremely unlikely.

Thus the evolution of an adequate programming language is not to be expected among single machine level systems, and any evolution of a second semantic level would be expected to occur in terms of physical machine levels—as in the consciousness model.

With two semantic levels, however, a language element can be taken either as invoking an environmental interaction, or as naming or defining or evoking a general procedure. In the latter sense, the language element can be construed as invoking an internal interaction of the knowing (perhaps the locating or constructing) of the procedure defined, a process that requires at least two machine levels: one for the knowing procedure and one for the known (defined) procedure. Such language elements can be used to identify and define procedures that are relevant to situations and conditions external to the immediate environment—without inducing inappropriate immediate behaviors. Furthermore, to the extent that such two level language constructions can be stored (since they don't have to be enacted or executed) in memory or cultural transmission or in writing, they can serve for recording and storing general knowledge, and for the indirect transmission of knowledge from one system to another.

I take this functional distinction to be the most useful distinction between signs and symbols. Thus any element invoking only a virtual procedure could only be a sign, while a symbol could invoke either a real procedure or a higher level—perhaps virtual—procedure that would "know" the corresponding real procedure. Symbols, then, could only exist for a system with at least one real level of procedures (that is, with at least two semantic levels for a concrete operational but not a preoperational child). The above argument thus claims that a language with strictly sign functions is not likely to evolve to any complexity, while a language with symbolic functions as well could evolve to a general programming level. In this perspective, the intension of a symbol could be construed as the real procedure to which it corresponds, and its extension to the extension or domain of that procedures. The same distinction could be made for a sign, but only for a symbol can the system itself make such a distinction. Such a usage of "sign" and "symbol" is different

from Piaget's (30) usage in two major senses, but I do not have the space to analyze and argue those differences at this time.

The usefulness of such a symbolic language seems clear, and the plausibility of its evolution, once possible (i.e., once two machine levels exist) seems equally so. The probable setting for its evolution would be social interaction, and a potential path of evolution would involve the social coordination of activities relevant to increasingly removed situations. The need for such communication among social predators has often been noted. As such a language became adequate to the coordination of arbitrary activities, it would thereby become adequate, at least in principle, to general programming functions.

Furthermore, the existence of such a language would allow cultural evolution to supplement biological evolution in the ascent to new semantic levels, and the speed of cultural relative to biological evolution would result in, in effect, a supplanting of biological evolution by cultural evolution with respect to semantic ascent above the second semantic level. This analysis would therefore lead to the expectation that cognitive operations (second semantic level) are biological in origin and are perhaps in part dependent on neural maturation for their timing, and that formal operations (third semantic level) are historical in origin and are dependent primarily on cultural experience and training for their timing.

The resolution of the "contradiction" between an indefinite number of semantic levels in the formal cognitive model and only two machine levels in the consciousness model thus rests upon (*a*) a distinction between logical semantic levels and physical machine levels, (*b*) a second look at the function of programming languages in semantic ascent, and (*c*) an analysis of the effects of the number of machine levels on the probability of language evolution. Together with the preceding section, in fact, this discussion has shown conscious knowing to be an integral part of cognitive development: the first operation of a second machine level is taken to be definitive of concrete operations, the general reflexive relationship of consciousness is taken to be identical with the relationship between adjacent stages, and reflexive consciousness is found to be essential to language processes allowing ascent to higher stages. Thus, two separate developments of the basic knowing model are shown to have an essential integration.

VI. CONCLUSIONS: SYSTEMS PSYCHOLOGY

Beginning with a basic model of knowing, I have demonstrated that any knowing system is constrained by an indefinitely extending hierarchical structure of potential objects of knowing, and have argued that this structure is sufficient to account for the cognitive developmental stages. Thus, I have argued that the cognitive stages are emergent from the essential logical character of knowing.

Knowing was shown to be a characteristic in at least a trivial form of any living system, and a path of potential evolution beginning with a basic knowing system was outlined. This path was later found to constitute the biological evolution of the second level of the logical knowing hierarchy, and it was argued that higher levels of the hierarchy could be attained in a strictly formal sense once this basic two level system structure had evolved. That is, the path of general evolution contains the knowing hierarchy as one of its dimensions, and it can be expected that the first two levels of the hierarchy will be attained in a physical sense, and higher levels in a formal sense.

Thus, I have demonstrated the logical existence of a hierarchical constraint on knowing, the plausibility of the physical existence of systems actually manifesting those constraints, and the potentiality for identifying those systems and constraints in the human cognitive developmental stages.

It is conceivable that the model of knowing that I have proposed is weaker than human knowing, that it is subject to limitations that do not hold for humans, and therefore, perhaps, that the hierarchical constraints on the knowing model do not apply to human beings. This would, of course, invalidate the constraints as an explanation of the cognitive stages. Demonstrating this, however, would involve disproving Turing's thesis in its general form, as well as the more specific arguments I have given for the adequacy of the knowing model to human knowing. This does not seem likely.

In general, the internal validity of the argument seems to me to be difficult to question, including its applicability to human beings. Details of the particular correspondence proposed between semantic levels and Piagetian cognitive stages may be in error—these details are not derived from the formal knowing model, but constitute independent hypotheses in their own right (for example, the conclusion that the sensory motor period does not constitute a full and independent semantic level could be invalidated by additional data)—but the general existence of such a correspondence seems assured.

(Strictly, the applicability of the semantic hierarchy to human knowing seems assured. It is conceivable that this constraint is somehow unrelated to the cognitive stages, but this seems unlikely in the extreme.) This correspondence with the Piagetian model would seem to constitute a confirmation of the primary conclusion of the argument—that is, the cognitive stages would seem to instantiate the formal knowing hierarchy. Conversely, the correspondence of the results of the logical and biological considerations of this discussion with the Piaget stages would seem to provide support for the Piagetian model.

Clearly, this application to cognitive development does not exhaust the potential scope of the model. First of all, even the cognitive application is given only preliminary attention—the correspondence to the cognitive stages would seem to be a nontrivial application, but, nevertheless, there are clearly vast areas of Piaget's model in particular and cognitive development and cognition in general that remain to be considered from the perspective of the knowing model—and, in addition, the identifications with learning, emotions, and consciousness remain to be explored and defended. Relative to the potential scope of the model, the application to the Piaget stages is a narrow base of confirmation: confidence in the applicability of the general model to psychology will be enhanced by the development of further applications, by the broadening of that base.

The structure of the model as it has been presented begins with knowing as a primary model, considers biological and logical characteristics of that model, then integrates these two characteristics and applies the results to the Piaget stages. From a broader perspective, however, the basic model is constituted by the learning, emotion, and consciousness models—a model of general psychological processes. Knowing then becomes a characteristic of this basic model through which it can be applied to cognition and cognitive development, and biological evolution becomes a dimension of coherence internal to the model. From this perspective, the presentation of the model is very incomplete: the missing explorations of learning, emotions, and consciousness become, not simply supportive, but critical to the foundations of the model. In spite of its incomplete presentation, however, this would seem to be a more fundamental way of viewing the model—an explication of primary psychological processes which can then be used as a foundation for the analysis of higher order processes. From this perspective, the potential scope of the model is greatly enlarged: the characteristics of motivation and cognition become directly relevant to the basic model, as well as the issues of physiological realizability and phenomenological plausibility, and the range

of potential applications of that model should cover much of psychology, including ultimately personality and social psychological processes and their pathologies.

The model as presented, in fact, is intended as much as an illustration of a method as a presentation of specific results (11). Cybernetic systems theory contains languages and results that are powerful enough in principle to explicate any process whatever, including psychological ones, without suffering the limitations of any particular data language, methodology, or analogy, and thus offers the opportunity of the construction of psychological explications that can mediate among all such languages, methodologies, and analogies—in particular, among the basic physiological, phenomenological, and behavioral epistemological perspectives in psychology. That is, cybernetic languages can in principle explicate any physical process, including neurophysiological ones, can manifest any possible input-output relationship, including behavioral ones, and offer their own epistemological principles as explications of phenomenological ones: a single model, then, could integrate all three fundamental approaches to psychology. Furthermore, formal considerations can introduce constraints that are independent of any particular empirical observations and that can yield their own structurings, explanations, and predictions of empirical results, as when a formal analysis of knowing yields a hierarchical constraint on what can be known.

Mathematics is more than a language of measurement; it is a language of structure and process as well, and it seems advisable to make maximal use of its potential. Formal systems languages offer a rigorous conceptual framework for the analysis of control and information processes, and, to the extent that these encompass psychological processes, offer the possibility of an integrated systems psychology. That possibility remains largely unexplored.

REFERENCES

1. ASHBY, W. R. Design for a Brain. London: Chapman & Hall, 1960.
2. AUSTIN, J. L. How To Do Things with Words. New York: Oxford Univ. Press, 1962.
3. BENECERRAF, P., & PUTNAM, H., *Eds.* Philosophy of Mathematics. Englewood Cliffs, N.J.: Prentice-Hall, 1964.
4. BETH, E. W. The Foundations of Mathematics. Amsterdam, Netherlands: North-Holland, 1959, 1968.
5. BETH, E. W., & PIAGET, J. Mathematical Epistemology and Psychology. Dordrecht, Holland: Reidel, 1966.
6. BRUNER, J. S., OLVER, R. R., & GREENFIELD, P. M., *et al.* Studies in Cognitive Growth. New York: Wiley, 1966.
7. DAVIS, M. Computability and Unsolvability. New York: McGraw-Hill, 1958.
8. DOBZHANSKY, T. Genetics of the Evolutionary Process. New York: Columbia Univ. Press, 1970.
9. FEIGENBAUM, E. A., & FELDMAN, J., *Eds.* Computers and Thought. New York: McGraw-Hill, 1963.
10. FLAVELL, J. H. The Developmental Psychology of Jean Piaget. Princeton, N.J.: Van Nostrand, 1963.
11. FODOR, J. A. Psychological Explanation. New York: Random House, 1968.
12. GINZBURG, A. Algebraic Theory of Automata. New York: Academic Press, 1968.
13. GROSSMAN, S. P. A Textbook of Physiological Psychology. New York: Wiley, 1967.
14. HALLE, M., & STEVENS, K. N. Speech recognition: A model and a program for research. In *The Structure of Language,* J. A. Fodor & J. J. Katz, *Eds.* Englewood Cliffs, N.J.: Prentice-Hall, 1964.
15. HOPCROFT, J. E., & ULLMAN, J. D. Formal Languages and Their Relation to Automata. Reading, Mass.: Addison-Wesley, 1969.
16. IWANAGA, M. Development of interpersonal play structure in three, four, and five year-old children. *J. Res. & Devel. in Educ.*, 1973, **VI**(No. 3), 71-82.
17. JOHN, E. R. Mechanisms of Memory. New York: Academic Press, 1967.
18. MACKAY, D. Mindlike behavior in artefacts. In *The Modeling of Mind,* K. M. Sayre & F. J. Crosson, *Eds.* New York: Simon & Schuster, 1963.
19. ———. Information, Mechanism and Meaning. Cambridge, Mass.: M.I.T., 1969.
20. MARLER, P., & HAMILTON, W. J. Mechanisms of Animal Behavior. New York: Wiley, 1966.
21. MARVIN, R. Attachment, Exploratory and Communicative Behavior of Two-, Three-, and Four-Year-Old Children. Unpublished Ph.D. dissertation, University of Chicago, Chicago, Illinois, 1972.
22. MAYR, E. Populations, Species, and Evolution. Cambridge, Mass.: Harvard Univ. Press, 1970.
23. MEAD, G. H. On Social Psychology. Chicago: Univ. Chicago Press, 1934, 1964.
24. MILLER, G. A., GALANTER, E., & PRIBAM, K. H. Plans and the Structure of Behavior. New York: Holt, Rinehart & Winston, 1960.
25. MINSKY, M. Computation. Englewood Cliffs, N.J.: Prentice-Hall, 1967.
26. MINSKY, M., & PAPERT, S. Perceptrons. Cambridge, Mass.: M.I.T., 1969.
27. NAGEL, E., & NEWMAN, J. R. Gödel's Proof. New York: New York Univ. Press, 1958.
28. NEISSER, V. Cognitive Psychology. New York: Appleton-Century-Crofts, 1967.

29. NEWELL, A., & SIMON, H. A. Human Problem Solving. Englewood Cliffs, N.J.: Prentice-Hall, 1972.
30. PIAGET, J. Play, Dreams and Imitation in Childhood. New York: Norton, 1962.
31. ———. Biology and Knowledge. Chicago: Univ. Chicago Press, 1971.
32. PIAGET, J., & INHELDER, B. The Psychology of the Child. New York: Basic, 1969.
33. ———. Mental Imagery in the Child. New York: Basic, 1971.
34. POWELL, P. Higher-Level Cognitive and Social Competence Stages: An Extension and Integration of Piaget's and Mead-Turner's Theories. Unpublished Ph.D. dissertation, University of Chicago, Chicago, Illinois, 1971.
35. PRIBRAM, K. H. Languages of the Brain. Englewood Cliffs, N.J.: Prentice-Hall, 1971.
36. QUINE, W. V. O. Word and Object. Cambridge, Mass.: M.I.T., 1960.
37. ———. Set Theory and Its Logic. Cambridge, Mass.: Harvard Univ. Press, 1963.
38. ———. Philosophy of Logic. Englewood Cliffs, N.J.: Prentice-Hall, 1970.
39. ROGERS, H., JR. Theory of Recursive Functions and Effective Computability. New York: McGraw-Hill, 1967.
40. SELIGMAN, M., & HAGER, J. L., *Eds.* Biological Boundaries of Learning. New York: Appleton-Century-Croifts, 1972.
41. SIMPSON, G. G. The Meaning of Evolution. New Haven, Conn.: Yale Univ. Press, 1949, 1967.
42. SIMPSON, G. G. The Major Features of Evolution. New York: Columbia Univ. Press, 1953.
43. SLAGLE, J. R. Artificial Intelligence. New York: McGraw-Hill, 1971.
44. SLOMAN, A. Functions and rogators. In *Formal Systems and Recursive Functions*, J. N. Crossley & M. A. E. Dummett, *Eds.* Amsterdam: North-Holland, 1965.
45. TAYLOR, J. G. The Behavioral Basis of Perception. New Haven, Conn.: Yale Univ. Press, 1962.
46. TURNER, M. B. Realism and the Explanation of Behavior. New York: Appleton-Century-Crofts, 1971.