COGNITIVE PSYCHOLOGY 1, 358–387 (1970)

An Information Processing Analysis of Some Piagetian Experimental Tasks[1]

DAVID KLAHR

Carnegie-Mellon University, Pittsburgh, Pennsylvania

AND

J. G. WALLACE

University of Warwick, Coventry, England

One of the fundamental problems in the study of cognitive development is the determination of the developmental relationship between particular cognitive skills. An indication of its complexity is provided by the research literature centered on Piaget's concept of a "stage" in intellectual development. The first steps in an attempt to tackle this issue from a new methodological orientation based upon an information processing analysis are presented. A simple model of the problem-solving processes which appear to underlie successful performance on a range of experimental tasks testing concrete operations is described. It is analogous to a computer assembly system, with the major processes consisting of encoding of external stimuli, assembly of a task-specific routine from a repertoire of fundamental processes, and execution of the task-specific routines. The postulated fundamental processes are described in detail, and task-specific routines sufficient to perform successfully are described in terms of the fundamental processes. It is argued that since the complex task-specific routines consist of components that contain motivational and attentional mechanisms, no simple ordering of concrete operational tasks can be expected. Some new directions for experimental work are suggested.

One of the fundamental problems in the study of cognitive development is the determination of the developmental relationship between particular cognitive skills. In the customary experimental paradigm, children are presented with a set of test items that are intended to tap the cognitive skills under investigation. The developmental sequence for any pair of items, A and B, can be determined by examining the fourfold table of passes and failures for the two items. For instance, if there are no occurrences in the pass A–fail B cell, then there exists strong evidence that the cognitive processes sufficient to pass item B are necessary to pass item A.

[1] This research was supported in part by a grant to the second author from the British Social Science Research Council. We are indebted to John H. Flavell and Walter Reitman for their constructive comments and careful cricitism of the manuscript.

Or, if the empty cells are pass A–fail B *and* fail A–pass B then there is evidence that both skills appear simultaneously. The complexity of the questions raised by this problem is clearly illustrated by the research literature centered on the concept of "stage" in Piaget's theory of intellectual development. A comprehensive review of this work has been recently provided by Pinard and Laurendeau (1969). One of the most controversial issues which they discuss stems from Piaget's (1941) contention that all of the groupings underlying the stage of concrete operations "appear at the same time without our being able to seriate (them) into stages" (p. 246). A review of the studies aimed at evaluating this assertion (Pinard & Laurendeau, 1969) provides an inconsistent picture: some of the results support synchronism while others reveal asynchronism.

It is extremely difficult to design experiments to resolve this issue, as the following brief consideration of the methodological difficulties encountered by Smedslund (1964) will indicate. Smedslund attempted to determine the nature of the interrelations within a set of test items regarded as tapping specific aspects of Piaget's level of concrete reasoning. Concrete reasoning is assumed to be reflected in certain types of inference patterns, and, in Smedslund's view, the unitary nature of the construct requires that these patterns should be exactly related. The results, however, revealed that not one of the fourfold tables covering the pass-fail relations between pairs of items contained an empty cell, and, thus, none of the pairs of items exhibited an exact relationship. In addition none of the items were free from inconsistent subitem responses. In searching for an explanation of the wide variations in children's performance on the tasks, Smedslund hypothesized that the inconsistency and absence of exact relations might be due to the fact that the items could not be meaningfully compared since they differed not only in the inference pattern involved but also in the nature of the stimulus situations presented to the subjects and in the goal objects which they were instructed to attain. Exact relations between inference patterns might be discovered if goal objects and stimulus situations were held rigidly constant. Smedslund (1966a, b) attempted to do this in a further series of experiments. Due to the extreme difficulty of retaining identical stimulus situations while varying the inference pattern under consideration, the focus of the work was a comparatively narrow task area involving comparisons of children's ability to determine the effect of various combinations of addition and subtraction of one unit on the relative amount in two unseen collections which the subjects were informed were equal at the outset. Contrary to Smedslund's expectations, even with these narrow tasks, exact relations between items proved to be nonexistent. The results demonstrated that the same logical task structure, with identical perceptual and conceptual contents, may yield radically different solution frequencies depending, for example, merely on position in the series of tasks. The lesson to be learned from these findings, Smedslund believes, is that tasks must be analyzed in much more detail than is provided by a description of their conventional logical structure. The general problem is to determine exactly how the input is encoded by the subject and what transformations occur between encoding and decoding. The objective task structure alone does not yield a valid description of the solution performance, and it is necessary to diagnose the actual psychological processes in great detail to obtain minute descriptions or well supported inferences about the actual sequence and content of the thinking processes. The way in which the work of diagnosis, description, or inference is to be carried out remains, at present, an unsolved methodological problem.

A second basic issue posing intractable methodological problems is the question of the nature of the transition rule in cognitive development, i.e., the mechanisms or processes which govern the child's movement from state to state through the developmental sequence. The difficulties which characterize research on this theme are exemplified in the experimental studies aimed at determining the relative importance of the factors invoked by learning theories and Piaget's concept of "equilibration" in accounting for the process of transition. As Laurendeau and Pinard (1962) have pointed out, an overview of the results of these studies leads to pessimism about the outcome of an experimental approach to the transition problem. The existing experimental results appear to be compatible both with Piaget's position and with that of the protagonists of learning theory, and a solution is still awaited to the methodological problem of devising a series of critical experiments.

An important underlying cause of this situation appears to be the level of generality at which the theoretical accounts of transition are presented. In the case of Piaget, for example, the focus on change is, in general, imperfectly developed in his formulation of intellectual development since, as Wohlwill (1966) has indicated, for all its formal elaboration and complexity his system remains at base a structural analysis of children's performance in cognitive tasks at different levels of their development. His treatment of the functional side of the problem, the nature of the processes by which these changes take place, is much less complete. With the solitary exception of an account of the appearance of conservation of continuous quantity couched in terms of the typical test situation, Piaget's (1957, 1960) descriptions of the functioning of equilibration are highly general statements which do not deal with the particular mechanisms governing developmental changes or specify the conditions under which they take place.

The practical effects of this lack of an account of equilibration in specific process terms can be seen in the confusion surrounding the numerous experimental attempts to accelerate the appearance of the various conservations (Sigel & Hooper, 1968). These studies feature a variety of competing acceleration treatments all of which are regarded by their authors as being based on Piaget's description of the underlying processes concerned. The absence of a precise process-performance link contributes to the extreme difficulty of putting Piaget's account of transition to an experimental test, since the lack of specificity makes it all too easy to argue that a wide range of experimental results are compatible with his position. Although a detailed argument in support of the assertion cannot be offered in the present paper, criticism of the level of generality of presentation and of the existence of a gap between the levels of process and performance appears to be equally applicable to the theoretical accounts of transition offered from the standpoint of learning theory.

The purpose of this paper is to report the first steps in an attempt to interpret the import of interitem associations and to explain transition in cognitive development from an information processing orientation, based on the well-known work of Newell and Simon (1963). The initial focus is upon a specific set of related tasks whose successful completion is considered to be indicative of a certain stage in cognitive development. Thus, although the major relevance of the work is to the first of the two problems, it also has methodological implications for the transition rule issue. The decision to concentrate initially on a state description of one point in intellectual development before tackling the problem of the transition rule determining the passage from state to state in development is consistent with the line of approach advocated by Simon (1963) and Flavell and Wohlwill (1969). It is, also, consistent with the approach adopted by the writers in studying the very specific skill of series completion (Klahr & Wallace, 1970). In terms of delineating the requirements for sufficiency, the criteria which a transition mechanism must fulfill are likely to be much clearer if a sufficient state description is already in existence before the question of self-modification of structure is addressed.

The specific set of seven classification tasks considered here, selected from a set of eleven tasks employed in a single study by Kofsky (1966) of the performance of 4- to 9-year-old children, belong to those which in Piaget's view require functioning at the level of concrete operations for successful performance. They were chosen because of the complex structural interrelationships which they exhibit and because they constitute a representative subset of the wide range of concrete operational tasks. Kofsky (1966) has provided a general description of them: "Tasks were developed that required *S*s to demonstrate their understanding of classificatory operations by correctly manipulating a set of geometric blocks. The blocks were either square, circular or triangular, and the colors were usually blue, red, green or yellow. The labels employed by *S*s in describing the colors and shapes during the initial screening device were subsequently used by *E* to describe the blocks throughout the regular testing session" (p. 194).

In contrast to the Piagetian paradigm of analyzing behavior in terms of the logical and algebraic properties of the experimental situation, our approach is to analyze the information processing requirements of the tasks. Accordingly, the initial question posed is "What routines for processing information would a child need in order to perform these tasks?" Our partial answer to this question constitutes the substance of this paper. We will briefly sketch the outlines of our answer as a prelude to a more extended discussion in the sections that follow.

We believe that the major task facing the child who has just been presented with an experimental task is to assemble, from his repertoire of fundamental information handling processes, a routine that is sufficient to pass the task at hand. We view the information processing demands of the tasks as being analogous to the compilation and execution of a computer program. In Fig. 1 we show the elements of the system. Incoming visual and verbal stimuli are first encoded into internal representations. Then the assembly system attempts to construct, from its repertoire of fundamental processes, a task-specific routine that is sufficient to meet the demands of the verbal instructions. Having assembled such a routine, the system then executes it.

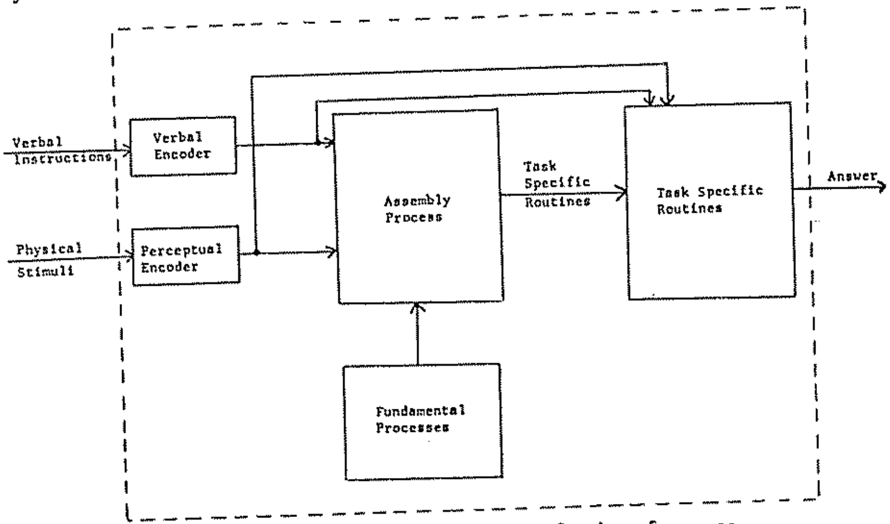

FIG. 1. Information processing model of task performance.

In the sections that follow we will offer detailed descriptions of three parts of this model: the internal representation of experimental objects, a collection of fundamental processes, and a set of task-specific routines consisting of particular arrangements of fundamental processes that are sufficient to produce a successful performance on the seven classification tasks. We shall *not* describe in any detail a mechanism for a linguistic encoder or an internal representation of task instructions, an encoder that maps external objects into internal representations, or the assembly process whereby the subject constructs the task-specific routines. All of these issues will, however, be considered later in the discussion.

REPRESENTATION

An important feature of any model of cognitive activity is the process whereby external objects are mapped into some internal representation. One approach is simply to finesse the problem by encoding those elements of an object that are deemed necessary into an internal model. From this orientation any assumptions about the difference between "reality" and internal representations of reality are a part of the model-builder who is doing the coding and may be as explicit as he cares to make them. They are, therefore, not an explicit part of the process model, although they affect the kinds of information available to the processes. Another approach is to postulate a model of the encoding process itself. This model is then given a very rich, almost "complete" external representation of objects and allowed to carry out the fabrication of the internal representation. Such an encoder would include explicit assumptions about such processes as perception, attention focus, redundancy elimination, and distortion.

While the second approach clearly produces a more complete model, it is not considered to be necessary in the case of the Kofsky tasks. In most of them the experimental situation is such that the subject is trained at the beginning of each task to acquire labels for the relevant values and attributes of all of the stimuli to be manipulated. Thus, no matter how sophisticated or complex an encoding process might be postulated for the general external-internal mapping, it would in these cases result in little more than a one-to-one mapping of (Experimenter) desired information from external object to internal representation. This relative simplicity on the encoding dimension makes the Kofsky tasks particularly suitable for use in trying out a new methodological approach.

Since the symbolic objects and collections of objects on which processes operate are inextricably linked to the processes themselves, any decisions taken about the nature of object representation clearly have considerable processing implications. The simple representation adopted will be explained and illustrated in terms of the first of the seven experimental tasks, consistent sorting (CS). This has been described by Kofsky (1966) as follows:

"A mixed array of blocks consisting of two red, one green, one blue, and one brown triangle; a red, a yellow, a blue and a green square; and a red, a yellow, and a blue circle was presented to *S*, who was to place together some things that were alike and to explain the reason for his grouping. Subjects who grouped just two objects were encouraged to find more things that were alike.

"Consistent classifiers selected three or more objects which were alike in some perceptual feature" (p. 195).

Each object is represented by a list of its values on a number of attributes. This list bears the name of the object. As Fig. 2 indicates, OBJECT 1 in the array employed in CS is represented by a list called OB1 comprising the values RED TRIANGLE. Each of the other objects in the

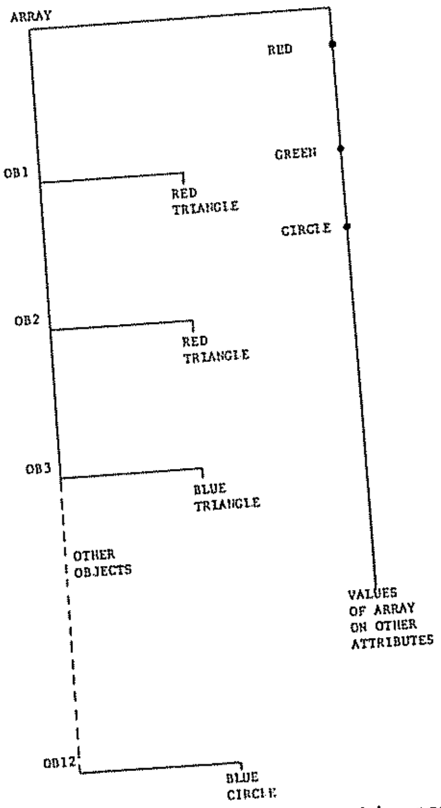

FIG. 2. Internal representation of array of objects used in consistent sorting (CS) task.

array is similarly represented by a list of values. The decision to compose the object lists of values alone rather than of attributes and values (e.g., OB1—COLOR–RED; SHAPE–TRIANGLE) was based on the psychological hypothesis that attribute names are not "carried" directly by objects. When tasks demand reference to the attributes represented by particular values, the names of the attributes in question are determined by a process called ATTOF. This makes use of lists each of which has the name of an attribute and comprises its possible values (e.g., COLOR—RED–BLUE–GREEN). These attribute lists are considered to be part of the child's knowledge of the world obtained as a result of previous experience. They do not constitute a part of the representation of specific task situations.

The method employed to link together the individual object representations into a representation of the array of objects presented to the subject is also based on the use of lists. The array, as indicated in Fig. 2, is basically represented as a list of objects (i.e., a list of lists). As in the case of individual objects, provision must be made for representing the values which characterize the array on specific attributes. Once again a list method is adopted. A description list of values is attached to the array. The information conveyed by this list could be expressed verbally as "There are red, green . . . objects in the array, there are triangles, squares. . . ."

The main list for the set, consisting of descriptions of each object, represents information about the *intensive* properties of the objects in the set, while the description list for the set itself consists of *extensive* properties. Such things as set size and the spatial distribution of objects could be included in the set description list.

FUNDAMENTAL PROCESSES

In this section we describe several fundamental information handling processes that constitute the "building blocks" for the assembler in Fig. 1 when it attempts to construct a task-specific routine.

We postulate the existence of these processes or their functional equivalents in the child prior to the presentation of the experimental tasks. The prior existence of the processes is not in itself sufficient to guarantee successful performance, however, for the processes (as we shall indicate below) are themselves error prone. Furthermore, the assembly process or the visual and verbal encoding processes may fail to produce the correct task-specific routine. The fundamental processes are divided into primary, secondary, and tertiary processes. The primary processes will be described in terms of their input/output relations, the secondary processes in terms of the primary processes, and the tertiary processes in terms of the primary and secondary processes. The similarity between some of the primary processes and IPL-V processes will be obvious to those familiar with that computer language (Newell *et al.*, 1964).

Primary Processes

It should be emphasized at the outset that the adjective "primary" is applied to these processes solely to indicate that they represent the level beyond which information processing analysis has not yet been attempted, and not by any conviction that they constitute elementary unanalyzable processes. In the following descriptions, we present the general form of each process, a specific example based upon the objects in Fig. 2, and an explanation of what the process does.

a. IS $(\alpha, \beta) \rightarrow$ *truth value. Ex. IS (OB1, RED)* $\rightarrow$ *TRUE.* This process provides an answer to the question "Does object α have value β?" It takes as input an object name and a value; it produces as output a truth value (i.e., "true" or "false") contingent upon the existence of the value on the list of values of the named object.

b. NOTICE $(\alpha) \rightarrow$ *value. Ex. NOTICE (OB1)* $\rightarrow$ *TRIANGLE.* This process provides a response to the command, "Name a value that object α possesses." It takes as input an object name and outputs some value possessed by that object. This usually involves selecting one of several potential outputs. The procedure for deciding which value is to be chosen is, at present, based on a simple, nonrandom rule for dealing with such choices among alternatives. This is based on an ordered list representing the relative saliency of values as a result of the child's experience to date. The item currently occupying top place in the list is selected.

In addition to selecting values possessed by single objects, NOTICE can also operate on an array, collection, or set of objects (i.e., a list of objects). The output consists of a single object selected from the set. When NOTICE operates upon a set, it marks objects as it processes them, so that it does not return to a just noticed object. As in the case of values of single objects, the selection process is guided by an ordered list which indicates the relative saliency of objects. The processes necessary for the formation and constant revision of the ordered saliency lists have not yet been defined.

c. VALOF $(\alpha, \beta) \rightarrow$ *value/nil. Ex. VALOF (OB1, COLOR)* $\rightarrow$ *RED.* This process answers the question, "What is the value of attribute β for object α?" It finds the value of the input attribute on the input object. If a value of the attribute does not exist for that object, the output is nil.

d. ATTOF $(\alpha) \rightarrow$ *attribute. Ex. ATTOF (BLUE)* $\rightarrow$ *COLOR.* Given a value as input, ATTOF outputs the attribute which can assume that value. It answers the question, "What is the attribute for which α is a value?" This process makes use of lists each of which corresponds to an attribute and carries its values. These lists are regarded as being part of the child's structured experience.

The primary processes IS, NOTICE, and VALOF deal directly with the internal representations of the task situations, and, consequently, any change in the form of the representations would necessitate a change in these processes. None of the other processes deal directly with the task representations since they can obtain the necessary information through IS, NOTICE, and VALOF. These processes can be nested to represent a complex question. For example, the question, "Is Object 2 the same color as Object 3?" can be represented as IS [OBJECT 2, VALOF (OBJECT 3, COLOR)]. The input/output relationships for all fundamental processes are summarized in Table 1.

TABLE 1
Fundamental Processes: Input-Output Relationships

Primary	
IS (object, value)	→ truth value
NOTICE (object)	→ value
VALOF (object, attribute)	→ value/nil
ATTOF (value)	→ attribute
Secondary	
SAMEANY (object *x*, object *y*)	→ value/nil
DIFANY (object *x*, object *y*)	→ attribute/nil
FINDONE (value, set)	→ object/nil
Tertiary	
FINDALL (value, set)	→ list of objects
PAIRSAME(set)	→ object *x*, object *y*, value/nil
PAIRDIF (set)	→ object *x*, object *y*, attribute/nil
COUNT (set, value)	→ number

Secondary Processes

The secondary processes differ from the primary processes in two respects. First, the secondary processes use the primary processes as components and are, therefore, at a higher level. Second, all the secondary processes include a motivational mechanism while the primary processes, at present, do not. The heterogeneous nature of the primary processes and the basic complexity of NOTICE and ATTOF make it expedient at the present time to handle the additional complexity introduced by motivational considerations at the level of secondary processes.

The initial method adopted of representing the effect on the functioning of the secondary processes of variations in the child's motivational state is extremely primitive. It enables each of the secondary processes to respond in one of three ways: by producing a correct object, value, or attribute; by producing an incorrect object, value, or attribute; or by failing to find anything when in fact the desired object, value, or attribute is available.

The procedure works as follows. If the required outcome has not been obtained at the conclusion of a pass through the statements comprising a secondary process, a function named DRIVE is called. DRIVE determines whether the search for the appropriate outcome will be continued by entering a loop which will lead to a recycling through the sequence of statements or discontinued with the consequent production of an erroneous output. DRIVE is a Boolean function which tests the current value of a parameter called MOTIVE. If MOTIVE lies within (arbitrarily determined) threshold values then DRIVE is "true," the loop is entered, and the execution of the process continues. When MOTIVE does not lie within the prescribed limits, execution of the secondary process ceases and the nature of the erroneous output from it is determined by a function named LIMIT. The variations in the output required by the secondary processes necessitate minor variants of the function LIMIT. All are identical, however, in their basic structure and in the two types of output which they can produce.

LIMIT gives rise either to an erroneous positive response, consisting of an object, value, or attribute which does not satisfy the conditions of the process, or to an erroneous negative response. The latter involves the production of a "nil" response indicating that no object, value, or attribute satisfying the necessary conditions has been found when, in fact, such an element may exist in the task situation. Which of these errors is committed on a particular occasion is, once again, determined by consulting the current value of the parameter MOTIVE. It is thus possible, for example, to link the making of a positive erroneous response to a level of MOTIVE higher than the upper threshold set for the production of a "true" value by DRIVE. This would be a step towards the simulation of an impulsive, relatively unreflective performance stemming from too high a drive level. Similarly a careless, uninvolved, or negativistic mode of performance associated with a low level of drive could be simulated by linking the making of an erroneous negative response to a level of MOTIVE lower than the lower threshold set for the production of a "true" value by DRIVE. The adoption of both of these links simultaneously by the experimenter, as in the example of LIMIT listed in Table 2, would represent a simulation of the hypothesis that a medium level of drive represents the optimal condition for successful task performance.

The above examples of the performance effects which can be obtained by manipulating the value of the parameter MOTIVE have all been discussed solely in terms of drive level. This parameter, however, is intended to be a crude representation of the motivational outcome of the complex interaction of a wide range of psychological processes. As Simon (1967) has indicated, these include the criteria adopted by the child to determine attainment of the goal towards which his behavior is directed and the interrupting and other effects of social interaction while a task is being tackled.

Although the model presented here lacks an encoder, and thus assumes that all relevant information is available for processing, it is possible to reflect changes in the child's attention through both changes in the current value of MOTIVE and in the saliencies of attributes and values.

A verbal description of the secondary processes will now be presented. It should be pointed out at the outset that they are only claimed to be well defined in the context of the particular task situations used by Kofsky. They do not, for example, include provision for coping with objects lacking a value on a particular attribute since this contingency does not arise in the task situations to be tackled. Since the only complete, detailed, and unambiguous reference for the processes is a listing of them in computer language, such a listing of the secondary and tertiary processes is provided in Table 2. A listing of one specific form of the function LIMIT is also included to illustrate its general structure. The language used, POP-2, is described in Burstall *et al.* (1968). A similar listing of the primary processes is not presented since the greater complexity necessitated by the fact that they deal directly with the task representations renders it unlikely that the programs would be intelligible to readers unfamiliar with the particular language being used. In the descriptions below, we indicate parenthetically which primary processes are used in the secondary processes.

a. SAMEANY takes the names of two objects as input and attempts to answer the question, "What is the value that object *x* and object *y* have in common?" The procedure is:

```
1. Notice a value of object x (NOTICE).
2. Determine whether object y has the same value (IS).
    2.1. If it has, exit and output the name of the value.
    2.2. If it has not, go to step 3.
3. Test DRIVE.
    3.1. If DRIVE is true, go to step 1.
    3.2. If DRIVE is false, call the function LIMIT to determine whether an erroneous
         negative output "nil" will be produced or an erroneous positive output (i.e., a
         value of the object x which the object y does not have).
```

b. DIFANY takes the names of two objects as input and attempts to

TABLE 2
Algorithms for Secondary and Tertiary Processes

```
Secondary processes
  function SAMEANY x y ⇒ val;
  loop: NOTICE (x) → val; if IS (y,val) then exit;
  if DRIVE then goto loop else LIMIT (val) exit end;
    (LIMIT val ⇒ val;
      if motive > max then val exit;
      if motive < min then nil → val exit end;)

  function DIFANY x y ⇒ att; vars val;
  loop:  NOTICE (x) → val;
  if not (IS (y,val)) then ATTOF (val) → att exit;
  if DRIVE then goto loop else LIMIT (val) exit end;

  function FINDONE val set ⇒ obj;
  loop: NOTICE (set) → obj; if IS (obj,val) then exit;
  if DRIVE then goto loop else LIMIT (obj) exit end;

Tertiary processes
  function FINDALL val set ⇒ olst; vars ob;
  loop: FINDONE (val,set) → ob; PLACE (ob, olst);
  if ob = nil then exit;
  if DRIVE then goto loop else exit end;

  function PAIRSAME set ⇒ obx oby val;
  loop: NOTICE (set) → obx; NOTICE (set) → oby;
  SAMEANY (obx,oby) → val; if val ≠ nil then exit;
  if DRIVE then goto loop else LIMIT (oby) exit end;

  function PAIRDIF set ⇒ obx oby att;
  loop: NOTICE (set) → obx; NOTICE (set) → oby;
  DIFANY (obx,oby) → att; if att ≠ nil then exit;
  if DRIVE then goto loop else LIMIT (obx) exit end.

  function COUNT set val ⇒ n; vars ob;
  0 → n
  loop: FINDONE (val,set) → ob;
  if ob = nil exit;
  n + 1 → n; go to loop
  end.
```

answer the question, "What is the attribute on which objects x and y exhibit different values?" The procedure is:

1. Notice a value of object x (NOTICE).
2. Determine whether object y has the same value (IS).
   2.1. If it has not, then determine the attribute of which this value is an example (ATTOF). Exit and output the name of this attribute.
   2.2. If it has, go to step 3.

3. Test DRIVE.
   3.1. If DRIVE is true, go to step 1
   3.2. If DRIVE is false, call the function LIMIT. An erroneous "nil" response or an erroneous attribute name will be output. The latter is the name of the attribute represented by the current value of object x being considered.

c. FINDONE takes as input a collection of objects and the name of a value and tries to comply with the order, "Find an object with this value in the collection." The procedure is:

1. Notice an object in the collection (NOTICE).
2. Test for existence of value on object (IS)
   2.1. If the object has the value in question, exit and output the name of the object.
   2.2. If the object does not have the value, go to step 3
3. Test DRIVE.
   3.1. If DRIVE is true, go to step 1.
   3.2. If DRIVE is false call LIMIT. An erroneous "nil" response or an erroneous object name will be output. The latter will be the name of the object currently under consideration

It should, perhaps, be reiterated at this point that the order in which the objects in the collection are considered is determined by the current state of the ordered list indicating the relative saliency of objects. This list forms the basis for the operation of the primary process NOTICE and also determines the sequence of pairs of objects considered in the tertiary processes PAIRSAME and PAIRDIF to be described below. Although the possibility of an infinite loop seems to exist in some processes such as SAMEANY, processing will in fact be terminated by both the saliency mechanism and the level of MOTIVE.

Tertiary Processes

The tertiary processes use both the primary and secondary processes as components and are, thus, at a yet higher level. A motivational mechanism similar to that included in the secondary processes is, also, a feature of the tertiary processes.

a. FINDALL takes as input a collection of objects and the name of a value and tries to comply with the order, "Find all the objects with this value and place them together." Whether it succeeds in detecting all or only some of the appropriate objects is largely determined by the function DRIVE. The procedure is:

1. Find an object in the set with the desired value (FINDONE).
   1.1. Place this object on the output list and delete it from the list representing the collection of objects. (This is carried out by a special function PLACE which represents the physical action of grouping the selected objects apart from the rest of the collection.)
2. Test for whether an object was found in step 1.

```
    2.1. If no such object is found, exit and output the list of objects selected to date.
    2.2. If an object has been found, go to step 3.
3. Test DRIVE.
    3.1 If DRIVE is true, go to step 1.
    3.2 If DRIVE is false, output the list of objects selected to date. (The nature of
        FINDALL does not necessitate a direct call on LIMIT; note that in step 1,
        FINDONE may call LIMIT.)
```

b. PAIRSAME takes as input a collection of objects and attempts to execute the order, "Find two objects in the collection which have the same value on an attribute and indicate what that value is." Output is the name of two objects and a value. The procedure is:

```
1. Select two objects from the collection (NOTICE)
2. Determine if the two objects (x and y) have the same value on an attribute
   (SAMEANY).
    2.1 If they do, exit and output the names of the two objects and the value.
    2.2. If they do not, go to step 3.
3. Test DRIVE.
    3.1. If DRIVE is true, go to step 1.
    3.2 If DRIVE is false, call LIMIT to determine whether an erroneous "nil" response
        or an erroneous value name will be output. The latter is the last value of x sought
        on y's list list of values.
```

c. PAIRDIF takes as input a collection of objects and responds to the order, "Find two objects in the collection which have different values on an attribute and indicate what the attribute is." Output is the name of two objects and an attribute. The procedure is:

```
1. Select two objects from the collection (NOTICE).
2. Determine if the two objects (x and y) have different values on an attribute (DIFANY).
    2.1 If they have, exit and output the names of the two objects and the attribute.
    2.2. If they have not, go to step 3.
3. Test DRIVE
    3.1. If DRIVE is true, go to step 1.
    3.2. If DRIVE is false, call LIMIT to determine whether an erroneous "nil" response
         or an erroneous attribute name will be output. The latter is the attribute
         represented by the last value of x to be sought on y's list of values.
```

d. COUNT tallies the number of objects having a specified value.

```
1. Set tally to 0.
2. Find an object with the specified value.
    2.1. If none exist, exit (FINDONE).
    2.2 If one found, go to step 3.
3. Increment tally by 1.
4. Go to step 2.
```

Recall that FINDONE uses NOTICE, and NOTICE, when operating upon a set of objects, marks them as having been processed. Thus there is no possibility for double counting in COUNT.

TASK-SPECIFIC ROUTINES

Having provided an account of the three levels of fundamental processes, it is now possible to describe the complex routines which the assembly process would have to construct from these processes to tackle the experimental tasks used by Kofsky. A full verbal account of these routines will be presented and, as in the case of the secondary and tertiary processes, Table 3 comprises a listing of them in POP-2.

a. Consistent sorting (CS). The general situation and the particular nature of this task have already been described above. It will be recalled that to succeed on the item the child had to select from an array of blocks three or more which were alike in some perceptual feature.

The routine for consistent sorting starts off by finding two objects with a common value and then collects some of the remaining objects in the array with that value. The duration of the collecting activity is determined by the nature of the goal which the child has set himself as a result of the instruction, "to place together some things that are alike" and, in particular, by the interpretation which he places on the word "some." This factor is represented in the routine by means of the DRIVE function and the MOTIVE parameter which operate through the secondary process FINDONE and determine when the child ceases to add objects to the collection. The same motivational mechanism can be used to represent the effect on a child's goal of the additional instruction to "Find more things that are alike," which was given to those who initially grouped only two objects. As in the case of the tertiary process FINDALL, the special function PLACE is used in the CS routine to represent the physical action of grouping together the objects that are alike. The complete procedure in the CS task routine is as follows:

```
1. Find two objects with a common value (PAIRSAME). Call them object x and object y.
   Remember their common value
   1.1. If no such objects exist, exit with "nil."
2. Start grouping by placing object x into "grouping region."
3. Put object y into grouping region.
4. Find another object in set with the value that object x and object y share (FINDONE).
   4.1. If no more exist, then quit.
   4.2. If one is found, call it object y; go to step 3
```

The output from this procedure is a list of the objects in the grouping region, all possessing a common value, and the name of that value.

b. Exhaustive sorting (EC). "A collection of blocks, including a red and a blue circle, one green and two blue squares, two red and two green triangles, was shown to *S*. He was to choose a block and put it in a box along with all the others that were 'like it.' After the first box was filled, the procedure was repeated with the remaining blocks until all the blocks had been chosen."

TABLE 3
Task-Specific Routines

```
function CS set ⇒ olst val; vars obx oby; nil → olst;
PAIRSAME (set) → obx → oby → val; if val = nil then exit;
PLACE (obx, olst);
loop: PLACE (oby, olst); FINDONE (set, val) → oby;
if oby = nil then exit; goto loop end;

function EC set ⇒ olst; vars obx valx attl;
NOTICE (set) → obx; PLACE (obx, olst);
NOTICE (obx) → valx;
FINDALL (valx, set) → olst
ATTOF (valx) → attl;
loop: NOTICE (set) → obx; PLACE (obx, olst); if obx = nil then exit;
VALOF (obx, attl) → valx;
FINDALL (valx, set) → olst;
goto loop end;

function MM set valx valy ⇒ truth; vars obx;
loop: FINDONE (valx, set) → obx; if obx = nil then true → truth exit;
if IS (obx, valy) then goto loop else false → truth exit end;

function HR set ⇒ olst; vars valx obx att1 att2;
NOTICE (set) → obx; PLACE (obx,olst); NOTICE (obx) → valx;
FINDALL (valx,set) → olst;
ATTOF (valx) → att1;
loopa: NOTICE (set) → obx; PLACE (obx,olst); if obx = nil go to alpha;
VALOF (obx,att1) → valx;
FINDALL (valx,set) → olst; go to loopa;
alpha: RESET;
loopb: NOTICE (set) → obx; PLACE (obx,olst); NOTICE (obx) → valx;
ATTOF (valx) → att2;
if att2 ≠ att1 go to beta else go to loopb;
beta: FINDALL (valx,set) → olst;
loopc: NOTICE (set) → obx; PLACE (obx,olst); if obx = nil exit,
VALOF (obx,att2) → valx;
FINDALL (set,valx);
go to loopc;
end;

function VC set ⇒ somlist, somval, allval; vars obx oby att1
loopa: NOTICE (set) → obx; NOTICE (set) → oby
DIFANY (obx, oby) → att1; if att1 = nil go to loopa;
VALOF (obx, att1) → somval;
FINDALL (set, somval) → somlist;
NOTICE (set) → obx; NOTICE (somlist) → oby;
SAMEANY (obx, oby) → allval
end;

function SA set valx valy ⇒ truth;
MM (set, valx, valy) → truth end;
```

TABLE 3 (*Continued*)

```
function CH1 set valx valy valz ⇒ valist; vars obx;
FINDALL (set,valx);
FINDONE (set,valy) → obx;
valy → valist;
if IS (obx, valz) then valz::valist → valist;
end;

function CH2 set valx valy valz ⇒ valist; vars nl n2;
FINDALL (set. valx);
COUNT (set,valy) → nl; RESET; COUNT (set.valz) → N2;
if nl > n2 valx → valist; return else
if nl < n2 valy → valist; return else
valx::valy → valist close
end;
```

"In an exhaustive sort, *S*s consistently used an attribute to select the contents of each box and filled the box with all the blocks that possessed the criterial attribute" (Kofsky, 1966, p. 195).

Note that in order to pass this task, the child must use the *same* attribute for each sort. If, for example, he selects objects to place in the first box on the basis of their color he must select other colors as the basis of subsequent sorts. He is *not* allowed to sort first all blue things and then all remaining triangular things, etc. (Kofsky, 1963, pp. 90–91)

1. Select a block from the collection and place it in the box (NOTICE).
   1.1. Select a value of the block (NOTICE).
2. Find all the blocks remaining in the collection that have the value selected in step 1.1. Place them in the box (FINDALL).
3. Determine the attribute of the value selected in step 1.1 (ATTOF).
4. Select a block from the remaining collection and place it in an empty box (NOTICE).
   4.1. If none are left exit; output is content of boxes
   4.2. If a block is found, go to step 5.
5. Find the value of the block just selected on the attribute determined in step 3 (VALOF).
6. Find all the blocks remaining in the collection that have the value determined in step 5 (FINDALL).
7. Go to step 4.

c. Multiple class membership (MM). "A set of triangles varying in size (large or small) and color (red or green) was placed in front of *S*. The plane surface of the small triangles measured four square inches in area, and the large ones were approximately nine square inches in area. The large triangles were either red or green, but all the small ones were red. The object of the task was to determine whether any of the blocks could be placed in more than one category.

"The experimenter asked:

1. This is a bag full of red things. Do all of the small things belong in the bag with the reds? Why? (Yes, all of the small blocks are red).
2. This is a bag for triangles. Do the greens belong in the bag? Why? (Yes, the greens are triangles).
3. Do the reds go in the bag for triangles? Why? (Yes, all the reds are triangles).
4. This is a bag for small blocks. Do the greens belong in it? Why? (No, the greens are all large).

"The order of questions was varied for each *S*. Successful performance entailed three correct responses. Acceptable answers for this and other tasks appear in parentheses" (Kofsky, 1966, p. 196).

All of the questions about multiple class membership are variants of the basic form, "Do all objects that have value *x* also have value *y*?" Question 2, for example, is equivalent to, "Are all the things that are green also triangular?" Since no model is offered at present of the process by which the features of the external task situation are encoded into an internal representation, no attempt has been made as yet to account for this linguistic mapping from the experimenter's question to an equivalent routine. The algorithm for multiple class membership first finds an object with value *x* and then tests for value *y* on that object. This operation is carried out repeatedly on the objects in the set. The procedure is :

```
1. Find an object with value x (FINDONE).
   1.1. If no such object exists, then (since all have been successfully tested) exit with
        true (i.e., the answer to the question is affirmative).
   1.2. If such an object is found, go to step 2.
2. Does the object have value y?
   2.1. If yes, go to step 1.
   2.2. If no, then an object has been found which has value x, but not value y. Exit with
        false (i.e., the answer to the question is negative).
```

d. Horizontal reclassification (HR). "The eight wooden blocks to be sorted included pairs of reds, yellows, greens, and blues. One of each color was a triangle and the other a square. Each *S* was (a) to sort all the objects that were alike into classes, (b) to sort a different way, and (c) to explain each complete grouping. To pass the test, *S* was required to sort the blocks into groups in which (1) all the blocks were alike in some respect, and (2) all the objects possessing the criterial attribute were included in the group. Subsequently, *S* changed his criteria for sorting to produce a new arrangement which conformed to the above requirements. In other words, one time he sorted completely by color and another time by shape" (Kofsky, 1966, p. 196).

The procedure is essentially an Exhaustive Sort done twice, with a different criterial attribute each time. The algorithm listed in Table 3 includes a special process RESET which corresponds to the experimenter's reassembly of the blocks after the first sorting to allow the child to produce a new arrangement.

```
1. Select a block from the collection and place it in an empty box (NOTICE).
   1.1. Select a value of the block (NOTICE).
2. Find all the blocks remaining in the collection that have the value selected in step 1.1.
   Place them in the box (FINDALL).
3. Determine the attribute of the value selected in step 1.1, call it att1 (ATTOF).
4. Select a block from the remaining collection and place it in an empty box (NOTICE).
   4.1. If no more objects remain in collection, go to step 7; output is contents of boxes.
   4.2. If object is found, go to step 5
5. Find the value of the block just selected on att1 (VALOF)
6. Find all the blocks remaining in the collection that have the value determined in step 5
   (FINDALL).
   6.1. Go to step 4
7. Experimenter intervenes by reassembling blocks (RESET) and giving instructions, "I
   want you to put the blocks together in a different way. Find some new ones to go in
   each box" (Kofsky, 1963, p. 201).
8. Select a block from the collection and place it in an empty box (NOTICE).
   8.1 Select a value of the block (NOTICE).
9. Determine attribute represented by this value, call it att2 (ATTOF).
10. Is att2 different from att1?
    10.1. If not different, go to step 8.1
    10.2. If different, go to step 11.
11. Find all blocks remaining in the collection that have the value selected in step 8.1.
    Place them in the box (FINDALL)
12. Select a block from the remaining collection and place it in an empty box (NOTICE).
    12.1. If no more blocks, exit; output is contents.
    12.2. If object found, go to step 13.
13. Find the value of the block just selected on att2 (VALOF).
14. Find all the blocks remaining in the collection that have the value determined in step
    13 (FINDALL)
    14.1. Go to step 12
```

Since this routine is the most lengthy thus far, we will point out some of its important features. Steps 1 through 6 above are identical to the first six steps in the Exhaustive Sorting routine. The routine first notices an object, then a value of that object, and then the attribute of the noticed value. The attribute (att1) then controls the first sort of the collection. Following the experimenter intervention in step 7, the routine then shifts attention from object to value to attribute once more (steps 8–9) and then performs the crucial test for reclassification by ensuring that it does not sort on the same attribute again (step 10). The remainder of the routine (steps 11–14) is similar to steps 2 through 6.

e. Hierarchical classification (VC). "In this task there were seven triangles, four of which were red and the rest blue, which were arranged in two parallel rows. Each row contained both reds and blues. The *E* said, 'all of these are called MEF's (points to each), but only some are TOV's.' 'What are MEF's?' 'Which are the TOV's?' The child was to point to the MEF's and TOV's and explain his actions. TOV, like MEF, was selected from the Glaze list of nonsense syllables eliciting few associations.

"To classify correctly, *S* had to define MEF's in terms of some attribute shared by all the members of the group (such as triangle). TOV's were defined by an attribute shared by one part of the group but not the entire group (such as 'blue')" (Kofsky, 1966, pp. 196–197).

The output of the routine for VC consists of (*a*) a list of objects that are TOV's; (*b*) the value that TOV's have in common; and (*c*) the value that MEF's have in common. The procedure is:

1. Select two objects from the collection (NOTICE), and determine any differences (DIFANY).
   1.1 If no difference is found, go to step 1.
   1.2. If a difference is found, go to step 2.
2. Remember the name of the attribute on which the objects differ, call it att1.
3. Find the value of one of the objects selected in step 1 on att1. This value will be the common value of TOV's (VALOF).
4. Find all the remaining objects that have the value determined in step 3, i.e., find all the TOV's (FINDALL).
5. Select one object from the list of TOV's, and one object from the remaining objects in the collection (NOTICE).
6. Find a value that is shared by the two objects selected in step 5 (SAMEANY). This value is the common value of MEF's.

f. "Some" and "all" (*SA*). "There were nine blocks differing in color and shape. Among the six blue figures were two triangles and four squares. The three red figures were all triangles. In Inhelder and Piaget's (1959) notation, a superordinate class is called B, its subclasses A and A′ with A the larger of the two subclasses. The class of blue figures (B) contained four squares (A) and two triangles (A′). The class of triangles (B) contained three reds (A) and two blues (A′). Each *S* had to determine whether the members of the superordinate class (B) were all members of one subordinate class (A), and the converse, if all the A's belong to B. The order of questions varied randomly among the *S*s. As in other tasks, the categories of blocks were mixed so that the subdivisions were not readily apparent to *S*. First *S*s were asked to find the reds, blues, triangles and squares. Then *E* asked:

1. Are all of the reds (A) triangles (B)? Why? (Yes, every red is a triangle.)
2. Are all the triangles (B) red (A)? Why? (No, some triangles are blue.)

3. Are all of the squares (A) blue (B)? Why? (Yes, every square is blue.)
4. Are all of the blues (B) squares (A)? Why? (No, there are some blue triangles.)

"Three correct responses were necessary for passing" (Kofsky, 1966, p. 197).

The information processing requirements imposed by these questions are identical to those for multiple class membership. The SA and MM tasks, thus, do not differ in their process or logical requirements but only in the way in which the questions are posed verbally to the child. Since the distinction between the tasks lies in the province of the encoder which the model at present lacks, the SA task is dealt with by the simple expedient of calling on the routine already outlined for tackling the MM item.

g. Conservation of hierarchy (CH). "Two blue wooden squares were mixed in among a half-dozen red ones. The experimenter asked: Are all of these squares? Are the red ones square? Are the blues square?" Then the experimenter asked:

1. "If I took away all the reds, are there just blues left, just squares left, or both blues and squares? Why? (Both blues and squares since the remaining blocks are all blue and square.)"
2. "If I took away all the reds, would there be more blues or more squares left or as many blues as squares? (Since all the remaining blocks are blue and square, there are as many blues as squares left.)" (Kofsky, 1966, pp. 197–198). Two correct answers are needed for success.

The procedure for CH1 takes as input the three values in the question and then proceeds:

```
1. Find all of the blocks with value $x$, i.e., "take away all the reds" (FINDALL)
2. Select an object in the remaining collection with value $y$ (FINDONE).
3. Add value $y$ to the output list.
4. Does the object just selected also have the value $z$?
   4.1. If yes, add value $z$ to output list
   4.2. If no, go to step 5.
5. Output the value list.
```

The detailed experimental instructions for the second part of the CH task are "If I took away all the reds and you counted what was left, would there be more blues left or more squares left or as many squares as blues? Why? Count the blues that would be left. Count the squares that would be left" (Kofsky, 1963, p. 202). The procedure takes as input the three values in question 2 and then proceeds:

```
1. Find all of the blocks with value $x$ (FINDONE).
2. Count all the blocks remaining with value $y$. Call this amount N1 (COUNT)
3. Count all the blocks remaining with value $z$. Call this amount N2 (COUNT).
4. Compare N1 to N2.
   4.1. If N1 > N2, put value $x$ on value list. Go to step 5.
   4.2. If N2 > N1, put value $y$ on value list. Go to step 5.
   4.3. If N1 = N2, put both value $x$ and value $y$ on value list.
5. Output value list. Exit.
```

DISCUSSION

In the preceding sections we have elaborated upon the general information processing model presented in Fig. 1. The primitive nature and narrow basis of the current form of the model are obvious: it lacks an encoder to map the features of the external physical and linguistic situation into internal representations; the effects of the complex interactions of a wide range of motivational and attentional factors are crudely represented by a single parameter; the sum total of the knowledge of the world is represented by a structure of attribute names and their associated values; no model of the assembly process is provided. Above all, the model does not include processes which would enable it to engage in self-modification of structure.

In spite of these current limitations, this model constitutes a sufficient theoretical representation of many of the psychological processes employed by a child capable of successfully completing the seven classification tasks described above. That is, a running program based upon the model in Tables 1, 2, and 3 would be able to simulate the actual behavior observed by Kofsky. In this section we will attempt to demonstrate the contribution that a model such as this can make towards answering the fundamental developmental questions raised at the start of the paper.

Interitem Associations

The process analysis of the group of classification tasks indicates that the extreme difficulty experienced in attempts to detect exact relations between items tapping specific aspects of Piaget's level of concrete reasoning may simply be due to the fact that no such exact relations can be expected to exist, even when subjects possess processes sufficient to deal with all of the items in question. The shift of emphasis from the logical and algebraic properties of task situations to the processes sufficient for their solution suggests that attempts to obtain a "pure" picture of the structural relationships between concrete operational tasks by removing the effects of variations in the stimulus situations and goal objects are doomed to failure. The reason for this gloomy prognosis is that some of the sources of the so-called "irrelevant" variations, such as the effects of attention and motivation, are, in our view, built into the fundamental processes. It appears inevitable that this feature will be characteristic of any other information processing model of childrens' performance on these tasks. Even in the case of error-free encoding of verbal and physical stimuli and error-free assembly, the motivational and attentional mechanisms become indissociable parts of the final task-specific routines because every task-specific routine consists of several fundamental processes that, through the parameter DRIVE and the functions ATTOF and NOTICE, contain sources of inconsistency. Thus, there seems to be little basis for the strategy of seeking to tap "situation-free" inference patterns through procedural refinements. Such attempts can affect the course of encoding and assembly, but they cannot remove the sources of variation within the fundamental processes.

The apparent contradiction between Piaget's (1956) view that the system of operations constituting concrete operations comes into existence simultaneously and the increasing number of empirical findings supporting varying sequential developmental relations between operations (Flavell & Wohlwill, 1969) can be resolved in terms of the model presented here. The requirement for the simultaneous coexistence of a set of fundamental processes for successful performance on a range of concrete operational tasks is consistent with Piaget's theoretical position, while at the task performance level the empirically demonstrated inconsistencies in the developmental sequences can be explained in terms of the sources of variation which form an integral part of the fundamental processes.

How can the validity of the argument advanced above be evaluated? A hypothesis which predicts the inevitability of inconsistency is difficult to falsify. Within a circumscribed area, such as that provided by the Kofsky tasks, a possible approach may lie through the prediction of relative task difficulty. If the argument is accepted, the ability to perform successfully on a particular type of concrete operational task is never viewed as an absolute. Instead, it depends upon the effect of situational and experimental factors at the moment when the subject is required to perform a particular variant of the task. Predicting task difficulty involves assessing the effect of these factors in advance. This could be attempted in the case of the Kofsky tasks by producing programs sufficient to cope with several variants of several of the types of tasks. A prediction of the order of difficulty of the variants could be made on the basis of their relative level of demand on the services of MOTIVE, NOTICE, and ATTOF. This approach would permit the mounting of convincing tests of the hypothesis. It should, for example, be possible to devise two variants of each of two types of tasks (A1, A2 and B1, B2) such that the program-based prediction of their relative difficulties would be that A1 is more difficult than B1, but B2 is more difficult than A2. Consider, for example, the tasks HR and VC. For each task, appropriate variations in the number of attributes, values, and objects would require different degrees of utilization of NOTICE, ATTOF, and MOTIVE. Changes in the verbal instructions might, also, be used to manipulate the relative difficulty of tasks that are, on the basis of a purely logical analysis, strictly ordered. If successful predictions of this type of inter-task inconsistency could be made then the search for stable developmental relations between operations would, indeed, appear to be the pursuit of a chimera.

What contribution can the information processing approach make to the elucidation of the psychological processes underlying task performance? This raises, once again, the general question of theory evaluation and the more particular one of the extent to which the model presented constitutes a valid description of the intellectual processes of children solving Kofsky's range of classificatory tasks. Formal criteria for theory evaluation have been proposed. Gregg and Simon (1967), for example, have advocated the use of universality, precision, simplicity, and flexibility as relevant choice criteria for accepting and entertaining theories. Another criterion is developmental tractability [i.e., the ease with which a model can be interpreted as both predecessor and successor of other models in a developmental sequence (Klahr & Wallace, 1970)]. Such formal criteria are, however, more relevant to assessing the relative merits of rival theories than to measuring the degree of conformity between a theoretical model and the measurable outcomes of children's thinking processes. The latter task requires the employment of empirical criteria founded on the use of experimental data.

The model presented here is compatible with existing experimental data at two levels. At the most general level, as already indicated, the most striking feature of the results of experiments dealing with concrete operational tasks is their apparent inconsistency over time, subjects, or materials (Lovell, 1960; Smedslund, 1964, 1966). The viewpoint adopted in the construction of the model—that sources of inconsistency, in the shape of the actions and interactions of MOTIVE, NOTICE, and ATTOF, are integral parts of performance—is consistent with this general picture.

At a more specific level, the model can be compared to the results obtained by Kofsky with the classification tasks. Kofsky demonstrated that the tasks could not be unidimensionally scaled. Instead of only twelve distinct response patterns on the eleven tasks presented to her 122 subjects (fail all, fail all but easiest . . . pass all but hardest, pass all) she found 63 different patterns. A summary of our analysis of the complex interrelationships among the tasks is shown in Fig. 3. The processes form a hierarchy in which an arrow running from x to y indicates that process x is directly called by process or routine y. If different subsets of the fundamental processes were unavailable, then a variety of error patterns could be evoked. Also, as indicated above, the processes depend to varying degrees upon the error prone motivational and attentional mechanisms.

A simple ordering of relative task difficulty based on the proportion of subjects passing each task indicated that, on the average, HR and VC were the most difficult and CS was the easiest. One rough measure of predicted task difficulty is the complexity of the paths leading from primary processes to task-specific routines in Fig. 3. Such a measure is consistent with Kofsky's findings except for MM, which should be the easiest of all. Part of this difference may be accounted for by the requirements for encoding of the verbal part of each task. Kofsky found that the difference in difficulty between CS and MM got smaller in increasing age groups and the difference between MM and SA reversed. In both cases a verbal encoder that increased in power with age would explain the differences.

Although such rough fits to Kofsky's empirical findings are possible, in fact our model makes no specific prediction of average overall difficulty of tasks because the effects on performance of the interactions of the

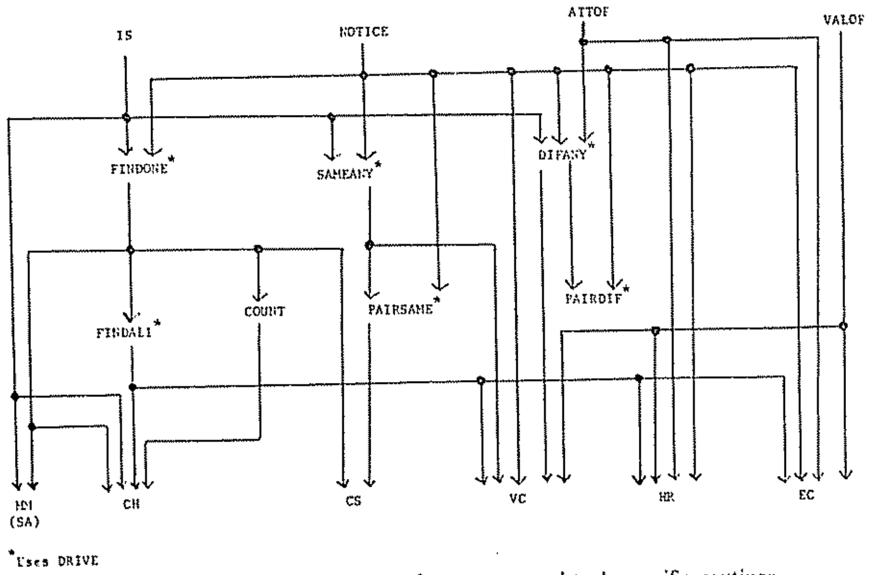

FIG. 3. Hierarchy of fundamental processes and task-specific routines.

sources of variation are unknown at this time. These sources consist of linguistic encoding, availability of fundamental processes, efficacy of the assembly process, size of the task-specific routine, number of fundamental processes in the task-specific routine, and the short-term memory requirements of the task-specific routine. Such effects can only be explored through simulation runs of a running program version of the model in Fig. 1 and Tables 1, 2, and 3.

Experiments designed specifically with the objective of assessing the extent to which the model presented above can be made to account for data are clearly necessary. There appear to be at least three promising lines of approach. The first of these involves the construction of test items corresponding to each of the fundamental processes to enable the compilation of a process profile of individual children. Versions of the model incorporating the details of the profiles could then be used to predict the individual performance of the children on the range of classificatory tasks, and the accuracy of the predictions would be assessed by comparison with actual performance data.

The compilation of process profiles could, also, be used as the initial step in an attempt to evaluate the model by means of a training study. Recall that in describing the methodological problems surrounding the transition issue it was asserted that training and acceleration studies have generated a considerable amount of confusion. This appears to have mainly arisen from a lack of specificity in theoretical statements which renders it difficult to divine their performance implications. Since the approach outlined in the present paper is deliberately designed to preserve a close link between hypothesized processes and performance, it is considered that the training experiment format could be adopted with relative impunity. The experimental group would consist of a random sample of 4 year olds, and the scope of the training provided would be determined by the process profiles of the individual children revealing the processes which they seemed to lack. Each child would, if necessary, be trained to criterion on all of the fundamental processes which the model suggests underlie successful performance on the classificatory tasks with the exception of the motivational processes subsumed by the parameter MOTIVE. No effort would be made to remove these sources of individual variation since the training is not aimed at the production of artificially perfect response patterns.

When all of the members of the experimental group have attained criterion levels of performance on the processes which they appeared to lack, they would be given Kofsky's group of classificatory tasks. The Kofsky tasks would, also, be given to groups of 7, 8, and 9 year olds. The results obtained by testing all four groups would be combined for purposes of analysis. This would enable the validity of the model to be evaluated by investigating the extent to which the individual response patterns of the trained children could be distinguished from those of the control group.

Another experimental paradigm suggested by the information processing approach is directed toward the issues raised by Smedslund (1966) in his microanalysis of concrete reasoning (which were discussed at the outset). Any one of the several tasks used as tests of concrete reasoning could be subjected to information processing analysis at the level of detail exemplified by Newell's (1966) study of adult behavior on crypt-arithmetic problems. The effect on performance of experimental manipulations such as varying the number and saliency of objects, values, and attributes in the set of physical stimuli and altering the form of the verbal instructions could be investigated.

Transition Rule

As indicated at the outset the major relevance of the work reported in this paper is to the problem of interpreting the import of interitem relations rather than to the issue of the transition rule in cognitive development. The model of cognitive performance in a narrow area which has been offered totally lacks processes which would enable it to engage in self-modification of structure. It does, however, include aspects through which such self-modifications could be introduced. ATTOF and NOTICE, for example, both possess processes which could be expanded to take on a developmental function, while additions to the repertoire of fundamental processes and modifications in the capabilities of the assembler and the encoders provide further broad areas for the introduction of self-modification.

Despite the early stage of the work, there are indications that the type of information processing analysis employed in the present paper may have considerable methodological implications for the transition issue. The main difficulty in tackling this thorny problem is the level of generality at which the major theoretical accounts of transition have been presented. It has already been pointed out that Piaget's descriptions of the functioning of equilibration, for example, do not deal with the particular mechanisms governing developmental changes or specify the conditions under which they take place. As a method of tackling the problem of transition, information processing analysis would appear to have the twin merits of commencing by focusing on specific task situations and of constantly demanding a completely specific statement of the processes which are hypothesized to underlie performance. These features should ensure that any theory of transition produced by this approach will, at least, comprise a detailed description of a mechanism which is demonstrably sufficient to account for developmental change and of its mode of functioning in a range of completely specified situations.

REFERENCES

Burstall, R. M., Collins, J. S., & Popplestone, R. S. POP-2 Papers. Edinburgh: Oliver and Boyd, 1968.

Flavell, J. H., & Wohlwill, J. F. Formal and functional aspects of cognitive development. In D. Elkind and J. H. Flavell (Eds.), *Studies in cognitive development*. New York: Oxford University Press. 1964. Pp. 67–120.

Gregg, L. W., & Simon, H. A. Process models and stochastic theories of simple concept formation. *Journal of Mathematical Psychology*. 1966, 4, 246–277.

Inhelder, B., & Piaget, J. *La genèse des structures logiques élémentaires*. Neuchâtel: Delachaux et Niestlé, 1959.

Klahr, D., & Wallace, J. G. The development of serial completion strategies: an information processing analysis. *British Journal of Psychology*. 1970, 61, 243–257.

Kofsky, E. Developmental scalogram analysis of classificatory behavior. Unpublished doctoral dissertation, University of Rochester, 1963.

Kofsky, E. A scalogram study of classificatory development. *Child Development*. 1966, 37, 191–204.

Laurendeau, M., & Pinard A. *La Pensée Causale*. Paris: Presses Univer. France. 1962.

Lovell, K., & Ogilvie, E. A study of the conservation of substance in the junior school child. *British Journal of Educational Psychology*. 1960. 30, 109–118.

Newell, A., & Simon, H. A. Computers in psychology. In R. D. Luce, R. R. Bush, and E. Galanter (Eds.), *Handbook of mathematical psychology*. Vol. I. New York: Wiley, 1963.

Newell, A. Studies in Problem Solving: Subject 3 on the Crypt-arithmetic Task Donald + Gerald=Robert. Carnegie Institute of Technology. July 1966.

Newell, A. *et al*., *Information processing language—V manual*. (2nd ed.) Prentice-Hall, 1964.

Piaget, J. Le mécanisme du développement mental et les lois du groupement des opérations. *Archives of Psychology. Genève*. 1941, 28, 215–285.

Piaget, J. Les stades du developpement intellectuel de l'enfant et de l'adolescent. In P. Osterrieth *et al*., *Le problème des stades en psychologie de l'efant*. Paris: Presses Univer. France, 1956. Pp. 33–41.

Piaget, J. Logique et equilibre dans les comportements du sujet. In L. Apostel, D. Mandelbrot, and J. Piaget *Logeque et Equilibre*. Etudes d' épistémologié génétique, 1957, 2, 27–113.

Piaget, J. In J. H. Tanner and B. Inhelder (Eds.), *Discussions on Child Development*, Vol. IV, 3–27. 77–83. London, Tavistock, 1960.

Pinard, A., & Laurendeau, M. 'Stage' in Piaget's cognitive-developmental theory: Exegesis of a concept. In D. Elkind and J. H. Flavell (Eds.), *Studies in cognitive development*. New York: Oxford University Press, 1969. Pp. 138–145.

Sigel, I. E., & Hooper, F. H. (Eds.) *Logical thinking in children*. New York: Holt. 1968.

Simon, H. A. An information processing theory of intellectual development. In W. Kessen and C. Kuhlman (Eds.), *Thought in the young child*. Lakebluff, Illinois: Child Development Publications, 1963. Pp. 150–155.

SIMON, H. A. Motivational and emotional control of cognition. *Psychological Review*, 1967, 74, 29–39.

SMEDSLUND, J. Concrete reasoning: a study of intellectual development. *Monographs of the Society for Research in Child Development*, 1964, 29 (Whole issue).

SMEDSLUND, J. Microanalysis of concrete reasoning: I. The difficulty of some combinations of addition and subtraction of one unit. *Scandinavian Journal of Psychology*, 1966, 7, 145–146. (a)

SMEDSLUND, J. Microanalysis of concrete reasoning: III. Theoretical overview. *Scandinavian Journal of Psychology*, 1966, 7, 164–167. (b)

WOHLWILL, J. F. Piaget's theory of the development of intelligence in the concrete operations period. *American Journal of Mental Deficiency Monograph Supplement*, 1966, 70, 57–83.

(Accepted May 14. 1970)