A theoretical framework is presented that distinguishes among three knowledge sources that form the basis for generative performance. The three knowledge sources, termed conceptual, procedural, and utilizational competence, were implemented as a computational model that derives plans for counting procedures. In a previous analysis, Greeno, Riley, and Gelman (1984) developed a characterization of the conceptual competence (implicit understanding of general concepts and principles) associated with the skill of counting and related conceptual competence to various models of performance. In the current work all three knowledge sources are formalized in a computer program (COUNTPLAN) that generates planning nets of counting procedures. The sufficiency of COUNTPLAN's knowledge components is demonstrated through its capacity to generate new plans for counting in novel settings from a core of conceptual competence. The utility of COUNTPLAN to facilitate the distinction between hypotheses of competence and hypotheses of performance is discussed.

INTRODUCTION

In recent years there have been numerous investigations regarding the nature of children's numerical competence (Gelman & Gallistel, 1978; Gelman & Meck, 1983; Gelman, Meck, & Merkin, 1986; Greeno, Riley, & Gelman, 1984). A major focus of this work has been the examination of young chil-
Children's performance on different counting tasks. By varying the task demands it has been possible to use performance data to make inferences about numerical competence. For example, a child will usually count a collection of toys arranged in a straight line by counting the objects in order from left to right. The leftmost toy is assigned the first number in the child's counting sequence and each additional toy is assigned the successive number. One explanation for this type of performance is that the child has learned a specific, rote counting procedure in which numbers are successively applied to objects. If this were the case, the child would not be expected to perform successfully on counting tasks requiring a different procedure. Gelman and Gallistel (1978) have shown, however, that many young children can adapt their counting procedures to handle novel constraints such as assigning the leftmost toy the number "four." In order to produce an accurate count the child must modify the "preferred" procedure. This ability suggests that the child understands the principles underlying counting procedures and this understanding is considered to be an aspect of the child's numerical competence.

This paper describes a computer program (COUNTPLAN) based on Greeno, Riley, and Gelman's (1984) formal specification of numerical competence as defined by Gelman and Gallistel (1978). The Greeno et al. (1984) model characterizes competence as implicit understanding of domain-specific and general procedural principles and is similar to Chomsky's (e.g., 1965) proposals regarding linguistic competence and Newell's (1982) discussion of functional characterizations of knowledge.

In COUNTPLAN schematic representations of domain-specific and general procedural principles form the basis for the derivation of plans for counting. These derivations take the form of planning nets (VanLehn & Brown, 1980); node-link representations that make explicit the relations between the goals of a procedure and the knowledge-enabling task performance. Consequently, COUNTPLAN provides the means to explore the nature of generative performance and to test hypotheses about relations between competence and performance. The specific goals of this research were to (a) verify that the schemata proposed by Greeno et al. (1984) represent a sufficient knowledge base to derive valid counting plans, and (b) formulate specific hypotheses about the different components of competence.

The schemata in the Greeno et al. (1984) analysis represented conceptual competence, an implicit understanding of the five counting principles defined by Gelman and Gallistel (1978). These principles are:

1. Stable order of numerals,
2. Order irrelevance of objects,
3. One-to-one correspondence between numerals and objects,
4. Cardinality,
5. Abstraction.
Since counting attributes a unique property—a number—to each entity of a set, an accurate counting procedure depends upon the "stable" or invariant application of numbers. This concept is captured in the stable-order principle. The order irrelevance principle, on the other hand, captures the fact that objects can, in principle, be counted in any order. In order to ascribe a unique number to each object, though, the stable order of numbers and the order irrelevance of objects must be coordinated. This is represented in the principle of one-to-one correspondence which specifies that the members of the counting and object sets be grouped into unique pairs. The principle of cardinality guarantees that the correct numerical property is attributed to the set, itself. Finally, the abstraction principle enables counting procedures to be applied to any set regardless of the properties of the set members.

The primary function of conceptual competence is to constrain the types of procedures that can be developed, but this alone is not sufficient to generate accurate and complete procedures. Greeno et al. (1984) hypothesized two other components of competence to account for the additional knowledge needed to generate procedures: procedural competence, knowledge of general principles about the goals and requirements of actions needed to form effective plans, and,utilizational competence, knowledge that enables the use of resources in the task setting. Figure 1 shows how these three components of competence are incorporated in COUNTPLAN.

The capacity of COUNTPLAN to produce valid plans for counting supports two basic claims. First, the schemata proposed by Greeno et al. (1984) provide sufficient conceptual knowledge for generating counting procedures. Second, a model based on an implicit understanding of principles provides a means to distinguish more clearly between hypotheses of competence and hypotheses of performance. The following discussion will focus on the nature of these claims and aspects of the implementation of COUNTPLAN. The first section describes the components of COUNTPLAN's knowledge and how these components function in a planning environment. The second section discusses the issue of generative capacity and how COUNTPLAN is able to plan procedures in novel settings. Finally, the third section discusses some ways in which COUNTPLAN contributes to an understanding of competence.

**THE STRUCTURE AND IMPLEMENTATION OF COUNTPLAN**

COUNTPLAN used the logic of planning, in that schematic representations of conceptual competence served as premises in the derivation of planning nets. The derivation process was controlled by procedural and utilization competence which integrated the schematic knowledge of general actions and constraints (action schemata) into specific counting procedures.
Planning can be viewed as a deductive process in which an initial goal is progressively refined to a level of "primitive" actions. In COUNTPLAN

Figure 1. Schematic diagram of COUNTPLAN.

1 Primitive actions are a relative concept and, therefore, a function of the level of analysis of the model. The level of analysis, in turn, is a function of the specific hypotheses about competence. In COUNTPLAN, competence is based on mathematical and procedural principles; the level of analysis and primitive actions reflects the essential characteristics of those principles. A different level of analysis and primitive actions would have been necessary if we had modelled competence in terms of perceptual and spatial operations.
this process involves searching a list of available actions for one which can satisfy the current goal. Once an action is found, the corresponding action schema is proposed as part of the evolving plan. Any preconditions of the action that cannot be satisfied in the current planning environment become goals for further planning. These goals initiate additional search for available actions. This hierarchical refinement process continues until a primitive action is reached. Next, the coherency and plausibility of the plan is verified by a set of theorem-proving techniques. This cycle of planning and verifying continues until a complete and valid plan has been generated.

The action schemata in COUNTPLAN have a structure similar to those developed by Sacerdoti (1977) in his analysis of planning. Each schema is based around a specific action and includes specifications of (a) the class of entities the action can operate on, (b) conditions to be met before the action can be performed, and (c) the outcomes of performing the action. The class of entities associated with counting actions are sets, subsets, and set members. Three types of conditions are associated with a typical action. Conditions that must be true before an action can be performed are called prerequisites. Conditions that must be true at the completion of an action are called postrequisites. Conditions that must be kept true throughout the performance of an action are called corequisites. Each action schema can also specify three types of outcomes. The primary outcome of an action is referred to as its consequence. Another type of outcome, referred to as effect, enables COUNTPLAN to keep track of other important changes in the problem setting, and as such provide a solution to the “frame problem” (e.g. Nilsson, 1980). The third type of outcome is referred to as a result. Results have the specific function of relating the abstraction of an entity to its “real-world” correlate. While it would be possible to specify the results of actions at any level of a plan, only those associated with primitive actions were considered in the current implementation of COUNTPLAN.

Planning can also be characterized as a process of constraint satisfaction in which domain principles act as constraints on an evolving plan. A planning net provides a formal characterization of how constraints and actions in the domain are integrated into a valid plan (VanLehn & Brown, 1980). A planning net represents a hierarchical decomposition of the goals and actions that comprise a plan that is consistent with domain principles. Therefore, a sequence of actions derived from a planning net for a counting task would be consistent with the constraints imposed by the mathematical principles of counting and the particular features of the task setting.

Figure 2 provides a general description of the components and relations that can be represented in a planning net. Actions are represented as rectangles with labeled links connecting them to requisite goals and outcomes. A consequence link connects an action with its primary goal. Goals that require further planning are represented as hexagons; goals that correspond to information available from features of the task setting appear as ovals.
Figure 2. Prototypical planning net.
Planning nets can also contain tests, represented as diamonds, for monitoring the effects of iterative actions.

Figure 3 shows a set of actions for counting objects arranged in a straight line that could be formulated as a process model to represent an hypothesis about performance. Counting starts by retrieving the first numeral, "one," and assigning it a property of bound. (An alternative implementation could
Figure 4. A partial planning net for the actions in Figure 3. SN and SL denote subsets of N and L, respectively. Attend goals are prerequisites of ASSIGN actions, and RETRIEVE actions have attend consequences. Some additional details are shown in Figure 8.

have been programmed with bound as a variable that would take a numeral as its value.) Next, the object at one end of the line of objects is found and assigned the property bound. This process iterates: On each cycle the next numeral beyond the current bound is retrieved from memory and becomes the new bound numeral, and the next object past the current bound object is located, becoming the newly bound object. When there are no more objects the process terminates. The last numeral to have the property bound designates the number of objects counted.

Figure 4 shows a planning net derived for these counting actions. Note that the actions represented along the bottom of Figure 4 are the actions
COMPETENCE FOR COUNTING

COUNT(L)
Prerequisites: $\text{Set of numerals } N, \text{order } (N)$
Postrequisites: $\text{equal } (N,SN), \text{bound } (SN,n)$
Consequence: $\text{number } (L)=n$.

MATCH(L,N)
Prerequisites: $\text{empty } (SL), \text{empty } (SN)$.
Corequisites: $\text{subset } (SL,L)$, where $SL=\{x:\text{tagged } (x]\}$,
$\text{subset } (SN,N)$, where $SN=\{y:\text{used } (y]\}$,
$\text{equal } (SL,SN)$.
Postrequisite: $\forall x(x \in L \Rightarrow x \in SL)$.
Consequence: $\text{equal } (L,SN)$.

KEEP-EQUAL-INCREASE(L,SL,N,SN)
Prerequisite: $\text{equal } (SL,SN)$.
Corequisites: $\forall x(x,a \in L; x \neq a \Rightarrow x \in SL \text{ before } \Leftrightarrow x \in SL \text{ after}),$
$\forall y(y,c \in N; y \neq c \Rightarrow y \in SN \text{ before } \Leftrightarrow y \in SN \text{ after}).$
Postrequisites: $\text{one-more } (L,SL,a), \text{one-more } (N,SN,c)$.
Consequence: $\text{equal } (SL,SN)$.

Figure 5. The three action schemata representing the core of conceptual competence.

shown in Figure 3. In the planning net these actions are connected to goals —displayed as hexagons. These goals are in turn related to higher order components of the plan. These higher order components embody the principles of counting and represent specific hypotheses about conceptual competence. Therefore, planning nets represent the general relationship between conceptual competence and performance.

Conceptual Competence
Conceptual competence is distributed among the three action schemata shown in Figure 5. These action schemata constitute an hypothesis about the principled understanding of counting. In order to see how this understanding is realized in COUNTPLAN, consider the actions shown in Figures 3 and 4. One aspect of cardinality, for example, its property number, is represented as a consequence of the COUNT action schema. If all the lower level plan components are properly executed the resulting procedure will yield a correct cardinal value.

The principle of one-to-one correspondence is distributed between the MATCH and KEEP-EQUAL-INCREASE schemata. MATCH establishes constraints for the construction of numerically equivalent sets whereas KEEP-EQUAL-INCREASE guarantees that this equivalence involves a correspondence between members of the two sets. More specifically, MATCH takes two sets, L and N, as inputs, specifies prerequisites for the existence of two empty sets, SL and SN, and has the consequence of making L equal to SN. Since MATCH is a very general action, the input sets can involve any denumerable entities. This generality of inputs enables COUNTPLAN to realize the principle of abstraction. In counting tasks, the input sets involve
objects (L) and the positive integers (N). During execution of a counting procedure the sets SL and SN take on members from L and N, respectively. The corequisites of MATCH constrain these subsets so that SL can only include objects that have been tagged and SN can only include numbers that have been used.

MATCH and KEEP-EQUAL-INCREASE are constrained by the goal to maintain equality between SL and SN. This constraint is a corequisite of MATCH and is a consequence of KEEP-EQUAL-INCREASE. Note, the equality of SL and SN is also a prerequisite of KEEP-EQUAL-INCREASE which means that SL and SN must be equal when KEEP-EQUAL-INCREASE is invoked and continues to be equal as long as KEEP-EQUAL-INCREASE is being applied. Further, the corequisites of KEEP-EQUAL-INCREASE (a) prevent the loss of members of SL and SN; and, (b) specify that only designated entities, a and c, can be added. Hence, the postrequisite of KEEP-EQUAL-INCREASE is the increase of SL and SN by the addition of new members from L and N, respectively. In summary, the application of MATCH and KEEP-EQUAL-INCREASE to any denumerable sets will yield a one-to-one correspondence between those sets. When one of those sets consists of an ordered set of positive integers (a counting string) the resulting one-to-one correspondence becomes the basis of a counting procedure.

One-to-one correspondence, however, is not sufficient to produce accurate counting. An additional constraint involves the stable order of members of N. One aspect of the stable order of numerals principle is represented by a prerequisite which specifies the existence of an ordered set of numerals. The final counting principle, order irrelevance of objects, on the other hand, is not apparent in any prerequisite structure: There are no actions requiring an ordered set of objects. Therefore, this principle is realized through the absence of explicit constraints and provides flexibility that enables a variety of potential counting procedures.

The discussion above demonstrates how, in COUNTPLAN, principle-based understanding is distributed among a set of schematic knowledge structures that include knowledge of the specific consequences and effects of an action as well as the dependencies between the action and its requisite conditions. This conceptual knowledge alone is not sufficient to guarantee the generation of a valid counting procedure. Additional knowledge, referred to as procedural competence, is necessary for conceptual knowledge to be realized.

**Procedural Competence**

Since COUNTPLAN was implemented as a computer program, it was possible to provide specific characterizations of the knowledge discussed by Klahr and Robinson (1981) and Brown and DeLoache (1978) in their analyses of children's planning behavior. In the current analysis this type of knowledge is referred to as procedural competence. The primary role of procedural
competence is to organize the action schemata of conceptual competence so that performance will be consistent with the domain principles. This involves understanding of general principles of planning and executing actions, including the relations among goals and actions as well as the conditions for performance. Reiterating, the important feature of procedural competence is its generality. The organization of action schemata into a valid plan is based on general concepts and relations and not on domain-specific information.

Procedural competence is divided into two major components. One component contains planning heuristics that, (a) search for actions to satisfy goals, (b) order the application of actions to avoid goal conflicts, and (c) monitor constraints generated during the course of planning. The second component is a theorem prover that, (a) tests the status of requisite conditions, (b) determines when actions can be applied, and (c) assesses the effects of actions.

**Planning Heuristics.** COUNTPLAN selects and orders actions for a specific plan on the basis of the constraints imposed by domain principles and setting features. These constraints influence planning through the processes of constraint posting, constraint propagation, and constraint satisfaction (Stefik, 1981). The action schemata COUNTPLAN selects from are shown in Figures 6 and 7.

**Figure 6.** The global action schemata for manipulating numerals and other orderable sets.


Constraints are introduced to the planning environment through the process of constraint posting. When an action is suggested as a potential plan component, each requisite condition is posted as a constraint. These constraints must be satisfied before an action can be executed. For example, the action INITIALIZE can only be applied to the first member of a set. This constraint is posted by the inclusion of INITIALIZE's prerequisite $first$.

Another aspect of constraint posting involves secondary specifications of requisite conditions. For example, KEEP-EQUAL-INCREASE has two one-more goals as postrequisites. When this action is instantiated in a counting plan, one postrequisite takes objects as an argument and the other takes numbers. Since the goal for one-more object could be satisfied by counting the same object repeatedly, a valid counting plan requires that these inputs be further constrained. The absence of such constraints could lead to a violation of the cardinality principle—the number reported would exceed the actual number of objects in the set. In fact, this type of error is often seen with young children (Gelman & Gallistel, 1978). COUNTPLAN prevents this violation of the cardinality principle by constraining the inputs of KEEP-EQUAL-INCREASE so it can apply once, and only once, to a given entity. Additional constraints are needed to insure that (a) the inputs match the specifications of the problem (e.g., count the blue chips); and, (b) once an object or number has been used, it is not lost from the used set. Together
these constraints insure that the variable element used in planning is mapped correctly to individual members of a set.

Keeping track of constraints on actions, objects, and values at different levels of plan refinement is particularly problematic for hierarchical planning systems (Sacerdoti, 1977). COUNTPLAN utilizes the process of "constraint propagation" to ensure that constraints posted at one plan level influence all other levels of the plan (Stefik, 1981). As previously mentioned, generating plans of counting procedures requires a mapping among variable elements and individual set members. That is, COUNTPLAN must link goals about sets with actions performed on individual objects. By including knowledge of set membership (cf. Barstow, 1979) and a procedure that includes loops to apply an action iteratively to members of a set, (cf. Manna & Waldinger, 1975) COUNTPLAN propagates constraints throughout the plan so the resulting procedure will apply to all members of the specified set.

This general constraint is formulated as a planning goal referred to as "for-all" (See Figure 4). The "for-all" goal links all the actions and goals that take the set L as an input, thereby constraining the procedure to include all members of the specified set. If any of the linked actions or goals is found to be unacceptable, all the connected plan components can be retracted and a new plan developed.

**Theorem Proving.** COUNTPLAN utilizes a set of theorem-proving techniques to determine the acceptability of plan components. The theorem prover generates deductive proofs in which requisite conditions serve as premises, and an action as the conclusion. If all the premises (requisite conditions) are true, then the conclusion (an action) is accepted as part of the plan.

Theorem proving is based on a list of true assertions about the planning environment. This list is generated from information contained in the problem setting, inferences generated from utilizational competence (discussed in the next section), and the effects of previous planning. When a goal matches a true assertion, it is immediately accepted as a valid component of a plan. For example, a prerequisite of COUNT is the existence of an ordered set of numbers. This goal is verified directly because the initial state of knowledge includes an ordered set of numbers. The theorem prover also satisfies prerequisites indirectly by making inferences on the basis of utilizational competence.

One function of the theorem power is to select among available actions. Planning would be relatively easy if there were only one way to satisfy a given goal. In many situations, however, there are many actions that will work. For example, COUNTPLAN has four different actions that satisfy the goal of one-more. The choice among actions is based on the deviations
of the theorem power with the main criterion being COUNTPLAN's ability to satisfy requisite conditions.

Another function of the theorem prover is to verify when goals are satisfied by actions included in the plan. Once an action is accepted as part of a plan, all its consequences, effects, and results become true assertions. For example, the consequence of MATCH is the equality of two sets, L and N. Once all of the requisite conditions for MATCH are verified, it is accepted into the plan and the proposition \( \text{equal}(L,SN) \) is added to the list of true propositions. This proposition subsequently enables the verification of any \( \text{equal}(L,SN) \) goal. In this way the consequences, effects, and results of an action immediately influence all parts of the plan. As long as an assertion remains true, any goal corresponding to that assertion will be verified. As soon as an assertion is falsified, it is deleted from the list, and subsequent goals requiring that assertion cannot be verified. In the case of corequisites it becomes necessary to implement actions that will return those assertions to the list.

The theorem power also handles the verification of existential and universal logical conjunctions necessary to plan actions involving sets. Since these expressions are logically complex, COUNTPLAN decomposes them into component propositions. The theorem prover then treats the component propositions as discussed above. In addition, logical conjunctions are verified on the basis of the effect of each proposition on the truth of the entire conjunction.

**Utilizational Competence**

Planning a valid procedure depends upon an accurate representation of features of the task setting. The primary setting features in counting tasks include physical characteristics (e.g., type, color) of the objects and their physical arrangement (e.g., order, location). Since COUNTPLAN's conceptual knowledge of counting is quite general, it is necessary to bridge between this knowledge and specific setting features. This bridge is constructed by utilizational competence which generates inferences about how setting features can be used to satisfy the prerequisites of global actions.

COUNTPLAN generates inferences about the setting at several different levels. An initial set of inferences is generated directly from the physical features of the setting. Subsequent inference generation depends upon the nature of the initial inferences and other task features. As a result, the pattern of inferences is different for each setting. For example, a requirement of counting is the partitioning of objects into two subsets: one containing objects that have been counted and one containing objects that have not been counted. When objects are arranged in a straight line, this partitioning can be achieved by counting from one end to the other and noting the most recently counted object. This object and those preceding it are members of the counted subset (SL) and all other objects are members of the uncounted
subset of L. In this particular setting COUNTPLAN infers a first object and a next relation between each successive object. This information is then used to demarcate the two subsets. In addition, this information enables COUNTPLAN to verify the necessary conditions for including INITIALIZE and INCREMENT into the plan.

Utilizational competence also enables the partitioning of the set of counting numbers. The stable order principle constrains the counting numbers to be used in a prescribed manner: beginning with the number "1" and incrementing successively by one. The counting numbers, therefore, can be partitioned in the same way as objects arranged in a straight line. That is, a set of numbers can be represented as a straight line (a number line) with a first element and next relations linking each subsequent number. This not only helps satisfy the constraints of the stable order principle but also facilitates maintenance of the one-to-one correspondence and cardinality principles.

**GENERATIVE KNOWLEDGE**

Generative performance, one of the hallmarks of competence, is dependent upon the capacity to perform in ways that have not been explicitly learned. Two aspects of COUNTPLAN’s generative performance were assessed: (a) flexibility, the capacity to perform in a variety of settings; and, (b) robustness, the capacity to modify plans to accommodate novel constraints.

**Flexibility**

COUNTPLAN’s flexibility is primarily a function of (a) the core of conceptual competence that reflects an implicit understanding of counting principles; and, (b) utilizational competence that enables the application of that conceptual core to settings which vary with respect to the mobility and arrangement of objects. Flexibility was assessed by generating plans in settings involving immovable objects arranged in a straight line, immovable objects that were scattered, and movable objects that were scattered. The setting with movable objects arranged in a straight line was not analyzed because of its functional equivalence to the "scattered but movable" setting.

Figure 8 shows the portion of a planning net that is derived from the core of conceptual competence. Every plan, regardless of the setting, is based on this core of conceptual competence. This core is comprised of the COUNT, MATCH, and KEEP-EQUAL-INCREASE action schemata; the schemata that represent the principles of counting. Any procedure that conforms to the set of constraints specified by these schemata will be a valid counting procedure.

The existence of this core of knowledge raises two important issues. First, the capacity to generate a variety of valid counting procedures from a core of domain knowledge provides evidence for the completeness of this knowledge component. Second, since these three action schemata represent
knowledge at a high level of generality, evidence of their existence would not necessarily be apparent in the overt aspects of counting performance. In general, evidence of the existence of this conceptual core must be inferred on the basis of performance on tasks like those developed by Gelman and Gallistel (1978).

The planning net derived from the conceptual core terminates with two one-more goals; one to increase the set of numbers and another to increase the set of objects. The one-more goal involving numbers can always be
satisfied by either INITIALIZE or INCREMENT because the set of counting numbers satisfies the requisite conditions of those actions. The primary difference in the plans derived in different settings involves the sequence of actions and goals that satisfy the goal of one-more object.

There are four global actions capable of satisfying the goal of one-more object: INITIALIZE, INCREMENT, PUT-IN, and ADD-MARK. The term global is used to denote the fact that these actions are not performed directly but must be further decomposed to the primitive RETRIEVE action. The major differences among these actions involves the nature of this decomposition as shown in Figure 9.

RETRIEVE produces the same effect regardless of its associated global action; the object in attention \( a \) becomes a member of the counted set (SL). The specification of this effect does differ depending upon the postrequisite of the global action. With INITIALIZE and INCREMENT, \( a \) becomes a member of SL by movement of a perceptual bound. PUT-IN and ADD-MARK denote membership in SL by a change-in-location and a mark (object-tagged), respectively.

As previously mentioned, selection of global actions is based on utilization competence which uses setting features to satisfy the prerequisites of actions. The most salient setting feature with regard to partitioning sets is the arrangement of objects in L. If the objects in L are arranged in a straight line, COUNTPLAN infers that L is ordered. Both INITIALIZE and INCREMENT can apply directly to ordered sets, whereas PUT-IN and ADD-MARK require the specification of additional conditions. The choice between INITIALIZE and INCREMENT is based on the distinction that the former applies only to the first member of a set, whereas the latter applies to all but the first set member. COUNTPLAN satisfies these prerequisites by inferring that ordered sets contain a first object and next relations connecting each subsequent object. In addition, the properties of first and next enable COUNTPLAN to specify conditions for starting and stopping the counting procedure.

When objects are not arranged in a straight line, COUNTPLAN is unable to satisfy the prerequisites of INITIALIZE and INCREMENT, and so a choice must be made between PUT-IN and ADD-MARK. The primary prerequisite of ADD-MARK is the availability of a marker, whereas PUT-IN requires movable objects. If the objects are scattered, cannot be moved, and no marker is available, COUNTPLAN must use some other physical or spatial feature to distinguish between the counted and uncounted objects.

Regardless of which global action is selected, COUNTPLAN must determine how to maintain the distinction between counted (SL) and uncounted (L) objects. This requires knowledge which identifies SL as an empty subset of L, along with a list of physical properties that distinguish between set members in the different settings. When objects are arranged in a straight line, SL is identified as the part of L to the left of the perceptual bound.
Figure 9. The decomposition of global actions into primitive actions.

When objects are scattered and movable, SL is identified as being in a different location than L (location-2 versus location-1). Finally, when objects are scattered and immovable, the members of SL are identified as being tagged or marked. This information is used to satisfy the postrequisite of the global action, and in doing so maintains one-to-one correspondence be-
tween numbers and objects and ensures the correct cardinal value for L. If the distinction between L and SL is not clearly specified, there is an increased likelihood one or more of the counting principles will be violated, resulting in an incorrect counting procedure.

Robustness

Perhaps the most powerful demonstration of the generative capacity of principle-based understanding involves the successful modification of an existing plan to satisfy novel constraints. A system that can demonstrate this capacity is said to be robust. COUNTPLAN's robustness involves: (a) the core of conceptual competence described in the previous section, and (b) procedural competence. The previous sections describe how COUNTPLAN generates plans for action by matching entailments of the counting principles with constraints imposed by the setting. What happens when COUNTPLAN encounters novel constraints—those that cannot be readily matched to existing knowledge structures? Do the three components of competence have the capacity to derive plans which satisfy these constraints?

To answer these questions COUNTPLAN generated plans for the "make this the n" tasks used by Gelman and Gallistel (1978). These tasks require an individual to count a linear array of objects with the added constraint that a particular object be tagged with a number different from the one that is normally assigned in a left-to-right counting procedure. For example, the individual might be asked to tag the leftmost object in the line with the numeral "four." Figure 10 shows the four types of "make this the n" or constrained counting tasks presented to COUNTPLAN. Two of these tasks involve a straight-line setting and two involve scattered settings. These tasks could not be achieved by any of the plans previously generated by COUNTPLAN, and therefore required the derivation of new plans. The new plans included the same components as the previous plans but the components had to be ordered in unique ways. The flow diagram in Figure 11 provides an overview of COUNTPLAN's generation procedure for these new plans.

In each constrained task the derivation process begins as it does for the standard "straight-line" task, and in each case the numbers are applied in the standard order. The difference across settings involves the sequence of actions COUNTPLAN uses to match numbers to objects. During the initial stages of planning, the designated object and number are tagged as "special." This information is propagated through the plan to "constrain" actions that operate on sets containing a special entity. In other words, COUNTPLAN recognizes that the setting differs from previous settings it has encountered, notes these differences, and takes general steps to facilitate future planning. For example, in the event a valid plan cannot be developed, the "constrained" actions can be located and modified. It is important to note, however, that COUNTPLAN does not respond to the existence of special entities
Setting: Straight-line/movable or non-movable

perceptual bound or marker

\[ \begin{array}{cccccc}
  & X & X & X & X & X & X \\
\end{array} \]

Setting: Scattered/movable

Location-1 Location-2

\[ \begin{array}{ccc}
  X & X & X \\
  X \\
\end{array} \]

Setting: Scattered/non-movable

\[ \begin{array}{ccc}
  X & marker & X \\
  X \\
  X & X \\
\end{array} \]

Figure 10. A characterization of the different settings and set demarcations.

by instantiating a preformulated sequence of actions. Rather, COUNTPLAN approaches this task as it would any other task—by invoking its general planning heuristics.

A critical aspect of COUNTPLAN’s derivation procedure is the existence of appropriate goals. In the case of the constrained tasks, this involves a goal to “match up” the two special entities which is generated directly from the task instructions. At the outset of planning, COUNTPLAN has no particular method to achieve this goal since it has never been encountered before. Rather, planning proceeds until one of the “special” entities becomes the input of a primitive RETRIEVE action. At this point COUNTPLAN detects a conflict between goals. The goal of matching the special entities can be satisfied by planning to retrieve the other special entity, but this conflicts with the goal of counting the entire set. In response to this situation, COUNTPLAN generates a new plan that achieves both goals without violating the principles of counting.
Figure 11. Counting procedures for the "make this the n" tasks.

Although each of the four constrained settings requires a different plan, there are certain features that remain constant. For example, in each setting it is necessary to suspend primitive actions that apply to special entities in order to prevent the execution of actions that would later require "un-
doing.” Once the primitive actions are suspended, COUNTPLAN determines which special entity has been encountered. If the special entity is an object, COUNTPLAN recognizes that it has not yet reached the special number and is therefore unable to count the special object. In this event, COUNTPLAN utilizes the flexibility afforded by the order indifference principle by planning to SKIP the special object. Since SKIP has the unique effect of moving the perceptual bound past an object without making it a member of SL, the principle of one-to-one correspondence is not violated. If the first object is skipped, INITIALIZE can still be applied to the next object, otherwise INCREMENT will be applied. In either case a plan is made to RETRIEVE the “next” object in the line which results in the movement of the perceptual bound, inclusion of that object in SL, and the verification of the goal one-more. This sequence of events violates the principle of one-to-one correspondence which is noted through the postrequisites of KEEP-EQUAL-INCREASE. To correct this imbalance a sequence of actions is planned to obtain one-more number. If that number is not designated as “special,” COUNTPLAN continues to pursue the goal of counting all the members of L. The object that was skipped will not become important until the special number is retrieved.

When the special number is reached, COUNTPLAN must determine how to match it to the object that was skipped. Neither INITIALIZE nor INCREMENT will apply because of their ordering constraints. The most likely action to satisfy the existing goal is ADD-MARK, since its only prerequisite is a means of marking counted objects. In most settings ADD-MARK is eliminated from consideration because no marker is available. When COUNTPLAN reaches an impass, however, it reassesses the applicability of previously discarded actions. In the case of the “make this the n” tasks, COUNTPLAN infers that the special object can be identified by means of a “perceptual marker.” This inference enables ADD-MARK to be used for that one object. Once ADD-MARK has been planned, the necessary conditions exist to match up the special entities. ADD-MARK has the effect of moving the bound past the special number and making it a member of the numbers used (SN). It also has the effect of making both the special object, and the object in attention, members of SL by moving the perceptual bound. This “compound” effect causes a corequisite imbalance of KEEP-EQUAL-INCREASE which requires planning of another action on numbers to maintain one-to-one correspondence.

1 In its current form, COUNTPLAN is based on operations that begin with objects. A perceptual process plans for the selection of an object and then a corresponding action is planned for application to the numbers. A more sophisticated scheme would be based on the initial goal and begin by planning actions on numbers. While this scheme has not been implemented, it appears that an “invariance”-noticing scheme such as the one proposed by Resnick and Neches (1984) could be applied to COUNTPLAN, and create a shift from the current “perceptual strategy” to a more efficient “conceptual strategy.”
A different set of constraints applies if the special number is reached before the special object. The stable order of number principle prevents COUNTPLAN from skipping over a number. Since objects are retrieved before numbers, COUNTPLAN plans a sequence of actions in which ADD-MARK is used to RETRIEVE the special object. As with the previous case, this satisfies the goal of matching the two special entities, but also creates a violation of the one-to-one principle. This situation is handled in the same way as previously discussed. A different problem exists, however, and that has to do with dealing with the special object when it is encountered again. COUNTPLAN solves this by planning to SKIP objects that are already members of SL. Since SKIP moves the perceptual bound without affecting set membership, there will be no violation of the one-to-one principle.

It should be evident from the discussion that the derivation of new plans to meet novel constraints requires considerable effort and resources. In addition to the memory demands associated with the special number, special object, numbers counted, and numbers used, this task also requires the coordination and effects monitoring of several ongoing activities. The capacity to derive plans to meet novel constraints addresses two issues. First, the knowledge embodied in COUNTPLAN is sufficient to meet the requirements of generating valid counting plans. Second, COUNTPLAN's generative capacity provides support for a distributed characterization of competence.

DISCUSSION

Implementing COUNTPLAN required several features of a planning system which extend the hypotheses of Greeno et al. (1984) concerning understanding general principles about procedures. One such feature involves postponed commitment, an aspect of many planning systems in artificial intelligence (e.g., Sacerdoti, 1977; Stefik, 1981). Actions are initially assigned a tentative status in the planning net. The final decision to include an action is withheld until it can be determined that all its requirements can be satisfied. Another feature of COUNTPLAN involves the ability to construct plans with branches and loops. This required the use of methods to recognize different conditions and to recognize that goals for manipulating sets can be achieved by applying operators to all members of a set. The solutions to these problems of procedural competence relied on methods that have been utilized in work on automatic programming (e.g., Barstow, 1979; Manna & Waldinger, 1975).

The specification of procedural and utilizational competence in COUNTPLAN helped to clarify some properties of conceptual competence. One point of clarification involves the distinction between understanding concepts and principles in a domain and understanding that allows those concepts to be applied in a specific task setting. In the current work the distinction is made by specifying competence for procedures of counting in a variety of
settings. The components of competence that are invariant across all the settings called "counting" constitute a core of concepts and principles for counting; those that vary represent competence for applying the counting concepts. The distinction is more definite and precise when procedural and utilizational competencies are specified in detail, as in the current analysis, than in the earlier analyses when those components were treated informally.

The implementation of COUNTPLAN has also clarified what is required for a generative knowledge base (competence) to meet criteria of flexibility and robustness. Flexibility is the ability to apply domain concepts and principles across situations. This requires general understanding of the concepts and principles represented in the schemata for "core" competence in the domain of counting. The application of domain knowledge also requires understanding of the procedural requirements, constraints, and resources in novel settings. These factors are represented as aspects of procedural and utilizational competence.

Robustness is the ability to modify procedures to satisfy novel constraints and still remain consistent with the domain principles. The current analysis suggests that accommodation of novel constraints in counting tasks is highly dependent upon procedural competence. In COUNTPLAN this required the addition of capabilities for recognizing and handling exceptional cases to the procedural competence that was sufficient to generate plans in standard counting situations.

One way to illustrate how COUNTPLAN clarifies the distinction between competence and performance is to examine the different levels of detail involved in the characterization of information in a cognitive task. Figure 12 provides a general framework for characterizing a cognitive task. Each level of description in Figure 12 is a derivative of the description at the next lower level. As a descriptive characterization there is, however, no direct mapping between levels (c.f. Newell, 1982). For example, the knowledge/competence level is a product of the contents of the declarative and procedural representations, but there is no direct mapping between knowledge encoded procedurally (representation level), and knowledge of procedures (knowledge/competence level). Since each level of description shown in Figure 12 has been discussed in various contexts, it is important to discuss how they relate to the proposed model.

Representation Level
The issue of different types of knowledge has been extensively discussed by philosophers and psychologists alike. For example, Ryle (1949) distinguished between knowing "what" and knowing "how." Connolly and Bruner (1974) made a similar distinction in their discussion of the difference between knowing "that" and knowing "how." More recently, this distinction has become an integral part of psychological theories such as Anderson's (1976,
### Performance Level

- **Meaningful**
  - General procedures that are based on conceptual and procedural competence.
- **Rote**
  - Problem specific procedures that involve some procedural competence but little or no conceptual competence.

### Knowledge/Competence Level

- **Utilizational**
  - Specific knowledge that enables the coordination of setting features and abstract goals.
- **Conceptual**
  - General principles and concepts of the task domain.
- **Procedural**
  - General principles that are involved in planning and executing procedures.

### Representation Level

- **Declarative**
  - Context free semantic networks containing information that can be shared among productions.
- **Procedural**
  - Context dependent productions that operate on declarative representations of information.

**Figure 12.** Levels of description for cognitive skills.

1983) ACT model. ACT distinguishes between two forms of knowledge representation: declarative knowledge which is essentially knowing "that," and procedural knowledge or knowing "how." Declarative knowledge is represented in context-free semantic networks, whereas procedural knowledge is represented in the form of production rules. Knowledge is initially encoded in declarative form, but over the course of time this declarative knowledge can be transformed into new production rules.

This variable encoding of knowledge provides a mechanism that avoids the problem of making absolute distinctions between knowledge structures and cognitive processes. Encoding knowledge procedurally enables the development of models that are not restricted to the more traditional distinction between computer programs as cognitive processes and data structures as knowledge representations. While this analogy is often helpful, it can also be the source of theoretical confusion. For example, some theorists claim that improvements in performance are marked by structural changes, while others argue that these same improvements are marked by the addition or modification of cognitive processes. The primary problem for both these claims is the lack of substantial supporting evidence. As Newell (1972) points out, a major problem with attempts to distinguish processes from structures is the lack of a stable reference point. In general, attempts to iso-
late performance factors which appear stable over time have not been success-
ful. Many of the consistencies in performance have been shown to be a
function of the task rather than factors associated with the problem solver.

While there are several issues of cognition that may eventually depend
upon a distinction between processes and structures, the need for such a dis-
tinction does not directly affect characterizations of understanding. For
example, a process-oriented characterization of understanding might em-
phasize the processes that transform knowledge structures, whereas a struc-
tural characterization of understanding might emphasize the knowledge
states and propositions associated with understanding. Clearly, these two
views cannot be completely separated. Any adequate characterization of
understanding should include formal specifications of both knowledge
structures and cognitive processes.

Knowledge/Competence Level
The knowledge/competence level represents a substantive reformulation of
the more general information encoded at the representation level. This re-
formulation enables an analysis of the structural properties of knowledge,
as well as the content of those structures. One way these properties can be
analyzed involves a distributed view of competence. In COUNTPLAN com-
petence is represented in the form of principles and concepts in the domain
(Greeno et al., 1984) along with inference rules that relate this conceptual
competence to particular tasks' demands.

A distributed view of competence directly addresses two issues regarding
the relationship between competence and performance. One issue is how the
contents of knowledge structures become realized as an organized sequence
of behavior. This problem has been discussed for decades, but during the
last 20 years there has been considerable progress towards the development
of integrated theories. These advances were foreshadowed by Miller, Gal-
ant, and Pribram (1960), who discussed the importance of understanding
the relationship between competence and performance by demonstrating
the necessity of both domain-specific knowledge structures, along with
general and domain-specific processes which operate on those structures.

A second issue has to do with the confusion between a lack of compe-
tence, on the one hand, and performance factors which prevent the full real-
ization of competence on the other. One reason for this confusion has been
an emphasis on action-based characterizations of competence. Connolly
and Bruner (1974), for example, equate competence with the ability to make

---

3 While there may be some instances where it is important to distinguish between knowledge
and competence, we do not feel that it is necessary in the current analysis, and we have used the
terms "competence" and "knowledge" interchangeably here. Some further distinctions are
discussed by Gelman and Greeno (in press).
changes and adapt to changes in the environment. An alternative approach has been to characterize competence as the ability to perform correctly, and to characterize performance as the factors that can produce errors. Unfortunately, it is not always possible to distinguish between errors that result from a lack of competence and errors that result from a performance deficiency (Flavell, 1970).

On the basis of the current analyses, the term performance refers to knowledge structures and cognitive processes used in performing a task, and competence refers to understanding of the general concepts and principles used in constructing or acquiring procedures (Greene et al., 1984). This distinction is similar to Miller and Chomsky's (1963) characterization of competence as a formal system with infinite capabilities, and performance as the expression of those capabilities by a system with prescribed limitations. This view is also consistent with Newell's (1972) characterization of competence as an abstract description of a system's capabilities.

Performance Level
Competence models are commonly criticized for being idealized descriptions of behavior which fail to account for the types of performance actually observed (Pylyshyn, 1973). We have developed a model of competence that provides a framework in which to analyze how certain factors might prevent the realization of competence.

Different degrees of understanding at the knowledge/competence level can lead to different types of performance. The distinction between rote and generative performance depicted in Figure 12 represents the outcome of two relatively extreme forms of understanding. This is similar to the Gestalt psychologists' distinction between rote and meaningful learning. In this conception correct performance is not always a function of understanding; many problem solutions are achieved through the rote application of a well learned sequence of operators (Dunker, 1945; Katona, 1940; Wertheimer, 1945).

The extreme form of rote performance involves little more than the application of problem-specific procedures. While rote performance can be based on a certain amount of procedural competence, it is not supported by the components of utilizational and conceptual competence. The inclusion of procedural competence in rote performance is motivated by the belief that even algorithmic behavior depends upon some understanding of procedures. (This, of course, is not the case with computer programs which can execute algorithms without knowledge of those procedures, e.g., Simon & Kotovsky, 1963.)

The increased understanding that differentiates meaningful from rote performance involves all three components of competence. Conceptual competence contributes an understanding of domain principles, procedural
competence contributes an understanding of procedural principles, and
utilizational competence enables the application of these principles in
specific settings. The knowledge structures representing this increased under-
standing include more features of the problem structure and setting, as well
as more specific connections between the setting features and the actions to
be performed in the procedure. The additional structure allows performance
to become more consistent across a wide variety of problem situations.

The theoretical framework presented above provides a means for develop-
ing and testing hypotheses of competence which are distinct from hypoth-
eses of performance. This framework could potentially be extended to in-
clude performance factors (e.g., memory, attention) that would enable more
specific analyses of the interplay between competence and performance. In
addition, this framework could be incorporated into a tutoring system to
analyze task performance and diagnosis particular competence-based deficits.

Original Submission Date: April 21, 1987.

REFERENCES

Press.
Bruner (Eds.), The growth of competence. New York: Academic.
(Eds.), Advances in child development and behavior (Vol. 5). New York: Academic.
Gelman, R., & Gallistel, C.R. (1978). The child’s understanding of number. Cambridge, MA:
Harvard University Press.
Gelman, R., & Greeno, J.G. (1989). On the nature of competence: Principles for understand-
ing in a domain. In L.B. Resnick (Ed.), Knowing, learning, and instruction: Essays in
343–359.
Development, 1, 1–29.
ing. Cognitive Psychology, 16, 94–143.
Klahr, D., & Robinson, M. (1981). Formal assessment of problem-solving and planning pro-
Intelligence, 6, 175–208.


