Recursion in Problem Solving*

by

Charlie McKavanagh
Faculty of Education
Griffith University
Nathan, Queensland

Presented at the 1992 Joint Conference of the
Australian Association for Research in Education
and the New Zealand Association for Research in Education

Deakin University, Geelong
22-26 November, 1992

* This research was supported by the Faculty of Education, Griffith University and its Centre for Skill Formation Research and Development, by Queensland Rail, and by the University of Queensland. The cooperation of Queensland Rail and especially its Training and Development staff is also gratefully acknowledged.

Recursion in Problem Solving

by

Charlie McKavanagh
Faculty of Education
Griffith University
Nathan, Queensland

Presented at the 1992 Conference of the
Australian Association for Research in Education
Geelong, Victoria

Abstract

A recursive model of problem solving is developed and methods for testing the model discussed. Recursion is a generative process which produces output through ordered stepping. The problem-solving model has a knowledge component in the form of directed acyclic networks and a recursive process which manages problem solving by forward and backward stepping through the maze of structural knowledge.

By reinterpreting knowledge structures recursively, it is possible to neatly explain the creation of new links. Also, tree hierarchies and linear sequences are shown to be substructures of directed acyclic networks, making the model consistent with many other schema theories.

Introduction

In this paper, a recursive model of problem-solving behaviour and cognitive

processing in a simulated work setting is developed. A means of testing the model with trainee and expert train drivers using computer controlled train simulators is proposed. The aim is to develop an effective cognitive model in order to predict how instructional practices will facilitate the development of expertise in train drivers. This research is part of a PhD study with Professor Glen Evans at the University of Queensland.

Recursion

The cognitive model developed is based on the concept of recursion drawn from the computer programming and artificial intelligence literature. There is no doubt that recursion is an elusive concept (Apostel, 1991; Vitale, 1989), but it is shown here, in line with Apostel and Vitale, that a degree of unification of its many explicitly stated and implied facets can be achieved. Two types of recursive processes are identified -- one is here called generalised recursion (and is similar to tail recursion of Kurland & Pea, 1985 and Abelson, Sussman & Sussman, 1985), and the other is known in computer programming as true or embedded recursion (Kessler & Anderson, 1986; Kurland & Pea, 1985). It is postulated that generalised recursion is involved in transforming and linking cognitive structures during learning and that embedded recursion controls goal-directed behaviour in problem solving. These two types of recursion are discussed in the sections which follow and are then applied to the problem-solving model in later sections of this paper.

Generalised Recursion

Generalised recursion is a generative process which transforms and links elements through an ordered series of steps to form a recursive structure. The linked elements may be called nodes and become joined by relations into a chain of the recursive structure being generated. For example, in the context of Logo, a programming language for children, Papert (1980) quotes this recursive riddle, "If you have two wishes what is the second? (Two more wishes)" (p. 74). The recursive process (the act of wishing) links elements (wishes), thereby generating an ordered structure (a list of wishes) through a series of steps (two wishes at a time).

In an earlier publication Papert gives this definition of recursiveness: "[the property of recursion being not the repetition of the same act as such, but the repetition of an act that is at the same time the same act and a different one]" (Papert, 1960, p. 123, English translation quoted by Vitale, 1989). Vitale (1989) gives a similar definition of recursiveness, emphasising the stepping from level to level and the features of permanence (in Papert's example above, all nodes are wishes) and change (at least one wish at each level may be different). Vitale also emphasises that there are persistent global features present at all levels (in Papert's example, the desire for something) as well as local features which may be unique to a level (each different wish).

Another aspect of recursion is its self-referential nature, or as Mueller and Page (1988) state, "Recursive definitions use the term being defined as part of the definition" (p. 65). For instance, in Papert's example, the act of making wishes involves making wishes. This self-referencing inherent in recursion is emphasised by Leron & Zazkis (1986), McCarthy, Abrahams, Edwards, Hart & Levin (1962), Poundstone, (1987), Vitale (1989) and many others.

Thus, the main characteristics of recursion may be summarised as self-referencing and level-stepping with global permanence, but with local change leading to a linked chain of nodes. The nodes are connected by relations. One implication is that the chain of nodes developed is an ordered (or directed) structure. Some nodes are closer to the starting point than are others derived further down the line. These characteristics also imply that an indefinite and perhaps infinite number of nodes may be linked, as the following example illustrates.

Burnham and Hall (1985) give a pair of rules coded in the computer language Prolog for a recursive definition of ancestor which is applicable to persons and, as will be argued later in this paper, cognitive structures as well:

> "ancestor (X,Y) :- parent (X,Y).
> ancestor (X,Y) :- parent (X,Z), ancestor (Z,Y)." (p. 43).

Where the assignment symbol ":-" separates a left hand from a right hand side of each rule. Each rule says to set the left hand side to be true if the right hand side is already true. "X", "Y" and "Z" are distinct instances for each rule. The instances within each parenthesis correspond on a one-to-one basis with other instances of the same rule.

This code may be quite literally translated into English as:

> X is an ancestor of Y, if X is a parent of Y; and simultaneously,
> X is an ancestor of Y, if X is a parent of Z, and Z is an ancestor of Y.

A less literal translation is that an ancestor may be defined as a parent and as a parent of an ancestor.

The code is recursive, since there is self-referencing -- "ancestor" is defined in terms of itself and level-stepping (between generations) is also apparent, with ancestor as the relation linking the levels (generations). In this example, the chain derived from the recursive relations is the ancestry which is ordered along a time sequence. By defining the relation in terms of ancestors and parents, it is implied that the direction of processing will be from the present into the past (i.e. searching for ancestors backwards through time). However, the presence of an ancestor-parent relation also implies a reciprocal descendent-child relation which

would be searched forwards in time. The reciprocal relation for a descendent may be stated in English as:

a descendent is a child of a node, or a descendent is the child of a descendent

where child is a directional relation between pairs of adjacent nodes.

There is self-reference in this recursive relation, with descendent being defined in terms of a descendent. Also, one node (the younger) of each adjacent pair is one order of magnitude (level, step or generation) lower than the other (older) node. Thus level-stepping is present. The relation is the directional link between adjacent nodes at two different levels. The older node gives rise to the younger node. The younger node of the pair is designated as the child node. This structure is also a hierarchy, since the older levels are more inclusive than the lower ones. The descendent-child relationship is illustrated in Figure 1, where an arrow head points to each child node.

In Figure 1, while the descendent-child relation is made explicit, an ancestor-parent relation is implied by the tails of the arrows. The node at the tail of an arrow indicates the ancestor of the node at the arrow head. The ancestor-parent relation is followed in the reverse direction from that of the descendent-child relation. This pair of reciprocal relations is analogous to the active and passive forms of a statement. For instance, the sentence, "Mary caught the ball." is in active voice and contains the same information as the passive, "The ball was caught by Mary." However, the information-bearing words are in reverse order in the two sentences. The pair contains the same information, but the ordering of elements is different. In the discussion that follows, reference is made to both reciprocal forms of the descendent-child and ancestor-parent relation.

Mueller and Page (1988) point out that, "Recursion makes it possible for definitions with only a finite number of terms to describe objects of unbounded size" (p. 60). The above examples of recursion accentuate the potentially infinite extent of chained, recursively generated structures, because there is neither an explicitly defined nor an implied termination. It is implied (by not being otherwise specified) that recursion does not terminate and that there is a single line of descent (and ancestry) from each node. However, by specifying additional qualifiers, the property of termination may be explicitly incorporated into the relation. Termination of a chain of nodes may occur if a qualification that it is not necessary for a node to have a child is introduced.

Other qualifiers also effect the shape of the recursive structure. By specifying that a node may have more than one child, branching of chains which gives rise to branching hierarchies or trees, becomes possible. Also, for the purpose of the theory in this paper, a new term confluence is

introduced for the case where two (or more) nodes give rise to a single child node. Confluence leads to joining of chains. When both branching and confluence are present along directed paths, the structure becomes a network, or more specifically, a directed graph (Deo, 1974; Fahlman, 1979).

Termination, branching and confluence provide variations on linkage patterns, so that linear chains, trees and networks are all possible structures. Bratko (1986) regards lists and trees as recursive structures. It is shown here that networks, through the addition of the confluence qualifier on relations, are a generalisation of lists and trees. This is in line with Stubbs and Webre who state that, "A graph structure is a generalization of a hierarchical structure which, in turn, is a generalization of a linear structure." (p. 400). Thus, it is argued that lists, trees and networks are recursive structures and that, depending on the allowed qualifiers of the relations, different linked structures are produced, as shown in Table 1 and diagrammed in Figure 1.

Table 1: Qualifiers on recursion and the corresponding linked structures produced

Linked structure	Qualifiers on recursion
list or linear hierarchy:	no more than one child; no more than one parent (i.e. no branching and no confluence);
tree or branching hierarchy:	none, one or more children; no more than one parent (i.e. branching, but no confluence);
network or directed graph:	none, one or more children; none, one or more parents (i.e. both branching and confluence).

While all nodes are similar (that is, they are of the same general class or type), they are not identical. Nodes may have differing connectivity. The number of relations with adjacent nodes contributes to differences in levels and chains to which the node belongs. For example, in a tree, there

is one node (the root) which has no parent and some nodes (the leaves) have no children. As will be discussed in more detail later in this paper, variations in recursive structure connections bear similarities to postulated memory structures of networks of nodes.

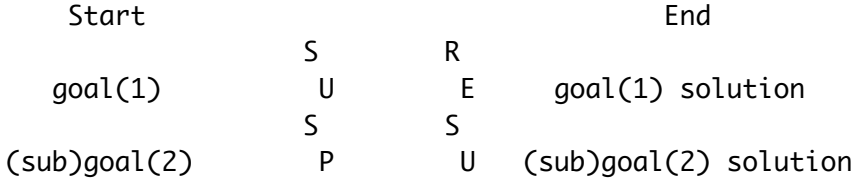
Embedded Recursion

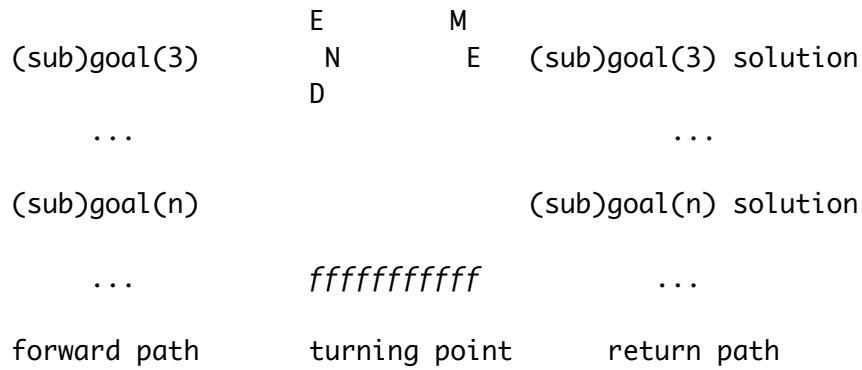
Embedded recursion (often called true recursion or simply recursion) is a more restricted concept than that of generalised recursion. Embedded recursion has all of the recursive properties noted above, including self-referencing and level-stepping; local and global aspects; and permanence with change. However, its execution is constrained and its prime role is the control of processing rather than the production of a linked chain structure. It is characterised by "a chain of deferred operations" (Abelson, Sussman & Sussman, 1985, p. 32). It consists of controlled forward level-stepping along a chain of activation to a definite turning point, followed by level-stepping returning back along the same path.

The depth of the recursion is the number of levels stepped through to the turning point. At each step forward, a decision is made as to whether or not the result currently under consideration can be evaluated (for example, has a trivial case been reached?). If the result can be provided immediately ("terminating case") then the recursion has reached its turning point; otherwise forward level-stepping continues ("recursive case") (Allen, 1978; Pirolli, 1986). To ensure eventual closure of the recursion, forward level stepping must result in progressive simplification from previous levels. Apostel (1991) refers to this as practical recursion.

At each forward step along the chain of activation towards the turning point, there is deferment of current activity pending the availability of intermediate results on the return from the next level along. It is at the turning point that intermediate results are first obtained, and, as the return phase begins, these results are provided to enable completion of the most recently suspended level. Progressively with each return step, intermediate results are made available as suspended activities are resumed. By this means, the activity that was initiated and suspended at the starting level is the last one to be carried to completion, by what is often referred to in computer programming as first in, last out processing. The forwards and backwards stepping are depicted in Figure 2.

Figure 2: Stepping forwards to a turning point and back to a solution during execution of embedded recursion.





Vitale (1989) offers a restricted definition of recursion, in terms of the control of problem solving behaviour, which is consistent with the discussion above and with other authors' use of embedded recursion.

"It is proposed that a restricted notion of 'recursion' could be usefully defined, entailing:

(a) that the attitude of the subject, with respect to the definition of a notion, the solution of a problem, the answer to a question, etc., should contain a measure of suspended attention, deferring in a way the final restructuring of the definition, solution, answer, etc., to the completion of a downward and then upward spiralling path;

(b) that the spiralling path should be describable by the dialectical coexistence of permanence (the path, global because relying on the various steps) and change (the pitch of the spiral, local because defined -- and possibly changing -- at every turn)." (p. 272).

From Vitale's definition, expanded and clarified by Apostel (1991), a number of inter-related characteristics can be abstracted and framed in terms of the similarities and differences between general and embedded recursion. The characteristics derivable from generalised recursion and which apply to both general and embedded recursion are: self-referencing and level-stepping; permanence with change; and local and global aspects of the entity existing concurrently. In Table 2, these characteristics are summarised and illustrated with reference to problem solving.

Table 2: Characteristics of Generalised Recursion

i) Self-Referencing

The problem is viewed through different (sub)goal perspectives, each of which is a partial image of the original problem. A goal or any subgoal can be viewed as at level n, and referred to as (sub)goal(n).

ii) Level-Stepping

There is a progressive focusing on (sub)goals in a stepwise fashion from n=1 to a depth determined dynamically during problem solving.

iii) Permanence
The same problem is acted on through several guises ((sub)goals).

iv) Change
(Sub)goals differ from one another in their focus and comprehensiveness.

v) Local and Global Aspects
Both local (subgoal) and global (goal) aspects of the problem are present simultaneously.

Additional characteristics derived from the restricted definition of embedded recursion are: stepping along a path to a definite turning point; progressive simplification, suspended attention and deferment of sub-processes during forward stepping; and progressive resolution during backward stepping. These are illustrated in Table 3 with reference to problem solving.

Table 3: Additional Characteristics of Embedded Recursion

vi) Forward and Backward Path
In successful problem solving, there is a forward stepping path ("top" to "bottom") towards a turning point, linked to and followed by a backward stepping path ("bottom" to "top").

vii) Suspended Attention
Work on goal(n) is delayed during forward stepping in order to work on and resolve a more specific (sub)goal(n+1).

viii) Deferred Solution
Solutions to (sub)goals, which are deferred in turn on stepping forward, are fulfilled in turn on stepping back after the turning point.

ix) Simplification
Each forward level stepping results in a simpler subgoal being worked on.

x) Finiteness
A goal is reached through a finite number of steps forward and backward through levels. All subgoals are also resolved in a finite number of steps.

It follows that, if the characteristics summarised in Table 2 and 3 can be identified in a given process, then that process is one of embedded recursion. This will enable testing of the recursive nature of the cognitive model which is developed in the next section.

Cognitive Model

The cognitive model developed in this section has three components -- a knowledge component in the form of a recursive structure, a learning process in the form of generalised recursion, and a control process in the form of embedded recursion. The control process manages cognitive processing in goal-directed activities by using knowledge where available. If critical knowledge is available in an organised form in the knowledge structure, then activity will proceed in an ordered and predictable way. However, if an impasse is reached because critical knowledge is not immediately available, then the learning process is invoked to try to link, transform or gather information to overcome the impasse. If the impasse is overcome, learning takes place, the learning process terminates and the control process resumes orderly progression towards the goal. This overview is illustrated in Figure 3. Failure to overcome the impasse leads to abandonment of the goal. Each of the three components -- knowledge structures, learning process and control process -- is examined in turn in the next sections.

Structural Knowledge Component

The knowledge component of the model is a network of nodes. Nodes are joined to one another along directional chains by recursive relations in the form of ancestor-child links. This network is a recursive structure, forming linear chains, branches and confluences linking the nodes. Each node is a schema of the general form developed by Rumelhart (1980) and Rumelhart and Ortony (1977). Each schema also incorporates goals, conditions and actions which bear similarities to the GOMS information-processing model of Card, Moran and Newell (1980; 1983) and to the productions of Anderson (1982, 1987).

Rumelhart and Ortony (1977) emphasised the role of schemata in representing concepts and connections among concepts in memory. This view was later broadened to emphasise goals and actions as well as concepts and the central role of schemata in cognition:

> "[Schemata] ... are the building blocks of cognition. They are the fundamental elements upon which all information processing depends. Schemata are employed in the process of interpreting sensory data (both linguistic and nonlinguistic), in retrieving information from memory, in organizing actions, in determining goals and subgoals, in allocating resources, and, generally, in guiding the flow of processing in the system." (Rumelhart 1980, pp. 33-34).

It is this broader view of schemata, which encompasses not only the perception, storage and retrieval of conceptual knowledge but also how that knowledge is used, that is represented in the current model. In the model, each schema is a node of a knowledge network.

The proposed model for representing cognitive structures in memory is based on recursively generated relations between pairs of nodes with cross linkages. Each link between a pair of nodes is a directional one, giving rise to a knowledge structure in the form of a directed network. This directional relation is referred to in this paper as a descendent-child relation and is accompanied by a reciprocal ancestor-child relation linking nodes in the reverse direction. Branching, where a parent node gives rise to two or more child nodes, produces trees or hierarchies within the network. The names of components of the networks is not coincidental as there are strong analogies between these networks and genetic inheritance in living organisms. Confluence, where chains are joined, gives rise to multiple inheritance whereby a single node may have many lines of ancestors. This is consistent with Fahlman's (1979) view that the "... [semantic] hierarchy is a tangled one. That means simply that a node may have more than one immediate parent, as well as an arbitrarily large number of offspring. To put it another way, the tree branches upwards as well as downwards." (Fahlman, 1979, p. 19). This view is shared by Way (1991). Hierarchies are substructures of networks, so that networks and hierarchies coexist and contain schemata as their nodes.

Many current schema models of cognition present a static view in which concepts and their interrelationships are emphasised. Procedures may be attached to some of these concepts. For example, Chi and Rees (1983) state that "Networks are very natural representations for factual, declarative knowledge, but they can also be used to represent procedural knowledge..." (p. 87). Chi and Rees call this "procedural attachment". The model developed here, which is primarily concerned with cognitive processing, does the reverse. Goals and actions are seen as the core of the schemata and some of these schemata may have "conceptual attachments".

In the proposed model, the schema has variable slots (spaces) which normally hold usual or default situations or events. The slots are: a single goal to be fulfilled; the actions which must be executed to fulfil the goal; the conditions which must be met before the actions can be performed; and the subgoals which would need to be pursued to solve the problem if the conditions cannot be met immediately. Any subgoal is more specific than the goal to which it is attached. Fulfilment of the subgoal will provide new information which may enable the goal to also be fulfilled. Some of the slots of the schema may be empty or have default values. The default value for the subgoal slot is to invoke the recursive learning process. These slots are similar to the components of the GOMS (Goals, Operations, Methods and Selection rules) model of Card, Moran and Newell (1980; 1983). In the current model, however, the components are seen as packaged together within a schema and related by an IF... THEN... ELSE... construct as will be discussed in the section on the recursive control process. Schemata are related to other schemata through goal-subgoal linkages. Taken together, the condition and action slots correspond to Anderson's (1982, 1987) productions which Chi and Rees (1983) describe as, "Each ... [production] consists of a condition side and an

action side and they are often informally represented as if=then pairs. If the condition side matches the contents of short-term memory then the action is taken. The condition side contains constants, which may must match specific items, or variables, which can match general classes of items. The actions are generally modifications of memory. This match-action structure makes production rules ideal for representing procedural knowledge.... Also goals and subgoals can be set." (p. 87).

Since the goals and subgoals of these schemata may be linked along a chain in the direction of increasingly finer detail, they may be viewed as being along an abstract-to-concrete dimension. In this case the subgoal would be seen at a more concrete level than its goal. Thus, it can be said that a goal may contain a more specific subgoal which may contain another more specific subgoal and so on. In this conceptualisation, the distinction between goals and subgoals becomes blurred. However, in the discussion which follows, "goal" is used for the starting point for a problem and "subgoal" for any goal structure at a finer or more concrete level of detail.

The functions of the variables in the control process during problem solving are summarised in Table 4.

Table 4: Functions of Schema Variables During Problem Solving

Variable Slots
Goal
(IF) Conditions
(THEN) Actions
(ELSE) Subgoals
Functions

Specifies the desired outcome and acts as an index to the schema

These conditions are tested when the schema is invoked

If the test returns true, then the actions are carried out and recursion unwinds

If the test returns false, then recursion continues to a deeper level by invoking the subgoals
The goal-subgoal chaining may not continue in a strictly linear fashion indefinitely. For example, a chain would terminate if the conditions were satisfied and the actions executed, and if no subgoal were available (the default value), then an impasse would develop. The non availability of a subgoal signals the activation of the learning process, discussed in the next section.

Learning Process

In the context of this paper, Chi and Rees' (1983) meaning of learning is accepted:

> "By learning we mean both the acquisition and structuring of knowledge....We propose that, while learning begins with the acquisition of declarative facts, it is knowledge structures which are the internal embodiment of competence. ... When one attains some new level of competence, it is because a new knowledge structure has been formed, perhaps by combining old ones, perhaps by creating an analog of an old one, perhaps in some other way." (p. 72)..

The learning process of the model conforms to the notion of generalised recursion developed above. The learning process, activated at an impasse, generates new schemata, builds new links among existing schemata and modifies the contents of schema slots. It generates new schemata (with goal, condition, action and subgoal slots), adds or modifies values to schema slots and links existing schemata in response to outside stimuli and internal cognitive processing such as remembering, inferring and deducing. It is invoked as the default option in the subgoal slot of a memory schema when that slot is accessed by the recursive control mechanism. The form of the recursive learning process is based on the chunking learning mechanism of SOAR (Laird, Newell & Rosenbloom, 1987; Laird, Rosenbloom & Newell, 1986a,b), which Newell (1990) sees as a candidate for a universal theory of cognition. In this model (Newell, 1990), learning occurs when an impasse is resolved. A chunk is built for each result involved in resolving the impasse. The action of the chunk is built from the result. The conditions of the chunk are built from the working-memory elements that existed before the impasse occurred and that led to the result. In the current model a

chunk is regarded as a schema (i.e. a node) within the knowledge network.

It is also argued that during problem solving, new knowledge acquired through reasoning, perceptual input and feedback, is added to the existing knowledge network. In the process, new relations linking existing schemata (at the nodes) may be constructed. New elements may also be added to existing schemata. The indeterminacy of the recursive process that builds the recursive knowledge structure provides flexibility. The indeterminacy and flexibility of recursion can provide dynamic extension to the size and shape of a linked recursive structure to adapt to changing demands. A recursively generated structure is malleable at its boundaries and can be extended (or reduced) dynamically without interference with the core of the structure. The structural chains may be extended or may branch or join with other chains in confluence, so that the knowledge structure may expand dynamically and thus account for learning and adaptability involved in cognitive processing.

Control Process

The third component of the model is the control process in the form of embedded recursion. It is postulated that the control process is recursive, universal, self-controlling, and relatively simple. The difficulty of problem solving is attributed to the large search space of what is known, to the organisational complexity of this knowledge network, and particularly to components missing from the knowledge network. In most instances the network will be incomplete with respect to the problem solution, at least at the time of problem presentation or recognition. It is argued that, in successful problem solving, a linear path with steps from problem givens to problem goal is activated and/or completed within the knowledge network and that actions are executed at steps along this path.

This view is in line with VanLehn (1989) who states, "... the overall process of schema-driven problem solving is recursive, in that executing one small part of the process can potentially cause complete, recursive invocation of the problem-solving process." (p. 548). Smith, Greeno and Vitolo (1989) have a similar view of the planning process in problem solving, "Planning can be viewed as a deductive process in which an initial goal is progressively refined to a level of 'primitive' actions". (p. 186). This view is based on the earlier work of Newell and Simon (1972) who hypothezied that planning requires a representation of the problem that distinguishes between essential and inessential problem features. From the essential features a plan is constructed as a sequence of subgoals and execution of the plan requires accomplishing these subgoals, including working out details corresponding to inessential features of the solution. This is similar to Sacerdoti (1977) who sees problem solving as an evolving plan with hierarchical refinement through a cycle of planning and verifying. During the process, domain principles and global goals related to higher order components constrain the evolving plan. Global actions

correspond to sets of more elementary actions that can be performed directly in the problem environment. Organized problem solving occurs when an individual constructs a sequence of global actions that lead to the main problem goal, and then, in turn, works out the details of each of the global actions.

During successful planning for problem solving, it is argued, the needed knowledge is constructed and combined with existing knowledge to elaborate the existing schemata in ways that make them similar to the planning nets of Greeno, Riley and Gelman (1984) and Leinhardt and Greeno (1986). VanLehn and Brown (1980) introduced the term planning net which they see as a directed graph or network, similar to the use of the term network in the current model.

"Planning nets are directed graphs. The nodes of the net represent plans, and the links represent planning inferences. That is, each node of the net stands for a flowchart containing a mixture of primitive actions and subgoals to be expanded. Two nodes are linked only if the application of some constraint or heuristic to one plan results in the other plan. The link is labeled with the planning rule that causes the change." (VanLehn & Brown, 1980, p. 114).

The recursive control process of the model has an IF... THEN... ELSE... pattern of the form shown below, where the underlined terms are variables, which correspond to slots in the schemata as discussed above.

```
SOLVE goal
        IF conditions are satisfied
                THEN execute corresponding actions
        ELSE SOLVE subgoals
```

It is argued here that the embedded recursive process, manages problem solving by controlled stepping through subgoal hierarchies. This is illustrated in Figure 4 which shows the control process in action. In this example, successful problem solving starts and ends at n=1.

Figure 4: Controlled processing of a Problem with One Goal and Several Subgoals in Procession

SOLVE (sub)goal(n)

IF the solution for (sub)goal(n) is available:

THEN: Exit from (sub)goal(n) and provide solution for use by the calling procedure.

ELSE:

SOLVE (sub)goal(n+1)

IF the solution for (sub)goal(n+1) is available:

THEN: Exit from (sub)goal(n+1) and provide solution for use by the calling procedure.

ELSE:

SOLVE (sub)goal(n+2).

IF the solution for (sub)goal(n+2) is available:

THEN: Exit from (sub)goal(n+2) and provide solution for use by the calling procedure.

ELSE:

SOLVE (sub)goal(n+3).

Etc. until a subgoal is resolved and control moves stepwise back through the levels.

... ...

Level 4 ˛ ˛ Level 4

Level 3 ˛˛ Level 3

Level 2 ˛˛ Level 2

Level 1 ˛˛ Level 1
Several aspects of the roles of the variables during execution of this process should be noted. Firstly, the process provides a coordinated link between the slots of the schemata and the variable values to fill the slots. This filling of slots by a particular set of values is conceived as what is known in computer science as binding (Allen, 1978). By this means, the recursive control process becomes instantiated. Secondly, when the control procedure becomes bound to a particular schema, the conditions will be tested and either the actions to solve the problem carried out (if the test returns true) or the process will be invoked recursively through the subgoal (if the test returns false). The control process will step through the chain of available goals and subgoals until some blockage occurs or until the action to solve the subgoal can be executed. When a subgoal is satisfied the recursion begins to step back along the chain from subgoal to goal. Thirdly, if the chain from original goal through to a subgoal which can be executed is unbroken, then the problem can be solved. If not learning must take place before the problem could be solved. Finally, the form of "Solve (sub)goal(n)" and "Solve (sub)goal(n+1)" are the same (both are recursive control procedures with an IF... THEN... ELSE... pattern), but the variables of each will contain different values for the goal, subgoals and other variables.

As explained by Bratko (1986) in relation to hierarchical lists and trees in the Prolog programming language, recursive processes complement recursive structures:

"One reason why recursion so naturally applies to defining relations in Prolog is that objects themselves often have recursive structure. Lists and trees are such objects. A list is either empty (boundary case) or has a head and a tail that is itself a list (general case). A binary tree is either empty (boundary case) or it has a root and two subtrees that are

themselves binary trees (general case). Therefore, to process a whole non-empty tree, we must do something with the root, and process the subtrees." (Bratko, 1986, p. 182).

This is reinforced by Abelson, Sussman, & Sussman (1985) who state:

"Recursion is a natural tool for dealing with tree structures, since we can often reduce operations on trees to operations on their branches, which reduce in turn to operations on the branches of the branches, and so on, until we reach the leaves of the tree." (p. 97).

This complementarity of structure and process means that the control process is controlled by the knowledge in the structure it accesses. In this sense, the control process is self-controlling. In another sense, the control is automatic -- from the information it uses. The slots of the IF... THEN... ELSE... pattern of the control process are instantiated by the values in the schema in memory with respect to a particular goal. The instantiation of the IF... THEN... allows the action to proceed (as in a production of Anderson, 1982) if the conditions are fulfilled. Otherwise the control process is directed to the next subgoal in the chain by the value of the ELSE... slot, if available. The controlling process, in essence, becomes controlled by the instantiated values in the memory structure (schema) it is accessing. If a subgoal is needed but is not available, then the default value of initiating the learning process is invoked. The completion of the learning process will revert control to the point from which learning was invoked. If the learning process has been successful in supplying the missing information, the control process will act on it and begin returning towards the initial goal. If learning has not been successfully accomplished, then the current goal may be abandoned.

In essence, the proposed model has schemata that are the source of particular instances for the goal, condition, action and subgoal variables which are accessed by the control process. The current values of conditions, actions and subgoals are determined (if needed) by binding to the corresponding slots in a single schema in memory or, if these are not available, by using blank or default values. The particular schema to be referenced is determined by the goal, which thus serves as an index between the variables and their binding values. If all goals, conditions, actions and subgoals are available through the ordered chain of schemata, then problem solving will be successful. If any link is missing, then either the problem cannot be solved by the current line of inquiry or some new learning must take place to fill gaps in the chain. The learning process is invoked to try to fill any gaps and thus overcome blockages.

The recursive control keeps broad goals in focus while working on detailed expectations. By accessing the knowledge structure and using instances stored in a particular schema to determine what happens next, the control process is in fact being controlled by the schema it is accessing. By this means, the learning process is also invoked if insufficient information is available, and if this too fails to produce a result, problem solving is

terminated. Thus, by interaction among the recursive control process, the recursive learning process and recursive knowledge structures, the entire process is self-controlling and self-terminating.

Empirical Testing of the Model

In this section the setting for the empirical study, the hypotheses to be tested and the means for testing them are raised. Discussion on the adequacy of the testing procedures is invited.

The Setting for the Empirical Work

The empirical work to test the model is being undertaken at the Rockhampton Driver Training Centre of Queensland Rail. The Centre has about twelve full-time instructors who prepare and teach introductory courses to trainee train drivers (here regarded as novices) and in-service courses to registered drivers (here regarded as experts). Facilities at the Centre include standard classrooms, computer-based learning materials (developed at the Centre) and a number of different train simulators. The simulators are used extensively for instruction in the "Train Handling" module of the trainee driver training course, and for monitoring driver response to different track and train conditions, equipment faults and environmental hazards.

The focus of the study is novice and expert drivers using one of the Diesel Electric Locomotive (DEL) 2400 train simulators. The DEL 2400 class of locomotive is used mostly for freight haulage in country areas of Queensland, such as the huge coal trains from the inland mining centres to the coastal ports. The simulator can be set up for a "run" over given sections of track. The simulator consists of a mock up of an engine cabin attached to a VAX computer running IIT Research Institute train simulation software. The driver sits inside the cabin and the instructor or assessor sits outside at a console from where the simulated run is monitored and controlled. The driver and instructor or assessor can communicate using microphones and speakers.

The train simulator uses a video projector which displays sections of actual Queensland train track to the driver who responds to the images, cabin noise, instrument dials and task instructions by manipulating the train controls. The track images, noise levels and instrument dials change under computer control in response to the driver's actions and the simulated motion of the train. Every few seconds, the computer monitors and calculates values for a range of variables, including those displayed on the cabin instruments, those associated with driver actions and train performance indicators. Some of these variables are listed in Table 5.

Table 5: Selected Variables Output During a Run on a DEL 2400 Train Simulator

Sensory Information Normally Available to the Driver

Immediate Indicators of Driver Action on Power and Braking Systems

Train Performance Indicators

Flags and Templates for Selected Events

Visual frame of the track

Throttle/ dynamic brake setting

Average train stretch (Steady state draft)

Heavy braking*

Speed

Brake pipe pressure on first wagon (i.e. degree of air brake application)

Maximum train stretch (Peak draft)

Air brakes before dynamic brakes*

Track speed limit*

Brake cylinder pressure (i.e. degree of locomotive brake application)

Average train compression (Steady state buff)

Locomotive wheels slipping*

Power use/ generation

Brake on/ off on rear wagon (i.e. brake action propagation to end of train)

Maximum train compression (Peak buff)

Fast throttle changes*

*These variables are reported only when changes occur.
(Other variables are reported every few seconds.)

The results are stored on computer file and can be printed out at the end of a run for review and analysis. Most of the results for the variables are also displayed dynamically in graphical form on a monitor normally only available to the instructor outside the cabin. The monitor graphics can be video-recorded and later played back. "Think aloud" talk by the driver can be recorded onto the video tape as well to provide a synchronised protocol for analysis. If requested, summary train and driver performance statistics and track characteristics can be calculated and printed.

Once the train is in motion, there are two main control systems used by the driver. These are the power system (consisting of throttle and dynamic brake) and the air brake system (consisting of wagon brakes and independent locomotive brakes). The driver manipulates these systems by levers. The graphic displays and run printouts from the simulator show these changes as readings on four variables every few seconds (See Table 5).

Manipulations to slow the train are particularly complex, with interaction effects of the dynamic, wagon and locomotive brakes. The length, weight and speed of the train, and the condition and grade (slope) of the track all effect the braking capacity. Also, with the wagon brakes, it takes a considerable time for the brakes of the rear wagon to engage after brake application, as the propagation must travel through the pressure pipe along the length of the train. Furthermore, once air pressure is depleted, further air brake application is ineffective.

In driving and in monitoring the train and their own performance, drivers use information from cabin noise, their view of the track and the many cabin gauges. They also depend on their memory for information about rules and regulations and, most importantly, for what lies ahead on the track. This "road knowledge" is particularly critical, since braking times are slow (it can take 1.6 Km to stop a 1.6 Km train from a speed of 80 Km/hr) and signal and speed limit signs usually occur only at the locations from which they become effective. Printed "road maps" are not available. Advanced warning about the locations of signals and signs is often not given along the tracks. Drivers must rely on their memory to know what lies ahead. They must constantly use this knowledge to plan and execute actions now that will have their full effect after the next corner or over

the next crest.

Even though there are only a few levers to manipulate when in motion, it is clear that drivers are engaged in a great deal of problem solving and related cognitive activity, such as planning, predicting and monitoring, especially under varying track conditions. That driving requires a high degree of cognitive processing is further supported by the fact that even very experienced drivers cannot play chess at the same time as they are driving on track with changing characteristics.

The amount of cognitive processing in which drivers engage, the high degree of control that is possible over initial conditions, and the range and detail of intermediate and output variables makes this setting ideal for empirical studies into problem solving processes. For example, in experiments based on the simulators, independent variables which could be manipulated include, instructions to the driver, the section of track to be covered, the number of wagons, the length of the train (up to 1.6 Km), the weight of the train (up to 8,000 tonnes) and equipment faults. Driver actions, track characteristics and train dynamics can be monitored.

Hypotheses to be Tested

A hypothesis to test for the application of each of the two recursive processes is proposed as well as a hypothesis which links the two processes through the synchronisation of control process and the recursive structures in memory. These hypotheses are testable through observing problem solving in action and by probing to explicate memory structures involved. In addition, a hypothesis is formulated which predicts the effects of training on problem solving performance. This hypothesis is testable by an educational experiment using a control group of subjects and a parallel training intervention group, coupled with pretesting and posttesting of both groups.

The recursive control process hypothesis may be stated as:

If a recursive control process operates during problem solving, then protocol traces will reveal process with IF... THEN... ELSE patterns as in the model, with variable slots for goal, conditions, actions and subgoals.

The learning process hypothesis may be stated as:

If schemata are generated by a recursive process during learning, then both new and existing schemata, as revealed by goal mapping and interview techniques, will show goal-subgoal stepwise linkage of schemata along directional chains. Also, since goals may have several subgoals and several superordinate goals, these chains may be interlinked into networks in the form of directed graphs.

The synchronisation hypothesis may be stated as:

If the variable slots of the control process are filled by reference to the components of the schema structures in memory (binding), then protocol traces of problem solving, with probes that also reveal the cognitive structures being utilised, will show synchronisation of process with structure.

If these hypotheses relating to the recursive process of the proposed model are not confirmed through structured observations, then the model will be revised. However, if the recursive process model is confirmed, then its validity can be further investigated by predicting and evaluating outcomes of instruction designed to complement the model. The hypothesis, which can be tested by an educational experiment, may be stated as:

For a given problem task, if the four components (goal, conditions, actions and subgoals) for each goal and subgoal are available in memory (as revealed by testing or interview), then the problem can be solved. If any component is missing and cannot be derived from other information available, then supplying it through appropriate instruction or prompting will enable problem solving.

The data sources required to evaluate each of the four hypotheses above are examined in the next section.

Data Sources

The first three hypotheses, dealing with the control process, the learning processes and synchronisation between these, can be tested through a set of observations on one group of subjects. Data collection would centre on protocols derived from think aloud transcripts of subjects solving a number of problems while driving which are linked to sensory and motor data accompanying the problem solving. Probes in the form of structured interviews coupled with a visual recording technique similar to concept mapping (Novak & Gowin, 1984) would provide further evidence of the linkage among knowledge structures and components.

The fourth hypothesis is at a more detailed level and would not be pursued until confirmatory evidence for the first three hypotheses was obtained. Furthermore, the educational experiment to test this hypothesis requires pre- and posttesting. The tests need to be developed and validated and would be based on detailed knowledge gained from the observations. Therefore, test development would be undertaken after collecting the observational data and before conducting the experiment. A sequence for the empirical work is shown in Table 6, which also incorporates details of subjects and data sources.

Table 6: Sequencing, Subject Groups and Data Sources for the Empirical Work

Order

Subject Groups

Hypotheses and Tests

Data Sources

Notes

1.

A few experts and novices

Control process

Protocols derived from think aloud transcripts linked to sensory and motor data

Each subject is given the same task

2.

Same group as above

Learning process

Reuse above protocols, plus interviews and goal mapping

Each subject is given the same task

3.

Same group as above

Synchronisation

Reuse all the above data

Each subject is given the same task

4.

Subjects who are not part of the experiment which follows

Piloting of pretests and posttests

Paper and pencil tests or tests based on simulated tasks

Development of pretests and posttests for the experimental investigation

5.

Novices in two treatment groups (control and experimental)

Prediction

Pretests, posttests and instructional or prompting sequences

Testing, instruction and prompting done in groups or individually while solving problems

Collection of Descriptive Data

Protocols will form the basis of the descriptive data for testing the hypotheses relating to recursive control and learning processes and the synchronisation of control with cognitive structures. Protocol analysis has been a widely used in investigations of problem solving and has been extensively reviewed by Ericsson and Simon (1980, 1984), Olson, Duffy & Mack (1984) and extended by Larkin and Rainard (1984) and Robertson (1990) and others. Chi and Rees (1983) especially promote the use of protocols

for investigating production systems used in cognitive processing:

"...[Production systems] explicitly take the contents of short-term memory into account. This means they can handle attentional processes quite naturally. Also, because it is the contents of this memory that are generally 'seen' by such techniques as protocol analysis (Ericsson and Simon, 1980), comparing models with experimental data may be facilitated." (Chi & Rees, 1983, p. 88?)

Productions are part of the model proposed here, and so the recommendations of Chi and Rees are followed.

Subjects will be three train drivers with experience ranging from novice to expert. The drivers will be selected according to reputation. The task will be to drive a given train for about twenty minutes over a section of track which has highly variable characteristics. Drivers will be asked to perform a clearly specified driving task, using a DEL 2400 train simulator. All drivers will drive over the same section of track (road) and in the same direction. Protocols will be collected from the output of the simulators (every few seconds; see Table 5) synchronised with voice recordings of the drivers as they "think aloud" about what they are doing as they plan their trip, carry out their plans and monitor the performance of the train in relation to their knowledge of the road. Drivers will be given experience in "thinking aloud" during trial runs prior to data collection.

The track chosen for the task will be one that is not familiar to any of the drivers. Drivers will "learn the road" for this track without actually driving on it. Following the advice of Larkin and Rainard (1984), instructions will be explicit and focused on the areas of interest, namely goals, plans and actions. Also in line with Larkin and Rainard, a list of permissible phrases for the experimenter will be used. The experimenter is to use these and no others during collection of protocols to control both the amount and content of experimenter talk. While many researchers (see Nisbett & Wilson, 1977; Olson, Duffy & Mack, 1984 for reviews) caution against the use of retrospective verbal reports as data for investigating cognitive processing, this technique is used in this study to probe further the meaning of protocols after completion of protocol collection, as recommended by Larkin and Rainard (1984).

Analysis of Descriptive Data

Preliminary analysis of protocol data will centre on the classification of statements into the proposed components of the cognitive structure, namely, goals, conditions, actions and subgoals. Time series derived from this classification will be examined for recursive patterns of embedded goal structures and stepping, with synchronisation between structure and process.

Experimental Study

The hypothesis which predicts the effects of training on problem solving performance is to be tested by an educational experiment. The hypothesis, formulated above, is restated here:

For a given problem task, if the four components (goal, conditions, actions and subgoals) for each goal and subgoal are available in memory (as revealed by testing or interview), then the problem can be solved. If any component is missing and cannot be derived from other information available, then supplying it through appropriate instruction or prompting will enable problem solving.

The experiment uses a control group of subjects and a parallel training intervention group, coupled with pretesting and posttesting of both groups.

Conclusion

In this paper, a recursive model for the control of problem solving has been presented. Hypotheses arising from this model have been proposed and empirical studies to test the hypotheses developed. Discussion is invited as to the adequacy of the proposed empirical studies.

The theoretical advances in this theses are fundamental to understanding the structuring of knowledge and the control of cognitive processing. The theory has significant implications for the representation of knowledge, the ways in which knowledge is used in problem solving and a better understanding of the dynamics of goal directed cognitive processing. It will contribute to a more coherent and integrated theory of cognition and will open the way for more refined artificial intelligence and computer-based expert systems. It also has significant applications in teaching and learning across many subject areas, but particularly for problem centred learning by apprentices in TAFE Colleges and other areas of vocational education. Directions for extending research are raised, especially with regard to the inheritance of characteristics from one level to another in a knowledge network and how different kinds of linkages between knowledge categories may be formed and maintained.

References

Abelson, H., Sussman, G. J., & Sussman, J. (1985). Structure and interpretation of computer programs. Cambridge: MA. MIT Press.

Allen, J. (1978). Anatomy of LISP. New York: McGraw-Hill.

Anderson, J. R. (1982). Acquisition of cognitive skill. Psychological Review, 89 (4), 369-406.

Anderson, J. R. (1987). Skill acquisition: Compilation of weak-method solutions. Psychological Review, 94, 192-210.

Apostel, L. (1991). Elusive recursiveness - The necessity of a dynamic and pragmatic approach: A response to Vitale. New Ideas in Psychology, 9(3), 367-373.

Bratko, I. (1986). Prolog programming for artificial intelligence. Wokingham: Addison-Wesley.

Burnham, W. D., & Hall, A. R. (1985). Prolog programming and applications. Houndmills: Macmillan Educational.

Card, S.K., Moran, T. P., & Newell, A. (1980). Computer text editing: An information-processing analysis of a routine cognitive skill. Cognitive Psychology, 12, 32-74.

Card, S.K., Moran, T. P., & Newell, A. (1983). The psychology of human-computer interaction. Hillsdale, NJ: Erlbaum.

Chi, M. T. H., & Rees, E. T. (1983). A learning framework for development. In M. T. H. Chi (Ed.), Contributions to human development: Trends in memory development research (Vol. 9, 00. 71-107). Basel: Karger.

Deo, N. (1974). Graph theory with applications to engineering and computer science. Englewood Cliffs, NJ: Prentice-Hall.

Ericsson, K. A., & Simon, H. A. (1980). Verbal reports as data. Psychological Review, 87(3), 215-251.

Ericsson, K. A., & Simon, H. A. (1984). Protocol analysis: Verbal reports as data. Cambridge, MA: MIT Press.

Fahlman, S. E. (1979). NETL: A system for representing and using real-world knowledge. Cambridge: MA. MIT Press.

Greeno, J. G., & Riley, M. S. (1987). Processes and development of understanding. In F. E. Weinert & R. H. Kluwe (Eds.), Metacognition, motivation, and understanding (chap. 10, pp. 289-313). Hillsdale, NJ: Erlbaum.

Greeno, J. G., Riley, M. S., & Gelman, R. (1984). Conceptual competence and children's counting. Cognitive Psychology, 16, 94-143.

Kessler, C., & Anderson, J. R. (1986). Learning flow of control: Recursive and iterative procedures. Human Computer Interaction, 2(2), 135-166.

Kurland, D. M., & Pea, R. D. (1985). Children's mental models of

recursive LOGO programs. Journal of Educational Computing Research, 1, 235-244.

Laird, J. E., Newell, A., & Rosenbloom, P. S. (1987). Soar: An architecture for general intelligence. Artificial Intelligence, 33, 1-64.

Laird, J. E., Rosenbloom, P. S., & Newell, A. (1986a). Universal subgoaling and chunking : The automatic generation and learning of goal hierarchies. Boston : Kluwer Academic Publishers.

Laird, J. E., Rosenbloom, P. S., & Newell, A. (1986b). Chunking in Soar: The anatomy of a general learning mechanism. Machine Learning, 1(1), 11-46.

Larkin, J. H., & Rainard, B. (1984). A research methodology for studying how people think. Journal of Research in Science Teaching, 21(3), 235-254.

Leinhardt, G., & Greeno, J. G. (1986). The cognitive skill of teaching. Journal of Educational Psychology, 78(2), 75-95.

Leron, U., & Zazkis, R. (1986). Mathematical induction and computational recursion. For the Learning of Mathematics, 6(2), 25-28.

McCarthy, J., Abrahams, P. W., Edwards, D. J., Hart, T. P., & Levin, M. I. (1962). LISP 1.5 programmer's manual. Cambridge, MA: MIT Press.

Mueller, R. A., & Page, R. L. (1988). Symbolic computing with Lisp and Prolog. New York: John Wiley.

Newell, A. (1980). Reasoning, problem-solving and decision processes: The problem space as a fundamental category. In R. Nickerson (Ed.), Attention and performance VIII (pp. 693-718). Hillsdale, NJ: Erlbaum.

Newell, A. (1990). Unified theories of cognition. Cambridge, MA: Harvard University Press.

Newell, A., & Simon, H. A. (1972). Human problem solving. Englewood Cliffs, NJ: Prentice Hall.

Nisbett, R. E., & Wilson, T. D. (1977). Telling more than we can know: Verbal reports on mental processes. Psychological Review, 84(3), 231-259.

Novak, J. D., & Gowin, D. B. (1984). Learning how to learn. Cambridge, MA: Cambridge University Press.

Olson, G. M., Duffy, S. A., & Mack, R. L. (1984). Thinking-out-loud

as a method for studying real-time comprehension processes. In D. E. Kieras & M. A. Just (Eds), New methods in reading comprehension research (Papers presented at a conference at the University of Arizona, Dec 10-11, 1981) (chap. 11, pp. 253-286). Hillsdale, NJ: Erlbaum.

Papert, S. (1960). Problämes ÇpistÇmologiques et gÇnÇtiques de la rÇcurrence. In P. GrÇco, J. B. Grize, S. Papert, & J. Piaget, Problämes de la cons[t]ruction du nombre. êtudes d'êpistÇmologie GÇnÇtique (Vol XI) (pp. 117-148). Paris: Presses Universitaires de France.

Papert, S. (1980). Mindstorms: children, computers and powerful ideas. Great Britain: Harvester Press.

Pirolli, P. (1986). A cognitive model and computer tutor for programming recursion. Human-Computer Interaction, 2(4), 319-355.

Poundstone, W. (1987). The recursive universe: Cosmic complexity and the limits of scientific knowledge. Oxford: Oxford University Press.

Robertson, W. C. (1990). Detection of cognitive structure with protocol data: Predicting performance on physics transfer problems. Cognitive Science, 14(2), 253-280.

Rumelhart, D. E. (1980). Schemata: The building blocks of cognition. In R. Spiro, B. Bruce, & W. Brewer (Eds.), Theoretical issues in reading comprehension (pp. 33-58). Hillsdale, NJ: Erlbaum.

Rumelhart, D. E., & Ortony, A. (1977). The representation of knowledge in memory. In R. C. Anderson, R. J. Spiro & W. E. Montague (Eds.), Schooling and the Acquisition of Knowledge (pp. 99-135). Hillsdale, NJ: Erlbaum.

Sacerdoti, E. D. (1977). A structure for plans and behavior. New York: Elsevier-North Holland.

Smith, D. A., Greeno, J. G., Vitolo, T. M. (1989). A model of competence for counting. Cognitive Science, 13, 183-211.

VanLehn, K. (1989). Problem solving and cognitive skill acquisition. In M. I. Posner, Foundations of cognitive Science. (chap. 14, pp. 527-579). Cambridge, MA: MIT Press.

VanLehn, K., & Brown, J. S. (1980). Planning nets: A representation for formalizing analogies and semantic models of procedural skills. In R. E. Snow, P. A. Federico & W. E. Montague (Eds.), Aptitude, learning, and instruction (Vol. 2: Cognitive process analyses of learning and problem solving, pp. 95-137). Hillsdale, NJ: Erlbaum.

Way, E. C. (1991). Knowledge representation and metaphor. Dordrecht: Kluwer.

Vitale, B. (1989). Elusive recursion: a trip in recursive land. New Ideas in Psychology, 7(3), 253-276.