simplified version of draw poker could be discovered from his behavior alone. Each <situation> was characterized by a number of features such as pot size, amount of last bet etc., and presented to the learning system together with the bet <decision> in that <situation>. When the induced rules were used to play the same hands against the same opponent (another program) the resulting behavior corresponded very closely to that of the subject. However, the strategy employed by the subject was found to be markedly different to that employed by the rules. In his (retrospective) protocol, one phrase recurred frequently: "I did not want the pot to get too high." This indicates that he was projecting a sequence of bets, in the course of which his contribution would exceed the limit he had subconsciously placed on each hand. The model, however, had no such rule -- merely one for calling whenever the pot became large.

This example illustrates a need for consideration of an extension to the "knowing how" and "knowing that" arguments discussed by Winograd [11]. Let us call it "knowing why". One program that knows why is Waterman's implicit trainer, which he says, "is merely an expression of the causal laws of which every poker player is aware". But how to teach a program such laws is a difficult problem (how to teach the teacher). Although the paradigm presented here is not capable of embodying such knowledge, its extension to a production system of the goal-directed type [5] may be a step in the right direction.

Knowing *why* might be expressed in terms of a high-level goal to a problem-solver. A scheme involving a hierarchical production system is currently under development, using the game of Go as a task domain. The system incorporates a number of sets of (condition, action, goal) rules, each set operating within a certain level of abstraction of the frame space. The goal entry in each rule indicates its applicability, the condition entry a requisite aspect of the current problem state, and the action entry an appropriate decision. Actions proposed by a goal directed scan of the nth level (of abstraction) set of rules are converted into goals to be achieved by the invocation of rules in the n-1th level sets. In this way, a top-level goal can be successively transformed by rules operating at various levels of abstraction into a zero level action (in the case of Go, a zero level action comprises a move together with some expected continuations). In addition, facility is provided for the construction of plans at any level of abstraction: if the condition part of an nth level rule does not find a match in the current problem state, it is transmuted into a goal (at the nth level) to achieve the desired problem state. In this way a sequence of nth level actions will be proposed, leading to a series of corresponding zero level actions. Conflict resolution is performed only when all plans have been expressed interns of zero level actions (thus no pruning of the search tree is made).

The plan selected is that whose actions occur most frequently (at any stage) for all the plans. If no single plan is deemed superior, the shortest path among the strongest competitors is chosen on the basis that it has less room for error -- in cases of equality an arbitrary choice is made.

Whether there is scope within this system for learning mechanisms is at present uncertain, although the use of a PS organization enables the addition of new rules to be performed without altering the control structure. When an action is taken, the rules used may be displayed on request in order that weaknesses may be removed by the addition or deletion of rules at any level.

10 Conclusions

The initial objective of the work reported here was to seek a completely general method of learning symbolic descriptions. To some extent, this has been achieved, but at the price of expertise; that is, the resulting concepts are not sufficiently expert in describing essential elements of an arbitrary task domain. The primary drawback is that the method has no facility for learning intentions, or goals. It may be hoped that the paradigm outlined in the last section will be capable of handling goals in diverse domains, but how it could learn these goals is very much an open question.

References

activation process operating on the network, the simulated ability to execute several procedures in parallel, and the use of strength measures to select among competing productions and competing paths in the network.

We have been working on a production system model of human cognition called ACT. An earlier version of the ACT system, called ACTE, is described in Anderson [3]. Anderson, Kline, and Lewis [5], and Kline and Anderson [21]. That system has been used to develop mini-models for retrieval from memory, inference making, language comprehension, question-answering, and problem solving. We are currently working on a new version of ACT called ACTF. This paper discusses a number of the design decisions underlying the ACT system. We will discuss how these design decisions are motivated by both psychological and artificial intelligence (AI) considerations.

What is ACT?
ACT is at the same time a high-level programming language and a theory of the cognitive mechanisms underlying human information processing. A high-level programming language is a formalism that facilitates programming certain kinds of algorithms. However, it may also be difficult to program algorithms that are more general than the intended kinds, so that high-level programming languages are often "special-purpose". ACT is a special-purpose programming language in this sense. But the fact that certain processes or algorithms can be coded in ACT more efficiently and easily than others is the means by which ACT provides a theory of the cognitive mechanisms that underlie human information processing. Humans are also more successful at certain cognitive processes than others. The hope is that ACT limitations correspond to human limitations. If this is so, psychological theories of specific cognitive behaviors automatically acquire a certain face validity when embodied as programs in the ACT programming language.

Given the adaptiveness and flexibility of human cognition no static ACT program can serve as an adequate psychological model. ACT must have the capability for evolving new programs and our current work is focused on developing a learning program. However, space limitations do not permit us to discuss this learning work, we will confine the discussion to the principles by which ACT programs are interpreted and executed.

A Relationship Between Psychology and AI
We conjecture that there is a strong relationship between cognitive psychology and those AI efforts concerned with developing general and adaptive systems. (This is what we will mean by AI for the remainder of this paper.) This conjecture can be stated as follows:

- Good cognitive psychology is good artificial intelligence. (1)
- This majorit of this paper will be devoted to illustrating how the psychological hypothesis has influenced the design of the ACT system. However, first we would like to discuss the implications of this claim.

What exactly does it mean? By "good psychology" we mean a theory that meets four scientific criteria: parsimony, effectiveness, broad generality, and empirical accuracy (see [3] for a discussion of these). Of principal importance to our points are the criteria of effectiveness and empirical accuracy. Effectiveness refers to the constraint that the theory be specified well enough that predictions can be rigorously derived from it. The important consequence of effectiveness is that a computer simulation of the theory can be produced. By empirical accuracy we mean that it successfully predicts human behavior in various cognitive tasks. Thus by conjecture (1) we mean that the computer simulation of a psychological theory that successfully accounts for empirical data will be translatable into a good AI program. By a good AI program we mean one that produces intelligent behavior under reasonable constraints of computational efficiency. Since a good psychology program will mimic human behavior, a reasonable standard of intelligence is guaranteed. The non-obvious aspect of our conjecture concerns the matter of computational efficiency.

If conjecture (1) is true, one way to attempt AI is to develop a program that models psychological data. Note, however, we are not claiming that this is the only way to develop a good AI theory. Moreover, we are explicitly not endorsing the claim that an AI program must meet standards of good psychology. That is, we are not claiming:

- Bad cognitive psychology is bad artificial intelligence, (2)
- nor its contrapositive equivalent:
- Good artificial intelligence is good cognitive psychology. (3)

Claim (3) has been put forward (but not endorsed) by Newell [25] as a possible relationship between AI and psychology. It has been endorsed informally by a number of workers in AI. (We wonder if its endorsers were aware of its equivalent contrapositive (2) which would make all AI endeavors subject to psychological test.) The argument for (3) is that the feats of human intelligence are so difficult that there is essentially only one way of accomplishing them on any physical device including a human brain or a serial computer. While what is meant by "essentially only one way" is uncertain, we feel the claim is implausible. We find this claim implausible in light of the non-identifiability results in psychology [3], the behavioral equivalences among different machines in automata theory, the existence of very different programs to perform a range of functions from sorting to parsing, and the observation that nature has evolved different mechanisms to serve the same function.

So, we feel there are different routes to the goal of good AI, only some of which involve psychological theories. For example, there would be nothing wrong with an AI program that found it easier to analyze a sentence than a picture, but there would be something wrong with a psychological theory that did. However, it is a fact that there is no current AI program that begins to meet the criteria of broad generality and adaptivity. Thus, while there may be many ways to achieve a good AI program, it clearly is proving difficult to discover any of these ways. Thus, while it is not necessary for the AI worker to follow psychological research in designing his systems, it might prove to be a good heuristic in trying to search for an adequate program.

The implications of conjecture (1) about the relationship between psychology and AI are not one-sided. This can be seen by considering the contrapositive of (1):

- Bad artificial intelligence is bad cognitive psychology. (4)

This places a new constraint on a psychological theory. It must be translatable into a good AI program. In particular, the implementation of a theory must obey certain constraints on computational efficiency. For instance, a theory of language processing would be judged inadequate if its best simulation produced comprehension times that were exponential functions of sentence lengths or which required exponentially increasing computational space.

We think complexity functions are better measures of computational efficiency than are absolute amounts of computational space or time. Judgments about what is an acceptable absolute amount are determined by current machine capabilities. It would be silly to cripple our psychological theory because current machine capabilities do not match those of the human brain. It would be particularly silly given the reasonable expectation that current limitations will be exceeded by many orders of magnitude over the next decades.

However, evaluating the efficiency of an algorithm in terms of complexity functions is not without its problems either. Complexity functions make no allowances for the fact that often only a restricted range of complexities are encountered in a practical problem. AI programs employing algorithms with superior performance on this restricted range would be...
preferred even if the asymptotic performance of these algorithms was very poor.

With suitable allowances for this problem with complexity functions, we might understand (1) and (4) as asserting that a good psychology program will employ algorithms which are characterized by reasonable complexity functions. But, there are still problems. Suppose our best simulation predicted that a certain process took an exponentially increasing amount of time as a function of problem complexity. That would not be unacceptable if it could be shown that the best known computer algorithm also displayed this complexity function and that human processing time increased exponentially with problem complexity. It is also the case that human behavior often displays a poorer complexity function than the best known algorithm. For instance, consider the time it takes humans to identify a concept in a concept-identification task [17]. Their identification times are often linear in the number of possible hypotheses rather than logarithmic. However, humans can be trained to implement a logarithmic algorithm. The upshot of these qualifications is that (1) and (4) amount to the requirement that the best algorithm implementable in a psychological theory like ACT display the best possible complexity function on a serial computer. We would conjecture that humans are capable of achieving this same complexity function after suitable training.

Now that we finally understand what (1) and (4) should be taken to mean, we see that for (1) and (4) to be valid the human brain and the serial computer must be subject to the same complexity constraints. That is, if either computing device can employ algorithms whose complexity functions are unattainable by the algorithms available to the other, (1) and (4) will probably be false.

The fact that humans can execute computer programs suggests that the brain is not at a disadvantage in this respect. On the other hand, we know of no physically realizable computational device (the brain included) that achieves better complexity functions than the serial computer.\(^1\)

Thus, barring the discovery that the brain has fundamentally superior computational abilities, (1) and (4) would appear to characterize the relationships between good psychology and good AL. If this characterization is accurate, there is potential for fruitful interaction between the two disciplines. The remainder of this paper will discuss how AI considerations and psychological considerations converge in the case of the ACT program.

Propositional Network

The data base in ACT is a propositional network. A propositional network is an associative network structured so it can be divided into units having a propositional status. There has been some concern whether associative networks have adequate expressive power (cf., [36]). However, in conjunction with the production system it can be shown that ACT has expressive power at least equivalent to the predicate calculus [3]. The condition of an ACT production specifies network configurations that must be present or absent for the production to apply. The execution of an ACT production can cause new propositions to be built in memory and old propositions to be modified.

The associative network structure of ACT is such that each concept indexes all the propositions it occurs in. The structure is double-linked so it is possible to go both from concept to proposition and from proposition to concept. This associative indexing feature of ACT nicely illustrates the convergence of psychological and AI considerations. There is a large amount of data indicating some sort of associative structure in memory. A simple phenomenon is that of word association: Take a word, generate a semantic associate to it, take that word and generate an associate to it, and so on. For instance, starting with dog: dog-bark-hear-eat-organ-body-weight-scale-step-foot-tail etc. The obvious way to simulate such free association protocols (cf., [20]) is a search process over an associative network. The computational advantage of associative indexing is also well understood. It serves to make time to retrieve a fact approximately independent of the number of facts in the data base.

There are also converging arguments for imposing a propositional structure on the data base rather than some simpler associative structure. Most AI programs that have performed inferential reasoning have used proposition-like representations of information. The reason is that a propositional structure abstracts out and makes salient aspects of the information relevant for deciding the validity (or plausibility) of inferences.

There are numerous sources of psychological data supporting the concept of propositional organization. For instance, there is considerable evidence (e.g., [6, 4, 16]) that propositions tend to be forgotten as units. That is, the conditional probabilities are high that if one element of a proposition is forgotten the remainder will also be forgotten. While subjects do occasionally recall only partial propositions (e.g., "The hippo touched somebody but I can't remember who") such reports are statistically rare. There is also considerable evidence (e.g., [28, 32]) that subjects, in remembering sentential information, often do not remember the exact wording of what was said but only the propositional content.

**Spreading Activation**

The concept of spreading activation has been quite popular in cognitive psychology (Collins & Loftus, 1975; Collins & Quillian, 1972; Kieras, 1977). There is an activation process that operates on the ACT network. Particular nodes can be activated either by stimulation from environmental events or by execution of production actions. With the passage of time activation spreads from the source nodes to associated structures. Productions, in matching their conditions, can only inspect the active portion of memory. Therefore, this activation process serves to focus attention. There are also mechanisms in ACT which deactivate all the structure that has been activated by a source when that source loses its activation.

A spreading activation process was suggested by Quillian [27]. There is considerable physiological evidence for an associative spread of excitation through the nervous system [29]. There is also considerable behavioral data pointing to the utility of the concept [3, 11]. For instance, subjects can be slowed in deciding a proposition is false if there is an irrelevant connection among the elements of the proposition (e.g., Madrid is in Mexico, see [11]). In other circumstances such irrelevant connections can facilitate processing [23]. Such effects indicate a diffuse activation process that is priming all connections, relevant or not.

It seems that Quillian's spreading activation notion has not received much acceptance in the AI community (see [24, 34] for criticisms). It is computationally expensive to compute a spread of activation. In fact, in our own simulation we have found it too expensive to simulate faithfully the conceptual-neural model we would want to endorse. Rather we approximate this with a much quicker activation process. We suspect that it is the cost of the activation computation that accounts for the inefficiency of its use in AL. By application of (4) we might be tempted to conclude that the conceptual-neural model is bad cognitive psychology. However, this problem with spreading activation may reflect only current computational limits. Given faster processors and the
prospect of parallel computation, we suspect that efficiency objections to the conceptual-neural model will be less serious with passing years.

The computational function of the activation process within ACT is to enable us to run a pattern-evoked production system in a propositional network but to avoid linear (or worse) growths in the amount of computation time with the size of the data base or the number of productions. This is achieved in two ways. First, the activation process serves to limit the amount of structure needing to be searched in determining whether the condition of a nominated production matches.

Second, productions are nominated for consideration by the activation process. Associated with each node is a list of productions that make reference to that node. When that node is activated these productions are considered. Since the number of nodes that are active is independent of the total number of productions, under this scheme the growth of computation time with number of productions will depend on the number of productions that are associated with any given node. Newell (personal communication—see also [22]) has argued that in realistic problem domains the number of productions tends to increase more rapidly than the number of data base elements. Thus, if this were true of ACT, there would be an increase in the production to node ratio with growth in the size of the system. Newell has observed from simulations that this increase in the ratio is approximately logarithmic with the number of productions. We would regard a logarithmic complexity function as acceptable.

The activation process basically serves to limit the amount of data that the system has available at any one time. Therefore, it is important to the operation of the system that the currently available set of data not be just randomly selected but rather be associated to those source nodes the system is currently focused on. These are the currently relevant data. For instance, when a word is heard it is made a source node, activation spreading from that node will activate syntactic and semantic information needed for its comprehension.

While the utility of activating the associative surround of items in focus is clear, the utility of stretching out this activation process over time may not be so obvious. Why not immediately activate all the structure that is going to be activated? The spreading process allows ACT to focus first on those data most likely to be relevant, and to try less relevant data later. Thus, the spreading process allows the system to "buy" its limited computational resources on what is most promising first.

This is facilitated by having strengths associated with network links. The strength of a link reflects how frequently and recently it has been involved in successfully matching the condition of a production. In the spread of activation, the total amount of activation energy at a node is divided among the links emanating from that node. The amount of energy given to a link is a function of its strength relative to the strength of all links. The rate at which activation will spread down a link to activate structures connected to the link is a function of this amount of energy. This means the structure that tends to be more rapidly activated is the structure that has more often proven useful in matching productions. This, of course, is a computationally sensible criterion for ordering the activation of structure. It is also a well-documented psychological fact [3, 4] that subjects can retrieve more quickly information which they have used often.

This strength mechanism is also responsible for forgetting in ACT. Forgetting of information occurs when the links encoding a structure become so weak relative to competing links that it becomes effectively impossible to reactivate the structure. This implies that forgetting is due to competition by interfering information—one of the most well-documented facts about human memory [4].

Virtues of Production Systems

Production systems by now are a familiar AI formalism and there are discussions of the advantages that they offer (cf., [14]). Therefore, we will focus mainly on the advantages production systems provide as a psychological theory and on those aspects of the ACT production system that are unique. A striking similarity has been noted [3, 26] between production systems and the other stimulus-response (S-R) theories in psychology. The connection between condition and action is quite similar to the S-R connection. There have been a series of psychological critiques of the S-R theory [3, 4, 8, 10, 18]. These critiques have basically been aimed at the computational power of the S-R formalism. It has been shown [18] that at least certain versions of S-R models do not have the computational power of finite state machines. However, production systems circumvent these limitations by use of variables, patterns, and memory [3]. Thus, production systems can be seen as having some of the psychological advantages of S-R theory without forfeiting computational power. The positive features of production systems discussed below are also true of S-R theories.

Data-Driven Character

One of the important features of production systems is their data-driven character. In each cycle of the production system there must be a reappraisal of the consequences of the current knowledge state for the control of behavior. Thus information that comes in and changes the knowledge state can have immediate effects on behavior. The survival advantage of a data-driven processing system for humans is obvious. There is also evidence that human cognition, even when survival is not at stake, operates in a strongly data-driven mode. For instance, the work on chess [9, 26] indicates that chess masters do much of their intellectual work by means of pattern recognition, and that chess masters usually perceive the correct move within a few seconds of scanning the board. It has been argued that chess masters' knowledge of board positions can be properly modeled by a large production system where individual productions contain board configurations as condition-patterns and appropriate responses (often moves) as actions.

Unity of Control and Data Store

One of the central features of production systems is their lack of distinction between the medium that stores control information and the medium that stores data. Unlike most programming languages there are no special facilities for storing control information—so separate program counter, pushdown stack, etc. All control information must be stored in the same data base (e.g., Newell's PS this is STM, in ACT it is the propositional network) that serves to store the input and results of computations.

Intuitively, it seems compelling to us that unity of control and data is a feature of human cognition. Fortunately, we do not have to rely just on intuition. There is some psychological data supporting this assumption. A variety of experiments have looked for a trade-off between the size of the immediate memory span and the computational complexity of an on-going task [7, 13, 30, 33]. For instance, Wanner and Maratos [31] has subjects hold a set of words in memory while the subjects tried to comprehend sentences of various syntactic complexity. In the ATN model they were testing, this complexity translated into amount of control information that had to be held. (It would translate into amount of control information into most computational models.) They found subjects were able to recall fewer of the words when comprehending sentences of greater syntactic complexity, and hence requiring more control information. This trade-off between the amount of information in memory span and the amount of control information is typical of the research in the area.
results in this area. The hypothesis that there is one storage medium for data and control information predicts this combined storage limitation.

Our feeling is that the use of a single representational formalism for control information and all other data has advantages for a system that is to acquire and modify productions. The role of an individual production in the overall flow of control becomes more apparent, and this comprehensibility should be as much of an advantage for a learning system which must make decisions about the function of a particular production as it is for a human programmer. The structure of productions and their rules of interpretation are also considerably simplified if there is no distinction between control information and other data, and this simplicity is also an advantage for production acquisition.

Unfortunately, while production systems tend not to make a formal distinction between control and data, in practice, one often finds an implicit distinction made by the programs written within production systems. This is seen in the use of special control conventions in order to achieve the same sequential execution of functions found in conventional programming languages. Although some of our earlier work in ACT had made extensive use of such control conventions, we are currently trying to avoid them because they forfeit some distinct advantages of production systems.

Modularity

Another important feature of production systems is the modularity of the individual productions. Because each production makes reference to a data base common to all productions and because no production makes reference directly to other productions, individual productions tend to be independent of one another. That is, if a particular production is added, deleted, or changed, the performance of the system tends to remain relatively unaffected. We feel that claims about the modularity of production systems have been sometimes exaggerated. There can be production systems designed such that the change of one production would have disastrous effects on the system. However, in a sense such production systems reflect "bad programming style". Production systems should be constructed to maximize the modularity of individual productions.

The advantage of modularity to the comprehensibility of the system is obvious. The advantage of modularity to the development of a learning system capable of self-extension is equally obvious. The basic modularity of human information processing is attested to by the gradual development of our processing abilities and by the fact that new skills almost never have disastrous interactions with old. (Of course, these facts about human modularity could be produced by formalisms other than production systems.)

Parallelism

There are a number of places in ACT where we have found it useful to simulate parallel computation. As discussed above, activation is conceived of as spreading simultaneously from all active nodes to the associated network structure and in the process, selecting the productions that reference these nodes for further testing. Not only does ACT select in parallel among a large set of productions on each cycle, it is also the case that on each cycle ACT simulates the ability to apply (test and execute) a number of productions in parallel. The evidence is quite compelling that humans can carry along a number of processes in parallel. For instance, consider the well-worn example of being able to hold a conversation and drive at the same time. We have found parallel procedures useful in language processing [5]. In comprehending a sentence one must perform a large number of operations--make perceptual judgments about words, switch attention from word to word, perform syntactic analysis, perform semantic analysis, resolve pronominal references, recognize the referents of definite descriptions, perform inferences, etc. It is implausible that each of these operations occupies separate segments of time with control switching among them. It is much more intuitively plausible to suppose that these various linguistic processes proceed in parallel.

While we do allow ACT to pursue more than one process in parallel, there is a capacity limitation on how much can be computed in parallel. There is a probabilistic parameter that places a maximum on the mean number of productions applied per cycle. In the current implementation this parameter is (arbitrarily) set to limit this mean to 10 productions per cycle. As the number of productions required per cycle approaches this limit, performance degrades. This degradation is produced by making productions wait multiple cycles before applying and forcing certain processes (sequences of productions) to drop out. The evidence is quite clear for such capacity limitations in the human case ([3], Section 6.3). A familiar example is the problem of holding a conversation in difficult traffic. As the amount of computation required for processing traffic information increases, less capacity is left over for the conversation. The AI motivation of such a limitation is simply that the limits of finite processing capacity must be respected in any physical machine (whether it has parallel processors or not).

Given that only a small number of productions will be allowed to apply in parallel, it becomes important to have some way of deciding which of those that can apply are most relevant. This is very much like the problem discussed above of insuring that the currently most relevant nodes in the network are those that are allowed to be active. A solution in terms of strength measures is also adopted in this case. Associated with each production is a strength reflecting the past success of that production. Details of how this strength is computed are motivated by learning considerations and so have been omitted from this discussion. For present purposes it is important only to note that the probability that a production will apply on a cycle when its condition is satisfied depends on the strength of that production relative to the strengths of all the other productions whose conditions are also satisfied. This means that the stronger productions (and hence the more successful processes) tend to be tried first and are less disturbed by processes going on concurrently.

Consider the implication of these strength mechanisms for processing of structural ambiguity in language where competing sets of productions are responsible for different interpretations of the same linguistic structure. The relative frequency of the two structural interpretations will be reflected by the relative strengths of the two sets of interpretative productions. We would therefore expect that subjects would tend to interpret the sentence according to the more frequently intended interpretation, but that they could retrieve the other interpretation and that they would sometimes choose the less frequent interpretation first. Also, we would expect that by bombarding the subject with examples of the less frequent interpretation we could make it the favored interpretation of the ambiguous structure. All these implications are known to be true [15].

References
MIXTURES OF STRATEGIES IN STRUCTURALLY ADAPTIVE PRODUCTION SYSTEMS: EXAMPLES FROM SERIATION AND SUBTRACTION

Richard M. Young
Department of Artificial Intelligence
Edinburgh University Forrest Hill Edinburgh EH1 2LH Scotland

Abstract
Production systems (PSs) can be written which are structurally adaptive in the sense of exhibiting different strategies, or mixtures of strategies, depending on the circumstances of the task. They consist essentially of the union of the individual PSs for each strategy. Such PSs begin to approximate the flexibility and adaptiveness of skilled human performance. An earlier analysis of children's seriation behavior is discussed in this light, and a PS for doing subtraction in a flexible manner is sketched. It raises a number of issues concerning the use of PSs as models of cognitive processes: the role of explicit rule-ordering; the representation and retrieval of facts; and the possibility of eliminating the need for explicit "housekeeping" rules. The question of ordering is particularly interesting, since it appears that, on the one hand, explicit ordering is needed in order to specify fully a particular method, while on the other hand a more faithful model results if ordering is ignored so that certain aspects of the behavior are left unspecified.

According to Newell and Simon [5], "production systems are the most homogeneous form of programming organization known". One consequence of this homogeneity is that a production system (PS) for solving a problem can be written in such a way that each individual rule represents an independent fragment of potential activity on the part of the problem solver and generates a meaningful component of the total problem solving process. Properly exploited, this independence enables one to write PSs which, instead of being committed to one particular strategy for tackling the problem, consist essentially of a mixture of different strategies and display an appropriateness and flexibility of behavior that begins to approximate the characteristics of skilled human performance.

---

1. This paper was written while the author was supported by a grant from the UK Social Science Research Council. I would like to thank: Benedict du Boulay for a discussion of the diferent ways of teaching subtraction, and for comments on an earlier draft; Michael Bennett, whose analysis of the errors in 1,360 sums done by 10 year olds [1] first got me interested in subtraction; and Tim O'Shea and an anonymous referee for perceptive criticism.