Adaptive Search Through Constraint Violations

Stellan Ohisson and Ernest Rees

The Learning Research and Development Center,
University of Pittsburgh,
Pittsburgh, Pennsylvania 15260

Technical Report No. KUL-90-01
January, 1990

Reproduction in whole or in part is permitted for any purpose of the United States Government. Approved for public release; distribution unlimited.

Copyright © 1990 Stellan Ohisson

Preparation of this manuscript was supported by ONR grant N00014-89-J-1681, and by the Xerox University Grant to the University of Pittsburgh. The opinions expressed do not necessarily reflect the positions of the sponsoring agencies, and no endorsement should be inferred.
Restructuring consists of a change in the representation of the current search state, a process which breaks an impasse during problem solving by opening up new search paths. A corpus of 52 think-aloud protocols from the domain of geometry was scanned for evidence of restructuring. The data suggest that restructuring is accomplished by re-parsing the geometric diagram.
Knowledge and Understanding in Human Learning

Knowledge and Understanding in Human Learning (KUL) is an umbrella term for a loosely connected set of activities lead by Stellan Ohlsson at the Learning Research and Development Center, University of Pittsburgh. The aim of KUL is to clarify the role of world knowledge in human thinking, reasoning, and problem solving. World knowledge consists of general principles, and contrasts with facts (episodic knowledge) and with cognitive skills (procedural knowledge). The long-term goal is to answer six questions: How can the conceptual content of a particular knowledge domain be identified? How can a particular person’s knowledge of a given domain be diagnosed? How is principled knowledge utilized in insightful performance? How does principled knowledge influence procedure acquisition? How is principled knowledge acquired? How can instruction facilitate the acquisition of principled (as opposed to episodic or procedural) knowledge? Different methodologies are used to investigate these questions: Psychological experiments, computer simulations, historical studies, semantic, logical, and mathematical analyses, instructional intervention studies, etc. A list of KUL reports appear at the back of this report.
# Table of Contents

Abstract 3  
Introduction 4  
Knowledge as Constraints on Possible Situations 5  
Learning from Constraint Violations 6  
Revising a Blocks World Rule 7  
Evaluation 9  
Discussion and Related Work 11  
References 13  
List of KUL Reports 14
Abstract

We describe HS, a production system that learns control knowledge through adaptive search. Unlike most other psychological models of skill acquisition, HS is a model of analytical, or knowledge-based, learning. HS encodes general domain knowledge in state constraints, patterns that describe those search states that are consistent with the principles of the problem domain. When HS encounters a search state that violates a state constraint, it revises the production rule that generated that state. The appropriate revisions are computed by regressing the constraint through the action of the production rule. HS can learn to solve problems that it cannot solve without learning. We present a Blocks World example of a rule revision, empirical results from both initial learning experiments and transfer experiments in the domain of counting, and an informal analysis of the conditions under which this learning technique is likely to be useful.
Introduction

The acquisition of control knowledge is a central problem in machine learning research. In one formulation of the control knowledge problem, a weak but general problem solver searches for the solution to a problem with an initial set of incomplete or faulty problem solving rules. Learning mechanisms such as discrimination (Langley, 1985), subgoaling (Ohisson, 1987a), or version spaces (Mitchell, 1982) can be applied to the information in the search tree to identify conditions that will enable the rules to solve the problem, or the relevant class of problems, with less search. Psychologists are interested in this learning scenario because it offers a possible model of how humans learn cognitive skills through practice (see, e. g., Anderson, 1989; Holland, Holyoak, Nisbett, & Thagard, 1986; Laird, Rosenbloom, & Newell, 1986; VanLehn, in press).

Psychological models of skill acquisition employ different problem solving mechanisms (forward search, backward chaining, means-ends analysis, planning, universal weak method) and different learning mechanisms (analogy, chunking, composition, discrimination, grammar induction, subgoaling), but with only a few exceptions (Anderson, 1989; Ohlsson, 1987b; Ohlsson & Rees, 1988) they have focussed on empirical learning methods. They identify rule conditions by performing some form of induction (in a broad sense) on the examples of correct and incorrect operator applications embedded in the search tree. Empirical learning methods contrast with analytical methods such as explanation-based learning (EBL) which identify rule conditions by applying knowledge about the relevant problem domain (Minton, 1988). But analytical learning methods are particularly interesting from a psychological point of view, because they offer a possible explanation of the facilitating effect of domain knowledge on procedure acquisition. Psychological experiments have shown that knowledge of the principles of a domain enables people to learn procedures faster and apply them more flexibly (see, e. g., Kleras & Bovair, 1984) as compared to conditions in which such knowledge is absent.

We describe a technique for knowledge-based procedure acquisition which is based on the idea that the main function of knowledge is to constrain the possible states of affairs. Incomplete control knowledge will frequently lead to the generation of search states that violate such constraints. The information contained in constraint violations can be used to identify new rule conditions adaptively, before a correct solution path has been found (Mostow & Bhatnager, 1986). The technique is implemented in a running simulation model called HS. We present data from both initial learning experiments and transfer experiments, and an informal analysis of the conditions under which our learning technique is likely to be useful. Our system is related to the FAILSAFE system described by Mostow and Bhatnager (1986), to the proceduralization hypothesis proposed by Anderson (1989), and to the planning net model of counting competence put forward by Smith, Greeno, and Vitolo (in press). A comparison with these systems will be postponed until the discussion section.
Knowledge as Constraints on Possible Situations

We are interested in the cognitive function of general knowledge. Many discussions of knowledge implicitly assume that the function of general knowledge is either to summarize particular facts or to enable explanations and predictions. There is no doubt that knowledge has those functions. However, we want to suggest that knowledge also can have the function of constraining the set of situations that one can reasonably expect to happen. The laws of conservation of mass and energy and the laws of commutativity and associativity of addition are examples of general principles that constrain the possible states of affairs. Faulty control knowledge, e.g., an incorrect laboratory procedure or a buggy addition algorithm, is likely to lead to violations of such constraints.

To capture the idea of general knowledge as constraints on possible situations, we encode a principle C as a state constraint, i.e., as an ordered pair of patterns \(<C_r, C_s>\) in which \(C_r\) is the relevance pattern and \(C_s\) is the satisfaction pattern. For example, the law of commutativity of addition expressed as a state constraint becomes \(if x + y = p and y + x = q, then it should to be the case that p = q.\) The principle of one-to-one mapping becomes \(if object A has been assigned to object B, then there should not be some other object X which also has been assigned to B.\) The law of conservation of mass becomes \(if M_1 is the mass of the ingredients in a chemical experiment, and M_2 is the mass of the products, then it should to be the case that M_1 = M_2.\) A constraint consists of a pair of patterns because all constraints are not relevant for all problem types. The relevance pattern of a state constraint specifies those search states (situations) in which the corresponding principle applies. The purpose of expressing domain knowledge in state constraints is to enable the HS system to efficiently identify search states that violate principles of the domain. This requires a \(\text{MATCH}(C, S)\) predicate that can decide whether a given pattern matches a given search state. We have used a RETE pattern matcher (Forgy, 1982) as our MATCH predicate.

HS is a relatively standard production system architecture that has been augmented with the state constraint representation. The system is given a problem space (an initial state, a set of operators, and a goal criterion), and a set of (minimally constrained) production rules. The initial state is a fully instantiated description of the problem, an operator consists of an addition list and a deletion list, and the goal criterion is a pattern. The system solves problems by forward breadth-first search through the problem space. Forward search is a very weak method, but since HS searches adaptively (Mostow & Bhatnager, 1987), improving its rules before it has found a complete solution path, it need not search the problem space exhaustively. HS searches until it encounters a constraint violation, learns from that violation, backs up to the initial state, and tries anew to solve the problem. If a state violates more than one constraint, HS selects one at random to learn from.

The identification of constraint violations proceeds as follows. When a production rule \(P: R \rightarrow O\) with condition \(R\) and action \(O\) is applied to a search state \(S_1\), thereby generating a descendent state \(S_2\), the relevance patterns of all constraints are matched against the new state \(S_2\). If the relevance pattern \(C_r\) of constraint \(C\) does not match \(S_2\), then \(C\) is irrelevant for that state and no further action is taken with respect to that constraint; if \(C_r\) does match, then \(C\) is relevant and the satisfaction pattern \(C_s\) is also
matched against $S_2$. If $C_s$ matches, no further action is taken. But if $C_s$ does not match, then a constraint violation is recorded. State constraints do not generate conclusions or fire operators; nothing is added to the problem description when a state constraint is applied. A state constraint functions as a classification device that sorts search states into those that are consistent with the principles of the domain and those that are not.

**Learning from Constraint Violations**

There are two types of constraint violations in the HS system. Suppose that production rule $P: R \rightarrow O$ was evoked in state $S_1$, leading to the generation of a new state $S_2$. In a Type A violation the constraint $C$ is irrelevant in $S_1$, and it is relevant but not satisfied in $S_2$. In a Type B violation the constraint $C$ is both relevant and satisfied in $S_1$, and it is relevant but not satisfied in $S_2$. Each type violation requires two different revisions of the rule $P$. The new rules are computed by regressing the constraint through the operator, but we will explain the technique with a set-theoretic notation which shows clearly why each type of violation gives rise to two new rules.

**Rule revisions for Type A violations.** If the relevance pattern $C_r$ does not match state $S_1$, but does match its immediate descendent $S_2$, then the effect of operator $O$ is to create expressions that enable $C_r$ to match. But since, *ex hypothesi*, the constraint $C$ is violated in $S_2$, $O$ does not create the expressions needed to complete the match for the satisfaction pattern $C_s$. This situation warrants two different revisions of the rule $P$ that fired $O$. First, the condition of $P$ should be revised so that the revised rule—call it $P'$—only matches in situations in which $O$ does not complete the relevance pattern for $C$, thus ensuring that the constraint remains irrelevant. Second, the condition of $P$ should be revised so that the revised rule—call it $P''$—only fires in those situations in which both the relevance and the satisfaction patterns of $C$ are completed, thus ensuring that the constraint becomes satisfied.

*Revision 1. Ensuring that the constraint remains irrelevant.* $O$ will complete $C_r$ when the parts of $C_r$ that are not added by $O$ are already present in $S_1$. Those parts are given by $(C_r - O a)$, where the symbol "-" signifies set difference. To limit the application of rule $P$ to situations in which operator $O$ will not complete $C_r$, we augment the condition of $P$ with the negated expression $not (C_r - O a)$. The new rule is

$$P': R \& not (C_r - O a) \rightarrow O$$

where "&" signifies conjunction.

*Revision 2. Ensuring that the constraint becomes satisfied.* To guarantee that $C_r$ will become complete, we augment the condition $R$ with $(C_r - O a)$. To guarantee that $C_s$ will also become complete we augment $R$ with those parts of $C_s$ that are not added by $O$. They are given by $(C_s - O a)$, so the desired effect is achieved by adding the entire expression $(C_r - O a) u (C_s - O a)$ to $R'$, where the symbol "u" signifies set union. The new rule is
Rule revisions for Type B violations. If the constraint $C$ is both relevant and satisfied in state $S_1$, and relevant but not satisfied in $S_2$, the effect of operator $O$ is to destroy the match for the satisfaction pattern $C_s$, but not for the relevance pattern $C_r$. This situation also warrants two revisions of rule $P$.

**Revision 1. Ensuring that the constraint is irrelevant.** Rule $P$ is revised so that it will only fire in situations in which constraint $C$ is not relevant and in which $C$ will not become relevant. This is accomplished by adding the negation of the relevance pattern $C_r$ to the condition $R$ of the rule. The new rule is

$$P': R \land \neg C_r \rightarrow O$$

**Revision 2. Ensuring that the constraint remains satisfied.** Rule $P$ is replaced by a rule $P''$ which only fires in situations in which the constraint remains satisfied. This is done in two steps. The first step is to constrain the rule to fire only in situations in which the constraint is relevant. This is accomplished by adding the relevance pattern $C_r$ to the rule condition. The second step is to constrain the rule to situations in which the match of the satisfaction pattern is unaffected by the action of operator $O$. This is accomplished by adding the negation of the intersection between the satisfaction pattern and the deletion list, $\neg(C_s \cap O_d)$, to the rule condition. The desired effect is attained by adding the entire expression $C_r \lor \neg(C_s \cap O_d)$, so the new rule is

$$P'': R \lor C_r \lor \neg(C_s \cap O_d) \rightarrow O.$$

The above description of the learning algorithm is simplified in the following respects: (a) Rules are not replaced by their descendents. The old rules are retained, but their descendents are preferred during conflict resolution. (b) In order to add parts of a constraint to a rule condition correspondences must be computed between the variables in the constraint and the variables in the rule. In the implementation those correspondences are computed by the regression algorithm. (c) A negated condition can cease to match as the result of the addition of expressions to a search state. Our revision algorithm handles those cases as well. (d) There are cases in which one of the two revisions results in the empty list of new conditions. In those cases only one new rule is created.

**Revising a Blocks World Rule**

The HS system has mainly been applied to arithmetic tasks such as counting a collection of objects, and subtracting multi-digit integers (Ohlsson & Rees, 1988). We nevertheless illustrate the rule revision algorithm with an example from the Blocks World, because of the widespread familiarity with this domain. Successful performance in the Blocks World requires knowledge of where blocks can be put down. Putting a block on the table or on top of a stack generally results in a stable situation, but trying to put a block on another block that already has other blocks stacked on top of it is likely to lead to the collapse of the stack. The following Blocks World rule says that if the hand is holding a block, and the goal is to put
the block down, and the hand is in the up position, and there is a possible support, then lower the hand:

\[ \text{(GOAL PUTDOWN } \langle \text{Block} \rangle)(\text{ISA BLOCK } \langle \text{Block} \rangle)(\text{HOLDING HAND } \langle \text{Block} \rangle) \]
\[ \text{(POSITION HAND UP)}(\text{ISA SUPPORT } \langle \text{Support} \rangle) \]
\[ \rightarrow \]
\[ \text{LowerHand}(\langle \text{Block} \rangle, \langle \text{Support} \rangle) \]

The operator LowerHand lowers the block onto the support, but does not let go of the block. It is defined by the deletion list
\[ O_d = \{(\text{POSITION HAND UP})\} \]
and the addition list
\[ O_a = \{(\text{POSITION HAND DOWN})(\text{ON } \langle \text{Block} \rangle \langle \text{Support} \rangle)\}. \]

Since blocks are members of the category supports, this rule will attempt to lower the block onto any other block in the world. If the supporting block is in the middle of a stack, this operation violates the principle that only one block can be on top of another block, which can be expressed as a state constraint with relevance pattern
\[ C_r = \{(\text{ON } \langle \text{Block} \rangle \langle \text{Support} \rangle)(\text{ISA BLOCK } \langle \text{Support} \rangle)\} \]
and satisfaction pattern
\[ C_s = \{(\text{not (ON } \langle \text{OtherBlock} \rangle \langle \text{Support} \rangle)(\text{not (EQUAL } \langle \text{OtherBlock} \rangle \langle \text{Block} \rangle)))\} \]

Lowering a block until it rests on a block that is not a top block, i.e., a block which has other blocks resting on it, leads to a violation of this constraint. Since the constraint cannot be relevant before the hand is lowered, this is a Type A violation.

Revision 1. Ensuring that the constraint remains irrelevant. The difference between the relevance pattern \( C_r \) and the addition list \( O_a \) is
\[ C_r - O_a = \{(\text{ISA BLOCK } \langle \text{Block} \rangle)\}. \]
The negation of this expression is added to the rule condition, so the new rule becomes:

\[ \text{(Goal: PUTDOWN } \langle \text{Block} \rangle)(\text{ISA BLOCK } \langle \text{Block} \rangle)(\text{HOLDING HAND } \langle \text{Block} \rangle) \]
\[ \text{(POSITION HAND UP)}(\text{ISA SUPPORT } \langle \text{Support} \rangle) \]
\[ (\text{not (ISA BLOCK } \langle \text{Support} \rangle) \}
\[ \rightarrow \]
\[ \text{LowerHand}(\langle \text{Block} \rangle) \]

where the new condition is in boldfaced typefont. This rule says that it is possible to put a block down on any support that is not a block. In the standard version of the Blocks World, the only support that is not a block is the table.

Revision 2. Ensuring that the constraint becomes satisfied. As noted above the difference \( (C_r - O_a) \) is in this case
\[ C_r - O_a = \{(\text{ISA BLOCK } \langle \text{Support} \rangle)\}. \]

Subtracting the addition list \( O_a \) from the satisfaction pattern \( C_s \) returns the satisfaction pattern itself.

January KUL-90-01 1990
because they do not have any expressions in common in this case. Adding \(((C_r - O_a) \cup (C_s - O_a))\) to the rule therefore generates the new rule

\[
\begin{align*}
\text{(Goal: PUTDOWN <Block>)(ISA BLOCK <Block>)(HOLDING HAND <Blocks>)} \\
\text{(POSITION HAND UP)(ISA SUPPORT <Support>)} \\
\text{(ISA BLOCK <Support>)} \\
\text{(not [(ON <OtherBlock> <Support>)(not (EQUAL <OtherBlock> <Block>)]})}
\end{align*}
\]

\[\rightarrow\]
\[
\text{LowerHand(<Block>, <Support>)}
\]

where the new conditions are in boldfaced typeface. This rule says a block can be lowered onto another block, if that other block is a top block, i.e., if it does not have any blocks resting on it.

In summary, the revision algorithm takes as input a violation of the constraint only one block can be on top of another block and sorts out the two action possibilities that are consistent with it—either put a block down on the table, or put it down on a top block—encoding each possibility in a separate production rule. The two new rules are not perfect, of course, and they will be revised further when they violate other constraints. Repeated revision of rules is a central feature of learning in the HS system.

**Evaluation**

The task of quantifying a collection of objects by counting them is interesting from the point of view of the cognitive function of principled knowledge, because observations of children show that they understand the principles that underly counting (Gelman & Gallistel, 1978; Gelman & Meck, 1986). Modifying slightly the analysis by Gelman and Gallistel (1978), we identify three counting principles: (a) The Regular Traversal Principle which says that correct counting begins with unity and generates the natural numbers in numerical order. (b) The One-One Mapping Principle which says that each object should be assigned exactly one number during counting. (c) The Cardinality Principle which says that the last number to be assigned to an object during counting represents the numerosity of the counted collection. These three principles form the conceptual basis of the procedure for standard counting, in which the objects are counted in any order. In order to probe children's understanding of counting, Gelman and Gallistel (1979) invented two non-standard counting tasks, ordered counting, in which the objects are counted in some pre-defined order (e.g., from left to right), and constrained counting, in which the objects are counted in such a way that a designated object is assigned a designated number. These three counting tasks require different procedures (control knowledge), but all three procedures are based on the above principles.

HS can learn the correct procedure for either of the three counting tasks. The input to the system consists of a problem space for counting, state constraint representations of the counting principles, and an initial rule set. Our representation for the counting task is very fine-grained, and the operations of setting and retracting goals are treated as search steps, so counting three objects requires 48 steps through the problem space. Since the initial rules are minimal, the branching factor before learning is between two and four, giving a search space of more than $60\times10^9$ states. This search problem is too large
Table 1: Initial Learning Effort for Three Counting Tasks.

<table>
<thead>
<tr>
<th>Counting task</th>
<th>Rule revisions</th>
<th>Production system cycles</th>
<th>Search states</th>
</tr>
</thead>
<tbody>
<tr>
<td>Standard</td>
<td>12</td>
<td>854</td>
<td>979</td>
</tr>
<tr>
<td>Ordered</td>
<td>11</td>
<td>262</td>
<td>294</td>
</tr>
<tr>
<td>Constrained</td>
<td>12</td>
<td>451</td>
<td>507</td>
</tr>
</tbody>
</table>

to be solved by brute force, but since HS searches adaptively, the system is nevertheless successful. Table 1 shows three measures of the amount of work required to learn each counting procedure. The number of rule revisions required is approximately the same (either 11 or 12) for each procedure. The number of states visited during learning is less than 10^3, so the system only needs to visit a very small portion of the total search space in order to find those rule revisions. In terms of either the number of production system cycles or the number of search states visited, standard counting is harder to learn than constraint counting, which in turn is harder to learn than ordered counting, a prediction which in principle is empirically testable.

Observations of children show that they can easily switch from standard counting to either of the two non-standard counting tasks (Gelman & Gallistel, 1978; Gelman & Meck, 1986). The most plausible explanation for this flexibility is that children can derive the control knowledge for the non-standard counting tasks from their knowledge of the counting principles. To simulate this flexibility we performed transfer experiments with HS. Once the system had learned a correct counting procedure, we gave it counting problems of a different type than the type on which it had practiced. For example, having practiced on standard counting, the system might be given constrained counting problems, and vice versa. To solve these problems the system had to adapt the already learned control knowledge to the new task. Since there are three different counting tasks, there are six possible transfers, all of which HS carried out successfully. Table 2 shows three measures of the amount of work required for each of the six transfers.

Three conclusions emerge from Table 2. First, the number of rule revisions is between one order of magnitude lower than the number of production system cycles or the number of search states visited, so HS predicts that the density of learning events during practice is low. Second, there is substantial transfer between the three counting tasks. The number of rule revisions required to learn any one of the three counting tasks from scratch is either 11 or 12; the number of revisions required to transfer to a different task is between 0 and 3 in five cases, a saving of approximately 75%. Third, transfer is asymmetric. Ordered counting does not transfer to constrained counting, but constrained counting transfers very well.
Table 2: Learning Effort for Six Transfer Tasks in the Counting Domain.

<table>
<thead>
<tr>
<th>Training task</th>
<th>Standard counting</th>
<th>Ordered counting</th>
<th>Constrained counting</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Standard</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Revisions</td>
<td>-</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>Cycles</td>
<td>-</td>
<td>110</td>
<td>127</td>
</tr>
<tr>
<td>States</td>
<td>-</td>
<td>119</td>
<td>141</td>
</tr>
<tr>
<td>Ordered</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Revisions</td>
<td>1</td>
<td>-</td>
<td>11</td>
</tr>
<tr>
<td>Cycles</td>
<td>184</td>
<td>-</td>
<td>297</td>
</tr>
<tr>
<td>States</td>
<td>209</td>
<td>-</td>
<td>334</td>
</tr>
<tr>
<td>Constrained</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Revisions</td>
<td>0</td>
<td>3</td>
<td>-</td>
</tr>
<tr>
<td>Cycles</td>
<td>162</td>
<td>154</td>
<td>-</td>
</tr>
<tr>
<td>States</td>
<td>180</td>
<td>190</td>
<td>-</td>
</tr>
</tbody>
</table>

to ordered counting. Although we do not yet possess the relevant observations, these predictions are in principle empirically testable.

Discussion and Related Work

In which task domains is constraint violation likely to be effective? The technique allows a system to identify, out of all possible paths in a search space, those paths which are consistent with the principles of the task domain. Let us call those correct paths. A correct path is not necessarily a useful path, i.e., a path that leads to a desired problem solution. Constraint violation is likely to be effective when (a) the ratio of correct to possible paths is small, i.e., when correct paths are rare, and (b) the ratio of useful to correct paths is high, i.e., when many correct paths are useful. In the counting domain every step is regulated by the counting principles, so every correct path is also a useful path. Another domain in which constraint violation might be useful is predicting the outcomes of chemical experiments, where all reaction paths that are consistent with the laws of chemistry need to be considered. But in proof spaces in algebra and geometry, where there are many mathematically correct paths which do not lead to a desired theorem, constraint violation is likely to be ineffective.

Our system is similar in basic conception to the FAILSAFE system described by Mostow and Bhatnager (1987) that operates in a floor planning domain. Both systems learn control knowledge during forward
search by using the information in failed solution paths to revise the rules that lead to those paths. Both systems encode domain knowledge as constraints on correct solutions, and both systems use regression to identify the new rule conditions. However, there are also differences. First, Mostow and Bhatnager (1987) argue that one of the advantages of adaptive search is that it becomes possible to make progress on problems for which the completion of a correct solution path through unconstrained search is infeasible. However, this advantage does not seem to be realized in the FAILSAFE system, since the system in fact completes an entire floorplan before testing whether it satisfies the constraints. The HS system applies its constraints after each problem solving step, and it learns before it has completed a correct solution. Second, the FAILSAFE system relies on the fact that the length of a floor plan solution is known a priori to identify failures. In contrast, the state constraint representation provides HS with a general method for identifying failures. Third, the FAILSAFE system learns one new rule for each failure, while HS learns two new rules in response to each constraint violation. The cause of this difference deserves to be analyzed in more detail than we can do here. Fourth, like other EBL systems, FAILSAFE uses its domain theory to construct explanations, a potentially complicated process which might require search, and which might fail if the domain theory is incorrect or incomplete. HS replaces the construction of explanations with pattern matching. Fifth, the FAILSAFE system can assign blame to rules which are several steps removed from the point of failure detection. This is an advance upon the HS system, in which blame is always assigned to the last rule to fire before failure detection.

Psychological models of learning do not usually address the problem of the cognitive function of general knowledge in procedure acquisition. One exception is the ACT* theory proposed by Anderson (1989), which claims that declarative knowledge structures are proceduralized during problem solving. The main difference between proceduralization and constraint violation is that in proceduralization declarative knowledge only participates in the creation of initial rules; further improvement of those rules is handled by empirical learning mechanisms such as composition and strengthening. In constraint violation declarative knowledge continues to influence rule revisions during the entire life time of the rule. The planning net model of counting competence proposed by Smith, Greeno, and Vitolo (in press) addresses the same phenomenon as the HS system—children’s flexibility in moving between different counting tasks—and their model also assumes that the source of this flexibility is a declarative encoding of the counting principles. However, Smith, Greeno, and Vitolo (in press) characterize their model as a competence model rather than as a process model, disclaiming any psychological reality for the processes they describe. It is therefore unclear how to conduct a comparison between their system and ours.

Acknowledgements
Preparation of this manuscript was supported by ONR grant N00014-89-J-1681, and by the Xerox University Grant to the University of Pittsburgh. The opinions expressed do not necessarily reflect the position of the sponsoring agencies, and no endorsement should be inferred.
References


1988


1989


1990


Distribution List

Ma, Linn B., Artelle
Code 1538
Naval Research Lab
Overlook Park
Washington, DC 20375-5000

Dr. Edwin Addisman
Media Laboratory
MIT
Cambridge, MA 02339

Dr. Beth Adelson
Department of Computer Science
Tufts University
Medford, MA 02155

Testimonial Document Center
AFFRDL-ISIS-TDC
West-Peapack AFB
NH 03572-4502

Dr. Robert Abler
Code N711
Human Factors Laboratory
Naval Training Systems Center
Orlando, FL 32813

Dr. Robert M. Altman
Computer Science Department
Temple University
Philadelphia, PA 19122

Mr. T. James A. Anderson
Philips Laboratories
345 Southborough Road
Bedford, Mass.
New York, NY 10530

Dr. James Anderson
Brown University
Department of Psychology
Providence, RI 02912

Dr. John R. Anderson
Department of Psychology
Carnegie-Mellon University
Schenley Park
Pittsburgh, PA 15213

Dr. Thomas H. Anderson
Center for the Study of Reading
174 Children’s Research Center
11 Geyo Drive
Champaign, IL 61803

Dr. Stephen J. Anderson, Chairman
Department of Information Systems
and Systems Engineering
George Mason University
4400 University Drive
Fairfax, VA 22030

Prof. John A. Assael
University of Warwick
Department of Psychology
Cowley CV4 7AL
ENGLAND

Edward Attan
Code 41231A
Naval Ocean Systems Center
Washington, DC 20375-5101

Dr. Patricia Beggsen
School of Education
443 E. University, Rm 1102D
University of Michigan
Ann Arbor, MI 48109-1259

Dr. James D. Baker
Director of Automation and Research
Advanced Computer Corp.
505 Main Street
Alexandria, VA 22314

Dr. Mary J. Baker
Navy Personnel R&D Center
San Diego, CA 92152-6500

Prof. G. O. Baras
University of Warwick
6022 Milano - via P. Storni 15
ITALY

Dr. Joseph C. Barbeau
86 Gage Avenue
Burney, PA 19121

Dr. Danis Bebe
University of Computer Science
Box 1488, Section B
Vanderbilt University
Nashville, TN 37235

Dr. John Becht
Teachers College, Box 8
Columbia University
525 West 126th Street
New York, NY 10027

Dr. Michael Blackmore
Code 942
Naval Ocean Systems Center
San Diego, CA 92152-5000

Dr. Arthur S. Blossom
Code N712
Naval Training Systems Center
Orlando, FL 32813-7100

Dr. Domart A. Bostom-Davis
Department of Psychology
George Mason University
4400 University Drive
Fairfax, VA 22030

Dr. Jay Bogost
Army Research Institute
AT&T: PEU-SP
5001 Eisenhower Avenue
Alexandria, VA 22353-5409

Dr. D. J. Bower
Guinness Technology, Inc.
800 Vessel Street
Pittsburgh, PA 15212

Dr. J. C. Broadnax
Center for Manufacturing
Engineering
National Bureau of Standards
Gaithersburg, MD 20899

Dr. Lyle E. Bourne, Jr.
Department of Psychology
Box 345
University of Colorado
Boulder, CO 80309

Dr. Hugh Burns
Department of English
University of Texas
Austin, TX 78710

Dr. Robert Callis
School of Education
Stanford University
Stanford, CA 94305

Dr. Joseph C. Campos
Center for the Study of Reading
University of Illinois
11 Geyo Drive
Champaign, IL 61803

Dr. James C. Cooper, Director
Center for Research into Previous
3145 Alberman Street
Washington, DC 20008

Dr. James G. Carbone
Computer Science Department
Carnegie-Mellon University
Schenley Park
Pittsburgh, PA 15213

Dr. Neil Carpenter
Center for Adaptive Systems
111 Commonwealth St., Room 244
Boston University
Boston, MA 02215

Dr. John M. Carroll
IBM Watson Research Center
User Interface Institute
P.O. Box 794
Yorktown Heights, NY 10598

Dr. Ruth W. Cheeley
CDEC
Hampton Hall
Carnegie-Mellon University
Pittsburgh, PA 15213

Dr. Fred Cheung
Pacific Bell
2600 California Street
San Francisco, CA 94108

Dr. Douglas chrono
Department of Psychology
Penn State University
University Park, PA 16802

Mrs. O. Clarke
818 South George Mason Drive
Arlington, VA 22204

Dr. Norma Cole
Columbia University
Department of Psychology
Univ. of So. California
Los Angeles, CA 90089-1061

Dr. Stanley Cogswell
Office of Naval Technology
Code 222
800 N. Quincy Street
Arlington, VA 22217-5200

Dr. Sara Conley
Carnegie Mellon University
Dept. of Education
Room 490 Roberts
Ithaca, NY 14853

Dr. Lynne A. Cooper
Department of Psychology
Columbia University
New York, NY 10027

Dr. Salma Danforth
3541 Hamptons Point
Cherry Hill, NJ 08035

Dr. Steve P. Day
Faculty of Law
University of Lasarburg
P.O. Box 414
Manchester
The NETHERLANDS 6290 ND

Dr. Kenneth B. Cross
Amazon Scrounges, Inc.
P.O. Drawer O
Santa Barbara, CA 93102

Dr. Gary Caudle
Institute for Instrumentation Systems
Texas Instruments A/C Labs
P.O. Box 60722
Dallas, TX 75226

Brian Ditchman
Training Technology Branch
3480 TICTW/TTXGC
Lowry AFB, CO 80226-5000
Dr. Harold P. O'Neill, Jr.
School of Education - WPH 801
Department of Educational Psychology & Technology
University of Southern California
Los Angeles, CA 90089-0011

Dr. Paul O'Rourke
Information & Computer Science
University of California, Irvine
Irvine, CA 92717

Dr. Stefan Olsson
Learning & D Center
University of Pittsburgh
Pittsburgh, PA 15260

Dr. James B. Olson
WICAT Systems
1879 South State Street
Omaha, NE 68105

Dr. Gary M. Olson
Cognitive Sciences and
Machine Intelligence Lab.
University of Michigan
725 Tappan Street
Ann Arbor, MI 48109-1234

Dr. Judith Ricebaum Olson
Graduate School of Business
University of Michigan
Ann Arbor, MI 48109-1234

Office of Naval Research,
Code ONR-25
800 New Jersey Avenue
Washington, DC 20306

Dr. James Ordway
Institute for Defense Analyses
1991 N. Beauregard St.
Alexandria, VA 22311

Dr. Warren Palmer
Mail Stop 229-3
NASA Ames Research Center
Moffett Field, CA 94035

Dr. O'donnell Park
Army Research Institute
PEN-2
5001 Eisenhower Avenue
Alexandria, VA 22333

Dr. Roy Pen
Institute for Research on Learning
2558 Hansberry Street
Patton, CA 94064

Dr. David N. Porteous
Project Zero
Harvard Graduate School of Education
7 Appian Way
Cambridge, MA 02138

Dr. C. Purves, Chair
Dept. of Psychology
Iowa State University
Cold Spring La-Hillen Rd.
Ames, IA 50010

Dr. Henry P. Perry
Naval Education and Training
Program Support Activity
Code 061
Building 2435
Pensacola, FL 32509-5000

Dept. of Administrative Sciences
Code 54
Naval Postgraduate School
Monterey, CA 93943-5026

Dr. Peter Polani
School of Education
University of California
Burlington, CA 94720

Prof. Turcato Poggio
Massachusetts Institute
of Technology 252-301
Center for Biological
Information Processing
Cambridge, MA 02139

Dr. Peter Polen
University of Colorado
Department of Psychology
Boulder, CO 80309-0340

Dr. Steven E. Porteus
Bowling Advanced Technology Center
PO Box 24346 and 7-L-64
Saratoga, CA 95812

Dr. Joseph Piddin
ATTN: PERI-SC
Army Research Institute
5081 Eisenhower Ave.
Alexandria, VA 22333-5409

Mr. Paul S. Rau
Code U-73
Naval Surface Weapons Center
White Oak Laboratory
Silver Spring, MD 20903

Dr. James A. Regan
University of Maryland
School of Medicine
Department of Neurology
22 S. Charles St.
Baltimore, MD 21201

Dr. J. Wesley Rigney
APHILID
Broom's AFB, TX 78233

Dr. Fred Rain
Physics Department
University of California
Burlington, CA 94720

Dr. Charles M. Rinehart
330 Huntington Hall
Syracuse University
Syracuse, NY 13244

Dr. David Rainberg
Rensselaer Polytechnic
Institute
University of Pennsylvania
Philadelphia, PA 19102

Dr. Laurent Rosenfield
Learning & D Center
University of Pittsburgh
2590 O'Hara Street
Pittsburgh, PA 15213

Dr. I. Jeffrey Richardson
Cancer for Applied AI
College of Business
University of Colorado
Boulder, CO 80309-0419

Dr. Edwin L. Rischard
Dept. of Computer and
Information Sciences
University of Massachusetts
Amherst, MA 01003

Mr. William A. Rizzo
Code 71
Naval Training Systems Center
Orlando, FL 32813

Dr. Linda G. Roberts
Scientific, Educational, and
Transportations Programs
Office of Technology Assessment
Congress of the United States
Washington, DC 20510

Dr. Ernest Z. Rodikof
AT&T Bell Laboratories
Room 3K-345
480 Mountain Avenue
Murray Hill, NJ 07974

Dr. Alan H. Schoolfield
University of California
Department of Education
Burlington, CA 94720

Lowell School
Psychological & Quantitative Foundations
College of Education
University of Iowa
Iowa City, IA 52242

Dr. James W. Schofield
Rm. LADC Building
University of Pittsburgh
3099 O'Hara Street
Pittsburgh, PA 15260

Dr. Ray Schubat
Computer Science Dept.
U.S. Naval Academy
Annapolis, MD 21402

Dr. Miriam Schumann
Code 12
Naval Personnel R & D Center
San Diego, CA 92152-6890

Dr. Judith W. Segal
ORSU
555 New Jersey Ave., NW
Washington, DC 20238

Dr. Robert J. Seidel
US Army Research Institute
5001 Eisenhower Avenue
Alexandria, VA 22333

Dr. Colleen M. Seldin
Institute for Cognitive Science
Mail Code C-413
University of California
San Diego
La Jolla, CA 92093

Dr. Michael G. Shafro
NASA Ames Research Center
Mail Stop 239-1
Moffett Field, CA 94035

Mr. Colin Sheppard
AXC2 Block 3
Advanced Research Establishment
Ministry of Defence Forton
Portsmouth Ha's 841344
UNITED KINGDOM

Dr. Lee S. Shulman
School of Education
507 Centre
Stanford University
Stanford, CA 94305-3084

Dr. Randall Shumaker
Naval Research Laboratory
Code 5118
4555 Overlook Ave., SW
Washington, DC 20375-5000

Dr. Edward Silver
LADC
University of Pittsburgh
3999 O'Hara Street
Pittsburgh, PA 15260