Appears in *Proceedings of the Thirteenth Annual Conference of the Cognitive Science Society* (1991). Chicago: Lawrence Erlbaum.

Learning, Memory, and Search in Planning

Pat Langley and **John A. Allen**
AI Research Branch (MS 244-17)
NASA Ames Research Center
Moffett Field, CA 94035

Abstract

This paper describes DÆDALUS, *a system that uses a variant of means-ends analysis to generate plans and uses an incremental learning algorithm to acquire probabilistic search heuristics from problem solutions. We summarize* DÆDALUS*' approach to search, knowledge, organization, and learning, and examine its behavior on multi-column subtraction. We then evaluate the system in terms of its consistency with known results on human problem solving, comparing it to other psychological models of learning and planning.*

Introduction

A central aspect of human intelligence is the ability to plan, that is, to generate action sequences that achieve one's goals. As a result, planning and problem solving are important topics within both cognitive psychology and artificial intelligence. However, relatively few planning systems have been designed with an eye toward explaining psychological findings, and even fewer provide accounts of human learning on planning tasks.

In this paper we describe DÆDALUS, a planning system that we designed with human behavior in mind. Unlike most recent work on learning and planning, DÆDALUS employs a combination of forward chaining and means-ends search, represents knowledge in a probabilistic framework, stores both cases and abstractions, and learns through an incremental process of concept formation. In the following section we discuss these aspects of the system, relating them to high-level knowledge of human behavior. After this, we consider its behavior on the domain of multi-column subtraction. Finally, we present a brief evaluation of DÆDALUS' and other models' abilities to explain robust psychological findings.

An Overview of DÆDALUS

Like most planning systems, DÆDALUS must solve problems that involve transforming an initial state into a desired state through the application of operators. The system describes each state as a set of literals (predicates with arguments), and it describes each problem or subproblem as an initial state conjoined with a set of differences that must be eliminated. DÆDALUS represents each operator in a similar manner, specifying its preconditions as a set of state descriptors and its effects as a set of differences. Later we present examples of states and problems from subtraction, but first let us consider how the system uses this representation to solve problems, organize knowledge, constrain search, and learn from experience.

Organization of Search

One of the most pervasive phenomena in problem solving is that humans carry out search through some problem space (Newell, 1980), and computational work on problem solving has focused on two basic approaches to organizing this search. In *forward chaining* or state-space search, one applies an operator to an initial state, another operator to its successor, and so forth. At each stage of this process, one considers an operator only if its preconditions exactly match the current state. In *means-ends analysis*, one chooses a difference between the current and desired state, selects an operator which reduces that difference, and attempts to apply the operator. If the operator's preconditions are not satisfied, the method is recursively called with the task of changing the current state into one that satisfies them. Once the operator's preconditions are met, a new state resulting from its application is generated. If the new state satisfies the goals, the method exits successfully; otherwise it recurses with the task of changing the new state into one that does satisfy the goals.

There is considerable evidence that humans use means-ends analysis when information is available about the desired state (Newell & Simon, 1972), letting them focus on operators relevant to goals. However, they can also solve problems for which there is no explicit goal description. Moreover, traditional means-ends systems examine only one difference at a time, and it seems unlikely that humans operate in such a non-gestalt manner. In response, DÆDALUS uses an alternative control scheme – *flexible means-ends analysis* – that prefers operators which reduce more differences and operators whose preconditions more closely match the current state. Thus, the retrieval process incorporates ideas from both approaches, biasing the system toward operators that have more effects and that are more nearly applicable. On tasks with explicit goals, the system takes both differences and state features into account; on less well-defined problems, it retrieves operators based on state descriptors alone. As we will see below, DÆDALUS can also place weights on each difference and state descriptor, letting the system attend to the features appropriate for a given domain.[1]

The system uses these biases to direct a heuristic depth-first search through the space of problem-solving traces. Figure 1 shows a successful trace that transforms the initial into the desired state for a problem involving multi-column subtraction. Each node in this trace corresponds

[1] DAEDALUS borrows the notion of flexible means-ends analysis from Jones' (1989) EUREKA system, which used a very similar idea with a different retrieval method.

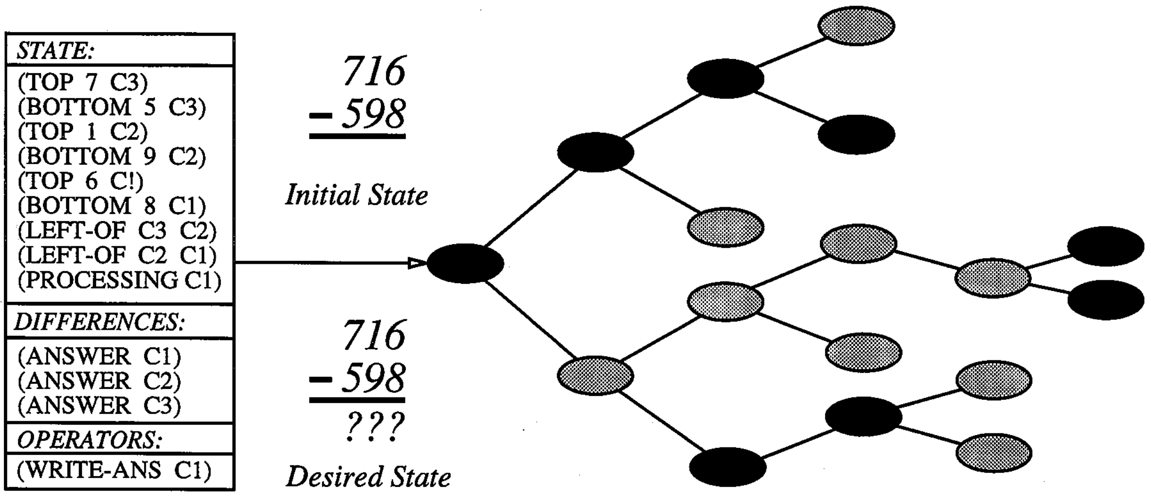


Figure 1: A multi-column subtraction problem, along with a problem-solving trace generated by DÆDALUS. Each node consists of a state description, a set of differences, and the selected operator. Black nodes correspond to problems on which the system initially selected the incorrect operator; gray nodes specify problems on which it made the right selection.

to a problem or subproblem that is described as a set of state predicates, a set of differences, and the operator selected to solve it. DÆDALUS constructs such traces from left to right and from top to bottom. For instance, it first selects an operator for the overall (leftmost) problem, which creates the two subproblems immediately to its right. The system then selects an operator for the first (topmost) subproblem, generates subsubproblems, and recurses. After solving this subproblem, it tentatively applies the original operator and attempts to solve the second subproblem of the original task. In situations where DÆDALUS cannot solve a subproblem, it backtracks and selects another operator. However, user-specified parameters limit the number of operators it considers at each level and the depth of the resulting trace; these bound the overall amount of search.

Organization of Plan Knowledge

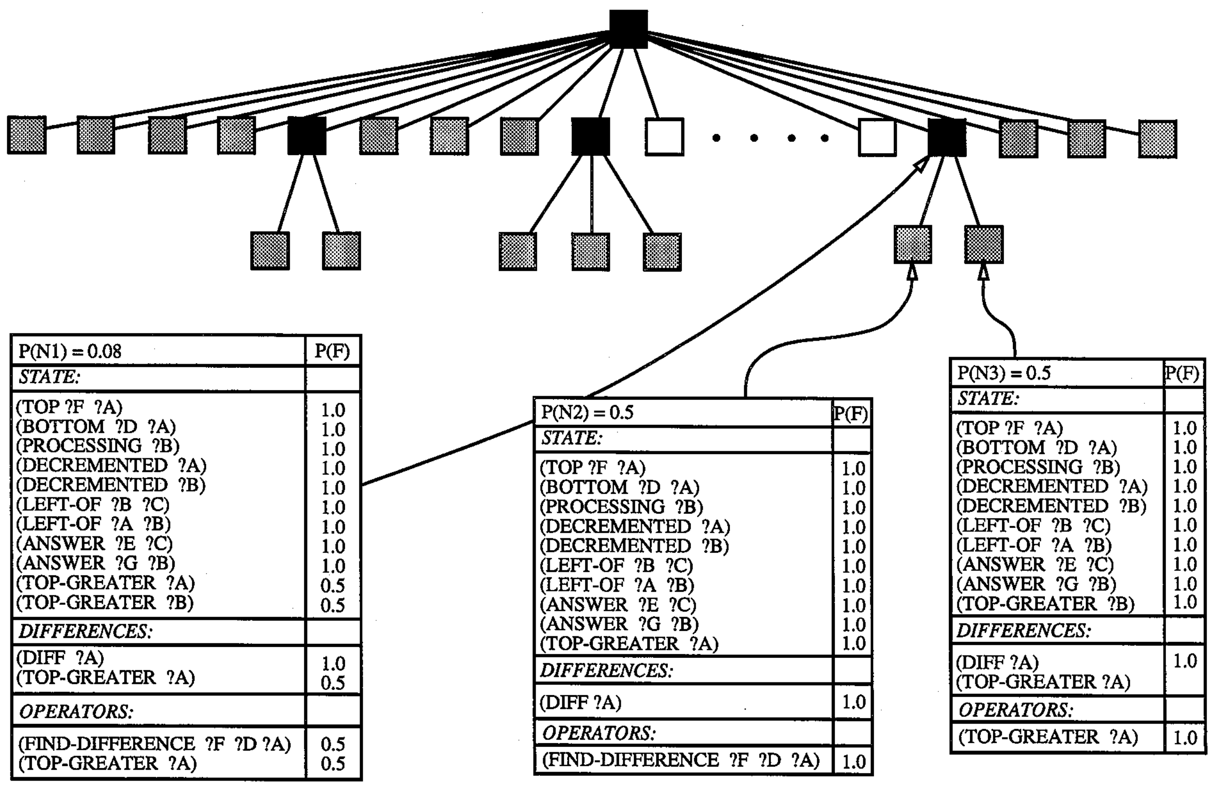
One common approach to encoding plan knowledge involves the use of abstract rules or schemas. For instance, Minton et al.'s (1989) PRODIGY uses abstract search-control rules and Mooney's (1990) EGGS employs general plan schemas. Each rule or schema covers many specific situations, letting these systems use a simple matching or unification algorithm to determine their applicability. Another approach encodes knowledge as specific cases from the domain, including particular problems or subproblems, desirable and undesirable approaches to these problems, and possibly the reasons for their desirability. Researchers in this *case-based* paradigm have proposed a variety of methods for using this information (e.g., Carbonell & Veloso, 1988; Hammond, 1990; Jones, 1989), many of them with direct mapping to techniques that assume abstractions. This approach relies on more sophisticated matching schemes than needed for abstract knowledge, often requiring relational partial matching (i.e., structural analogy).

However, anecdotal evidence suggests that humans store both cases and abstractions in long-term memory. Early knowledge of a domain takes the form of specific problem-solving traces, which must be accessed through some form of analogical retrieval. Later, additional experience forms the basis for abstract schemas, which 'blur together' a number of similar cases. DÆDALUS takes such an integrated view, storing both cases and abstractions in a single probabilistic concept hierarchy. Figure 2 shows the probabilistic hierarchy for the subtraction domain after DÆDALUS has incorporated its experience with the problem from Figure 1. The terminal nodes in gray represent components of the problem-solving trace generated during solution of this task. As we saw earlier, each such trace component corresponds to a problem described as a set of state predicates, a set of differences, and the operator used to solve it.

The figure includes full descriptions for two of these cases (nodes N2 and N3). The additional terminal nodes (in white) represent the original operator schemas that were already present in memory. The hierarchy also contains some abstractions (in black) that DÆDALUS created during the process of storing the trace components, as we describe later. The figure also shows the full description of one abstraction (node N1), which provides a probabilistic summary of the nodes (N2 and N3) below it. Each such description includes an overall probability of occurrence, together with a conditional probability for each difference, state descriptor, and operator. Briefly, the interpretation of each node is that, given the probabilistically specified state descriptors and differences, one should select the operator with the highest probability. This is the form of knowledge that DÆDALUS uses to constrain operator retrieval and thus to direct search during planning.

Constraining Search with Knowledge

Another basic finding is that experts use their knowledge of a domain to reduce or eliminate search (Chi, Glaser, & Rees, 1982). Early research on planning and problem solving (e.g., Newell et al., 1960; Fikes et al., 1971) had little to say about this topic, and in time a separate tradition emerged that focused on the retrieval of relevant plans or components from memory (e.g., Hammond, 1990). This work has emphasized solution of familiar problems using



P(N1) = 0.08	P(F)
STATE:	
(TOP ?F ?A)	1.0
(BOTTOM ?D ?A)	1.0
(PROCESSING ?B)	1.0
(DECREMENTED ?A)	1.0
(DECREMENTED ?B)	1.0
(LEFT-OF ?B ?C)	1.0
(LEFT-OF ?A ?B)	1.0
(ANSWER ?E ?C)	1.0
(ANSWER ?G ?B)	1.0
(TOP-GREATER ?A)	0.5
(TOP-GREATER ?B)	0.5
DIFFERENCES:	
(DIFF ?A)	1.0
(TOP-GREATER ?A)	0.5
OPERATORS:	
(FIND-DIFFERENCE ?F ?D ?A)	0.5
(TOP-GREATER ?A)	0.5

P(N2) = 0.5	P(F)
STATE:	
(TOP ?F ?A)	1.0
(BOTTOM ?D ?A)	1.0
(PROCESSING ?B)	1.0
(DECREMENTED ?A)	1.0
(DECREMENTED ?B)	1.0
(LEFT-OF ?B ?C)	1.0
(LEFT-OF ?A ?B)	1.0
(ANSWER ?E ?C)	1.0
(ANSWER ?G ?B)	1.0
(TOP-GREATER ?A)	1.0
DIFFERENCES:	
(DIFF ?A)	1.0
OPERATORS:	
(FIND-DIFFERENCE ?F ?D ?A)	1.0

P(N3) = 0.5	P(F)
STATE:	
(TOP ?F ?A)	1.0
(BOTTOM ?D ?A)	1.0
(PROCESSING ?B)	1.0
(DECREMENTED ?A)	1.0
(DECREMENTED ?B)	1.0
(LEFT-OF ?B ?C)	1.0
(LEFT-OF ?A ?B)	1.0
(ANSWER ?E ?C)	1.0
(ANSWER ?G ?B)	1.0
(TOP-GREATER ?B)	1.0
DIFFERENCES:	
(DIFF ?A)	1.0
(TOP-GREATER ?A)	
OPERATORS:	
(TOP-GREATER ?A)	1.0

Figure 2: A DÆDALUS concept hierarchy that incorporates cases (gray) and abstractions (black) resulting from storage of components from a problem-solving trace for subtraction, along with the original operator schemas (white) for this domain.

stored knowledge, but downplayed results about problem solving in novel domains, where search plays a major role.

DÆDALUS attempts to model the importance of both search and knowledge in planning. The system operates within a problem-space framework, using domain-specific knowledge stored in its probabilistic concept hierarchy to constrain search when available. However, upon encountering new problems for which it has little knowledge, DÆDALUS gracefully falls back on a more search-intensive approach to plan generation. In both cases, the hierarchy plays the same role for DÆDALUS as does the table of connections for Newell et al.'s GPS (1960), letting it select operators for flexible means-ends analysis.

In order to select an operator, the system invokes COBWEB$_R$, a variant of Fisher's (1987) COBWEB that carries out heuristic classification on relational descriptions. The COBWEB$_R$ module accepts a problem description, which consists of the current state and a set of differences, and sorts it through the concept hierarchy in an attempt to retrieve the most relevant piece of knowledge to bias operator selection. As Allen and Thompson (1991) describe in detail, the system uses the conditional probabilities stored with each node to carry out a heuristic search through the space of partial matches, attempting to find the best match in each case. COBWEB$_R$ invokes a reduced version of *category utility*, the evaluation function used in Fisher's system, to guide search through the space of matches.

After determining a best match between the problem description and each node at a given level of the hierarchy, COBWEB$_R$ selects the node with the highest-scoring match and then recurses to the next level. It continues in this manner until it reaches a terminal node, then selects the operator associated with that node and uses the partial match for the node to determine the operator's bindings. The usefulness of operators retrieved in this manner is related to the amount of knowledge stored in long-term memory. If DÆDALUS has access only to the operators for a domain, it will often select an unprofitable one, leading to backtracking or failure. However, if memory contains detailed knowledge about the situations that occur in a domain, the system is more likely to select an operator on the solution path, reducing search and increasing success rate.

Acquisition of Plan Knowledge

Human problem solvers learn from experience, reducing their search as they become familiar with a domain. Most recent work on learning in planning domains has taken an analytical approach (e.g., Laird, Rosenbloom, & Newell, 1986; Minton et al., 1989; Mooney, 1990), which involves compiling existing knowledge into new forms. Although elegant, such mechanisms tend to predict faster learning than occurs in humans, who must often work through a problem many times to eliminate search (Anzai & Simon, 1979). Anderson (1983) describes one response, which involves combining an analytical learning mechanism with an empirical technique for strengthening rules. DÆDALUS takes a different approach, employing an inductive learning method to index cases and to construct probabilistic abstractions. Some research has focused on inductive approaches to learning search-control knowledge (e.g., Langley, 1985), but unlike earlier work on this topic, DÆDALUS supports the acquisition of probabilistic plan knowledge. Fisher and Langley (1990) review psychological evidence for probabilistic representations of concepts, and we believe that plan knowledge is stored in the same manner.

Learning in DÆDALUS occurs whenever the system finds a solution to a problem or subproblem, at which point it incorporates the problem description and the successful operator into long-term memory. For instance, each of the nodes from the problem-solving trace in Figure 1 would be sorted through the concept hierarchy and stored in the hierarchy. Like the COBWEB system on which it is based, DÆDALUS makes use of two main learning operations. If a trace component reaches a terminal node in memory, the COBWEB$_R$ module extends the hierarchy downward, creating a new node N that is a probabilistic summary of the case and the terminal node, and making them both children of N. In contrast, if the trace component is summarized poorly by all the children of node K, COBWEB$_R$ creates a new child of K based on the case. During learning, this module uses the full version of category utility to determine when a new branch is justified; this differs from retrieval, in which problems are always sorted to terminal nodes.

In addition, when in learning mode, the system also considers merging and splitting existing concepts, in an attempt to minimize effects due to training order. Again, COBWEB$_R$ uses the full version of its evaluation function to select between these learning operations and those described above. If the routine decides to incorporate a training case into a given node N, it simply updates the probability for N and the conditional probabilities for each of the features (both problem descriptors and operators) associated with it. If a feature occurs in the training instance that is not present in the node, COBWEB$_R$ adds it to the node's description with a low probability. This averaging process occurs for each node through which the instance passes. One important point is that DÆDALUS incorporates each plan component into long-term memory during problem solving. This scheme is consistent with the incremental nature of human learning.

DÆDALUS on Multi-column Subtraction

In the previous section we referred to an example involving multi-column subtraction; here we consider DÆDALUS' behavior in this domain in more detail. We focus on subtraction tasks because they have educational relevance, they have been widely studied, and they can be clearly defined yet cause difficulty for many students. The task used in Figure 1, 716 – 598, provides a relatively simple example. Given two rows of digits, one must find their overall difference – in this case 118 – and write this in a third row. Tasks that involve borrowing (i.e., in which the top number in a column is smaller than the lower one) require more operators and are more difficult for humans; this holds especially for tasks containing zeroes in the top row.

For this domain we initialized DÆDALUS' memory with nine operators, each defined in terms of preconditions, add lists, and delete lists, as in STRIPS (Fikes et al., 1971) and in many production-system models (e.g., Anderson, 1983). Three of the operators – FIND-DIFFERENCE, FIND-TOP, and SKIP-ZERO – are responsible for finding the answer associated with a particular column, but are applicable under different conditions and involve slightly different actions. Two other operators – ADD-TEN and DECREMENT – implement borrowing, whereas SHIFT-LEFT and SHIFT-RIGHT change the column on which attention is focused. Finally, the operators WRITE-ANSWER and TOP-GREATER make inferences that support the solution process.

The operators are quite similar to those used by Langley, Wogulis, and Ohlsson (1990) to model subtraction errors, which in turn were based on the rules used in error models proposed by Young and O'Shea (1981). However, this earlier work assumed that students solved subtraction problems in a forward-chaining manner. This runs counter both to intuitions about borrowing and to strategies presented by expert teachers (S. Ohlsson, personal communication), which suggest that means-ends reasoning occurs in multi-column subtraction tasks. DÆDALUS employs its knowledge in this fashion, selecting operators (e.g., FIND-DIFFERENCE) that would make progress toward the desired state, noting that preconditions are not met, and then selecting other operators (e.g., ADD-TEN) to achieve these preconditions. This framework also differs from the problem-reduction scheme used by Brown and Burton (1978) and by VanLehn (1990). Both approaches decompose problems into subproblems, but problem-reduction methods specify such decompositions in advance, whereas DÆDALUS generates them dynamically.

Given the correct operators for multi-column subtraction, DÆDALUS cannot explain the errors commonly observed in students' behavior on this domain (Brown & Burton, 1978; VanLehn, 1990). Because the system only applies operators when their preconditions have been met, its answers are guaranteed to be correct. However, unlike the previous models of subtraction behavior, DÆDALUS may require search to solve a problem even when its operators are correct; this occurs because it may not *select* the best operator on its first attempt. Combined with the system's constrained search algorithm, which considers only a few alternatives at each level, this means DÆDALUS may fail entirely to solve certain problems. This differs from the behavior observed in many students who, upon encountering a difficult problem, produce syntactically correct answers with incorrect digits. In future work, we plan to extend DÆDALUS to generate abstract plans that ignore the violation of certain operator preconditions. The execution of abstract plans would explain some catalogued subtraction errors, such as students' tendency to subtract the top number from the lower one when the latter is larger.

Although DÆDALUS does not model subtraction errors, we believe it provides a viable model of learning in this domain. To demonstrate this, we presented the system with worked solutions to eight problems taken from VanLehn (1990, p. 55), in each case providing DÆDALUS with a problem-solving trace like that in Figure 1. The system stored each component of these traces in long-term memory for use on future tasks. We then presented DÆDALUS with the original tasks and a separate set of twelve test problems taken from the same source, which it had to solve in the absence of further learning. For comparison, we also let the system work on the same problems with no knowledge except the correct operators. Table 1 gives the results: DÆDALUS solved all of the training problems after learning, but poor operator selection let it solve only one of them before learning. Similarly, it solved two of the twelve test problems after learning but none before, showing modest transfer. Moreover, the problems solved after learning were handled with no search. Thus, the system suggests one

Table 1: DÆDALUS' behavior on multi-column subtraction problems taken from VanLehn (1990).

	PROBLEMS SOLVED	
	TRAINING	TEST
BEFORE LEARNING	12%	0%
AFTER LEARNING	100%	17%

possible mechanism through which students improve their ability by training on worked sample problems. However, our purpose here is not to claim that DÆDALUS provides a better model of subtraction behavior than other frameworks, but simply to demonstrate that it supports learning in an interesting domain. Elsewhere (Langley & Allen, in press) we report similar results on the blocks world.

Psychological Adequacy of DÆDALUS

Earlier we noted some high-level aspects of human cognition that are reflected in DÆDALUS, but this was only in passing. Now let us reconsider the system's psychological plausibility in more detail, drawing on VanLehn's (1989) excellent review of the major findings with respect to human problem solving. These phenomena are qualitative in nature, but they still provide constraints on the operation of cognitive simulations. Table 2 lists most of the behaviors that VanLehn reports which are directly related to problem solving, along with others noted by Jones (1989). The table also shows how DÆDALUS fares relative to three other models of problem solving and learning: Anderson's (1983) ACT, Laird, Rosenbloom, and Newell's (1986) SOAR, and Jones' (1989) EUREKA. Langley and Allen (in press) present a more extensive comparison.

The first three phenomena address issues about basic problem-solving strategies rather than learning. Both DÆDALUS and EUREKA incorporate a version of means-ends analysis, which humans appear to use in novel domains; in contrast, SOAR can produce means-ends behavior with preference rules, but the process is not built into the architecture, and ACT provides support for backward chaining but not true means-ends analysis. Of the four frameworks, only EUREKA mimics the nonsystematic strategy of humans, who often explore a search path in depth, then return to the initial state to consider an alternative (Newell & Simon, 1972). All four architectures should account for the relative difficulty of problem isomorphs (Kotovsky, Hayes, & Simon, 1985), but this ability relies on representational assumptions outside the systems themselves.

Some additional behaviors concern changes in performance as humans gain experience in a problem-solving domain. The most basic finding is that learning leads to reduced search on a class of problems. As we report elsewhere (Langley & Allen, in press), DÆDALUS generally carries out less search with experience, as do EUREKA, ACT, and SOAR, although the latter two employ explanation-based learning methods, Jones' system relies entirely on analogical reasoning, and DÆDALUS uses concept formation. The different architectures also differ in their rate of learning. Another phenomenon involves the asymmetry of *transfer* across problems, in which experience on difficult tasks aids the solution of simpler problems more than the reverse situation. Presumably this occurs because the structures needed for the simpler task are subsumed by the

Table 2: Psychological adequacy of four models of learning in problem-solving domains in terms of whether they account ($\oplus$), fail to account ($\ominus$), or partially account ($\odot$) for phenomenon from VanLehn ($*$) and Jones ($\diamond$).

	ACT	SOAR	EUREKA	DÆDALUS
Means-ends$^{\diamond}$	$\odot$	$\oplus$	$\oplus$	$\oplus$
Nonsystematic$^{\diamond}$	$\ominus$	$\ominus$	$\oplus$	$\ominus$
Isomorphs*	$\odot$	$\odot$	$\odot$	$\odot$
Reduced search*	$\oplus$	$\oplus$	$\oplus$	$\oplus$
Asymmetries*	$\oplus$	$\oplus$	$\oplus$	$\oplus$
Einstellung*	$\oplus$	$\oplus$	$\oplus$	$\oplus$
Verbalization*	$\oplus$	$\oplus$	$\ominus$	$\ominus$
Reduced time*	$\oplus$	$\oplus$	$\ominus$	$\ominus$
Rare analogy*	$\ominus$	$\ominus$	$\odot$	$\odot$
Superficiality*	$\ominus$	$\ominus$	$\oplus$	$\oplus$

more difficult one. Since all four models decompose problems into subproblems, then learn methods for solving these subproblems, all should produce this result. Another well-established transfer effect, *Einstellung* (Luchins, 1942), occurs when one is trained on problems with complex solutions and then given problems with analogous solutions but also with simpler ones. Under such conditions, subjects inevitably solve the new problems in the complex way that has worked in the past. EUREKA and ACT have been explicitly shown to produce this behavior, and we expect SOAR and DÆDALUS to generate similar results.

In addition, the skills of experienced problem solvers are more *automatized* than those of novices, in that they can carry them out with little attention. Experts typically solve problems much more rapidly, even when their solutions involve the same number of steps, and they tend to verbalize much less, suggesting that they have lost access to intermediate subproblems. Both DÆDALUS and EUREKA have difficulty explaining these phenomena, in that they never change the steps taken in generating a solution; learning may eliminate poor choices, but each node in the problem-solving trace must still be constructed one step at a time. In contrast, ACT and SOAR actually eliminate subproblems through learning, which explains the reduction in verbalization and some of the observed speedup.

A final set of empirical results concern problem solving by analogy. Experiments reveal that spontaneous cross-domain analogy is quite rare (e.g., Gick & Holyoak, 1980). People can solve problems by analogy when given an explicit mapping between source and target problems, but they cannot always find such a mapping on their own. Moreover, in cases of spontaneous retrieval, the reminding is usually based on some superficial, surface similarity that may produce a misleading analogy (e.g., Ross, 1984). EUREKA relies on a form of analogical retrieval that operates on surface-level descriptions, and Jones (1989) has shown that it replicates the basic finding that experience with one problem increases the likelihood of solving an analogous problem, but that spontaneous analogy across domains is far from guaranteed. Because DÆDALUS also uses a form of analogical retrieval, it should produce similar results, but (as with EUREKA) only provided it is given states and operators that share some surface features across the domains. ACT and SOAR have more difficulty with these phenomena, since neither has any architectural mechanism for analogy, though both could mimic analogy using explicit rules.

Conclusions

In the previous pages we described DÆDALUS, a planning system that was designed to be consistent with knowledge of human problem-solving behavior. We found that the system directs problem-space search using a flexible version of means-ends analysis, organizes both cases and abstractions in a probabilistic concept hierarchy, employs domain knowledge to constrain its generation of plans, and acquires plan knowledge through an incremental learning process. In particular, DÆDALUS retrieves relevant operators by sorting new problems through its concept hierarchy, and it stores the components of successful plans in this hierarchy for future retrieval. We demonstrated the system's behavior in the domain of multi-column subtraction.

We also examined DÆDALUS' ability to explain aspects of human cognition at a qualitative level, finding that the system is consistent with a variety of robust phenomena that have been observed in human problem solving. However, three previous models also explain roughly the same behaviors. DÆDALUS differs from Laird et al.'s SOAR and Anderson's ACT in its partial coverage of analogical reasoning, an area it shares with Jones' EUREKA system. However, DÆDALUS fails to explain the reduction of verbalization and the automatization observed in highly-skilled problem solvers, and its search organization does not mimic the nonsystematic behavior found in human problem solving, which only EUREKA has attempted to handle.

In addition, the system fails in the broader sense that humans are physical agents who interleave planning with other processes. A fuller model of human behavior would explicitly link cognition with action and perception. Such issues arise even in constrained domains like subtraction, in which students cannot hold the entire problem in short-term memory, and thus must perceive and alter a physical display. Elsewhere (Langley & Allen, in press) we have described our ideas for extending DÆDALUS along these lines, and developing this augmented model remains an important direction for future research.

Acknowledgements

We thank Randy Jones, Doug Fisher, Steve Minton, Wayne Iba, Kevin Thompson, John Gennari, Kathleen McKusick, and Deepak Kulkarni for lengthy discussions that led to many of the ideas in this paper.

References

Allen, J. A., & Thompson, K. (1991). Probabilistic concept formation in relational domains. *Proceedings of the Eighth International Workshop on Machine Learning*. Evanston, IL: Morgan Kaufmann.

Anderson, J. R. (1983). *The architecture of cognition*. Cambridge: Harvard University Press.

Anzai, Y., & Simon, H. A. (1979). The theory of learning by doing. *Psychological Review*, *86*, 124–140.

Brown, J. S., & Burton, R. R. (1978). Diagnostic models for procedural bugs in basic mathematical skills. *Cognitive Science*, *2*, 155–192.

Carbonell, J. G., & Veloso, M. (1988). Integrating derivational analogy into a general problem solving architecture. *Proceedings of the DARPA Workshop on Case-based Reasoning* (pp. 104–121). Clearwater Beach, FL: Morgan Kaufmann.

Chi, M. T. H., Glaser, R., & Rees, E. (1982). Expertise in problem solving. In R. J. Sternberg (Ed.), *Advances in the psychology of human intelligence*. Hillsdale, NJ: Lawrence Erlbaum.

Fikes, R. E., Hart, P. E., & Nilsson, N. J. (1971). STRIPS: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence*, *2*, 189–208.

Fisher, D. H. (1987). Knowledge acquisition via incremental conceptual clustering. *Machine Learning*, *2*, 139–172.

Fisher, D. H., & Langley, P. (1990). The structure and formation of natural categories. In G. H. Bower (Ed.), *The psychology of learning and motivation: Advances in Research and Theory* (Vol. 26). Cambridge, MA: Academic Press.

Gick, M. L., & Holyoak, K. J. (1980). Analogical problem solving. *Cognitive Psychology*, *12*, 306–355.

Hammond, K. J. (1990). Case-based planning: A framework for planning from experience. *Cognitive Science*, *14*, 385–443.

Jones, R. (1989). *A model of retrieval in problem solving*. Doctoral dissertation, Department of Information & Computer Science, University of California, Irvine.

Kotovsky, K., Hayes, J. R., & Simon, H. A. (1985). Why are some problems hard? Evidence from tower of Hanoi. *Cognitive Psychology*, *17*, 248–294.

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

Langley, P. (1985). Learning to search: From weak methods to domain-specific heuristics. *Cognitive Science*, *9*, 217–260.

Langley, P., Wogulis, J., & Ohlsson, S. (1990). Rules and principles in automated cognitive diagnosis. In N. Fredericksen, R. Glaser, A. Lesgold, & M. G. Shafto (Eds.), *Diagnostic monitoring of skill and knowledge acquisition*. Hillsdale, NJ: Lawrence Erlbaum.

Langley, P., & Allen, J. A. (in press). A unified framework for planning and learning. In S. Minton (Ed.), *Learning and planning*. San Mateo, CA: Morgan Kaufmann.

Luchins, A. S. (1942). Mechanization in problem solving: The effect of Einstellung. *Psychological Monographs*, *54* (248).

Minton, S., Carbonell, J. G., Knoblock, C. A., Kuokka, D. R., Etzioni, O., & Gil, Y. (1989). Explanation-based learning: A problem solving perspective. *Artificial Intelligence*, *40*, 63–118.

Mooney, R. J. (1990). *A general explanation-based learning mechanism and its application to narrative understanding*. San Mateo, CA: Morgan Kaufmann.

Newell, A. (1980). Reasoning, problem solving, and decision processes: The problem space hypothesis. In R. Nickerson (Ed.), *Attention and performance VIII*. Hillsdale, NJ: Lawrence Erlbaum.

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

Ross, B. H. (1984). Remindings and their effects in learning a cognitive skill. *Cognitive Psychology*, *16*, 371–416.

VanLehn, K. (1989). Problem solving and cognitive skill acquisition. In M. I. Posner (Ed.), *Foundations of cognitive science*. Cambridge, MA: MIT Press.

VanLehn, K. (1990). *Mind bugs: The origins of procedural misconceptions*. Cambridge, MA: MIT Press.

Young, R. M., & O'Shea, T. (1981). Errors in children's subtraction. *Cognitive Science*, *5*, 153–177.