Derivational Analogy and its Role in Problem Solving

Jaime G. Carbonell
Computer Science Department,
Carnegie-Mellon University,
Pittsburgh, PA 15213

Abstract

Derivational analogy, a method of solving problems based upon the transfer of past experience to new problem situations, is discussed in the context of other general approaches to problem solving. The experience transfer process consists of recreating lines of reasoning, including decision sequences and accompanying justifications, that proved effective in solving particular problems requiring similar initial analysis. The derivational analogy approach is advocated as a means of implementing reasoning from individual cases in expert systems.[1]

1. Introduction: The Role of Analogy in Problem Solving

The term "problem solving" in artificial intelligence has been used to denote disparate forms of intelligent action to achieve well-defined goals. Perhaps the most common usage stems from Newell and Simon's work [22] in which problem solving consists of selecting a sequence of operators (from a pre-analyzed finite set) that transforms an initial problem state into a desired goal state. Intelligent behavior consists of a focused search for a suitable operator sequence by analyzing the states resulting from the application of different operators to earlier states.[2] Many researchers have adopted this viewpoint [12, 26, 23].

However, a totally different approach has been advocated by McDermott [19] and by Wilensky [32, 33] that views problem solving as plan instantiation. For each problem posed there are one or more plans that outline a solution, and problem solving consists of identifying and instantiating these plans. In order to select, instantiate, or refine plans, additionals plans that tell how to instantiate other plans or how to solve subproblems are brought to bear in a recursive manner. Traditional notions of search are totally absent from this formulation. Some systems, such as the counterplanning mechanism in POLITICS [6, 3], provide a hybrid approach, instantiating plans whenever possible, and searching to construct potential solutions in the absence of applicable plans.

A third approach is to solve a new problem by analogy to a previously solved similar problem. This process entails searching for related past problems and transforming their solutions into ones potentially applicable to the new problem [24]. I developed and advocated such a method [7, 8] primarily as a means of bringing to bear problem solving expertise acquired from past experience. The analogical transformation process itself may require search, as it is seldom immediately clear how a solution to a similar problem can be adapted to a new situation.

A useful means of classifying different problem solving methods is to compare them in terms of the amount and specificity of domain knowledge they require.

- If no structuring domain knowledge is available and there is no useful past experience to draw upon, weak methods such as heuristic search and means-ends analysis are the only tools that can be brought to bear. Even in these knowledge-poor situations, information about goal states, possible actions, their known preconditions and their expected outcomes is required.
- If specific domain knowledge in the form of plans or procedures exists, such plans may be instantiated directly, recursively solving any subproblems that arise in the process.
- If general plans apply, but no specific ones do so, the general plans can be used to reduce the problem (by partitioning the problem or providing islands in the search space). For instance, in computing the pressure at a particular point in a fluid statics problem, one may use the general plan of applying the principle of equilibrium of forces on the point of interest (the vector sum of the forces = 0). But, the application of this plan only reduces the original problem to one of finding and combining the appropriate forces, without hinting how that may be accomplished in a specific problem [5].
- If no specific plans apply, but the problem resembles one solved previously, apply analogical transformation to adapt the solution of that similar past problem the new situation. For instance, in some studies it has proven easier for students to solve mechanics problems by analogy to simpler solved problems than by appealing to first principles or by applying general procedures presented in a physics text [9]. As an example of analogy involving composite skills rather than pure cognition, consider a person who knows how to drive a car and is asked to drive a truck. Such a person may have no general plan or procedure for driving trucks, but is likely to perform most of the steps correctly by transferring much of his or her automobile driving knowledge. Would that we had robots that were so self-adaptable to new, if recognizably related tasks!

Clearly, these problem solving approaches are not mutually exclusive; for instance, one approach can be used to reduce a problem to simpler subproblems, which can in turn be solved by the otner methods. In fact, Larkin and I [5] are developing a general inference engine for problem solving in the natural sciences that combines all four approaches.

[1] This research was supported by the Office of Naval Research (ONR) under grant numbers N00014-79-C-0661 and N00014-82-C-50767.

[2] In means-ends analysis, the current state is compared to the goal state and one or more operators that reduce the difference are selected, whereas in heuristic search, the present state is evaluated in isolation and compared to alternate states resulting from the application of different operators (to states generated earlier in the search), and the search for a solution continues from the highest-rated state.

As discussed earlier, only direct plan instantiation and weak methods have received substantial attention by AI practitioners. For instance, Newell and Laird's recent formulation of a universal weak method [17] as a general problem solving engine is developed completely within the search paradigm. Expert systems, for the most part, combine aspects of plan instantiation (often broken into small rule-size chunks of knowledge) and heuristic search in whatever manner best exploits the explicit and implicit constraints of the specific domain [30, 11, 20]. I am more concerned with the other two approaches, as the they could conceivably provide powerful reasoning mechanisms not heretofore analyzed in the context of automating problem-solving processes. The rest of this paper focuses on a new formulation of the analogical problem solving approach.

2. Analogy and Experiential Reasoning

The term *analogy* often conjures up recollections of artificially contrived problems asking: "X is to Y as Z is to ?" in various psychometric exams. This aspect of analogy is far too narrow and independent of context to be useful in general problem solving domains. Rather, I propose the following operational definition of analogical problem solving consistent with past AI

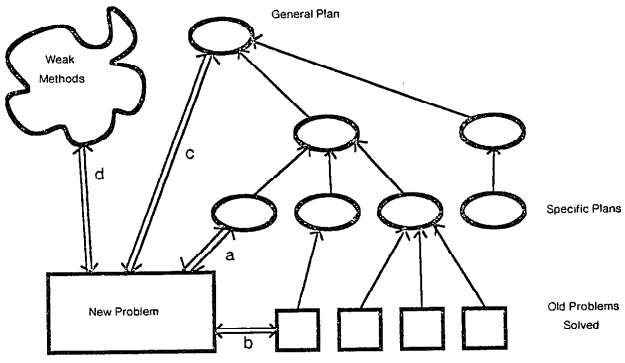

Figure 1-1: Problem solving may occur by a) instantiating specific plans, b) analogical transformation to a known solution of a similar problem, c) applying general plans to reduce the problem, d) applying weak methods to search heuristically for a possible solution, or e) a combination of these approaches.

research efforts [16, 34, 35, 13, 4, 8].

> **Definition:** *Analogical problem solving consists of transferring knowledge from past problem solving episodes to new problems that share significant aspects with corresponding past experience.*

In order to make this definition operational, the problem solving method must specify:

- what it means for problems to "share significant aspects",
- what knowledge is transferred from past experience to the new situation,
- precisely how the knowledge transfer process occurs,
- and how analogically related experiences are selected from a potentially vast long term memory of past problem solving episodes.

The remainder of this paper discusses two major approaches to analogical problem solving I have analyzed in terms of these four criteria. The first approach has been successfully implemented in ARIES (Analogical Reasoning and Inductive Experimentation System), and we are actively experimenting with the other approach. This short paper focuses on a comparative analysis of the two methods, rather than discussing implementation techniques or examining our preliminary empirical results.

2.1. Analogical Transformation of Past Solutions

If a particular solution has been found to work on a problem similar to the one at hand, perhaps it can be used, with minor modification, for the present problem. By "solution" I mean only a sequence of actions that if applied to the initial state of a problem brings about its goal state. Simple though this process may appear, an effective computer implementation requires that many difficult issues be resolved, to wit:

1. Past problems descriptions and their solutions must be remembered and indexed for later retrieval.
2. The new problem must be matched against large numbers of potentially relevant past problems to find closely related ones, if any. An operational similarity metric is required as a basis for selecting the most suitable past experiences.
3. The solution to a selected old problem must be transformed to satisfy the requirements of the new problem statement.

In order to achieve these objectives, the initial analogical problem solver [8] required a partial matcher with a built-in similarity criterion, a set of possible transformations to map the solution of one problem into the solution to a closely related problem, and a memory indexing mechanism based on a MOPS-like memory encoding of events and actions [28]. The solution transformation process was implemented as a set of primitive transform operators and a means-ends problem solver that searched for sequences of primitive transformations yielding a solution to the desired problem. The resultant system, called ARIES-I, turned out to be far more complex than originally envisioned. Partial pattern matching of problem descriptions and searching in the space of solution transformations are difficult tasks in themselves.

In terms of the four criteria, the solution transformation process may be classified as follows:

1. Two problems share significant aspects if they match within a certain preset threshold in the initial partial matching process, according to the built-in similarity metric.
2. The knowledge transferred to the new situation is the sequence of actions from the retrieved solution, whether or not that sequence is later modified in the analogical mapping process.
3. The knowledge transfer process is accomplished by copying the retrieved solution and perturbing it incrementally according to the primitive transformation steps in a heuristically guided manner until it satisfies the requirements of the new problem. (See [8] for details.)
4. The selection of relevant past problems is constrained by the memory indexing scheme and the partial pattern matcher.

Since a significant fraction of problems encountered in mundane situations and in areas requiring significant domain expertise (but not in abstract mathematical puzzles) bear close resemblance to past solved problems, the ARIES-I method proved effective when tested in various domains, including algebra problems and route planning tasks. An experiential

learning component was added to ARIES that constructed simple plans (generalized sequences of actions) for recurring classes of problems, hence allowing the system to solve new problems in this class by the more direct plan instantiation approach. However, no sooner was the solution transformation method implemented and analyzed than some of its shortcomings became strikingly apparent. In response to these deficiencies, I started analyzing more sophisticated methods of drawing analogies, as discussed in the following sections.

3. The Derivational Analogy Method

In formulating plans and solving problems, a considerable amount of intermediate information is produced in addition to the resultant plan or specific solution. For instance, formulation of subgoal structures, generation and subsequent rejection of alternatives, and access to various knowledge structures all typically take place in the problem solving process. But, the solution transformation method outlined above ignores all such information, focusing only upon the resultant sequence of actions and disregarding, among other things, the reasons for selecting those actions. Why should one take such extra information into account? It would certainly complicate the analogical problem solving process, but what benefits would accrue from such an endeavor? Perhaps the best way to answer this question is by analysis of where the simple solution transformation process falls short and how such problems may be alleviated or circumvented by preserving more information from which qualitatively different analogies may be drawn.

3.1. The Need for Preserving Derivation Histories

Consider, for instance, the domain of constructing computer programs to meet a set of pre-defined specifications. In the automatic programming literature, perhaps the most widely used technique is one of progressive refinement [2, 1, 15]. In brief, progressive refinement is a multi-stage process that starts from abstract specifications stated in a high level language (typically English or some variant of first order logic), and produces progressively more operational or algorithmic descriptions of the specification committing to control decisions, data structures and eventually specific statements in the target computer language. However, humans (well, at least this writer) seldom follow such a long painstaking process, unless perhaps the specifications call for a truly novel program unlike anything in one's past experience. Instead, a common practice is to recall similar past programs and reconstruct the new programming problem along the same directions. For instance, one should be able to program a quicksort algorithm in LISP quite easily if one has recently implemented quicksort in PASCAL. Similarly, writing LISP programs that perform tasks centered around depth-first tree traversal (such as testing equality of S-expressions or finding the node with maximal value) are rather trivial for LISP programmers but surprisingly difficult for those who lack the appropriate experience.

The solution transformation process proves singularly inappropriate as a means of exploiting past experience in such problems. A PASCAL implementation of quicksort may look very different than a good LISP implementation. In fact, attempting to transfer corresponding steps from the PASCAL program into LISP is clearly not a good way to produce any LISP program, let alone an elegant or efficient one. Although the two problem statements may have been similar, and the problem solving processes may preserve much of the inherent similarity, the resultant solutions (i.e., the PASCAL and LISP programs) may bear little if any direct similarities.

The useful similarities lie in the algorithms implemented and in the set of decisions and internal reasoning steps required to produce the two programs. Therefore, the analogy must take place at earlier more abstract stages of the original PASCAL implementation, and it must be guided by a reconsideration of the key decisions in light of the new situation. In particular, the derivation of the LISP quicksort program starts from the same specifications, keeping the same divide and conquer strategy, but may diverge in selecting data structures (e.g. lists vs arrays), or in the method of choosing the comparison element, depending on the tools available in each language and their expected efficiency. However, future decisions (e.g. whether to recurse or iterate, what mnemonics to use as variable names, etc.) that do not depend on earlier divergent decisions can still be transferred to the new domain rather than recomputed. Thus, the derivational analogy method walks through the reasoning steps in the construction of the past solution and considers whether they are still appropriate in the new situation or whether they should be reconsidered in light of significant differences between the two situations.

The difference between the solution transformation approach and the derivational analogy approach just outlined can be stated in terms of the operational knowledge that can be brought to bear. The former corresponds to a person who has has never before programmed quicksort and is given the PASCAL code to help him construct the LISP implementation, whereas the latter is akin to a person who has programmed the PASCAL version himself and therefore has a better understanding of the issues involved before undertaking the LISP implementation. Swartout and Balzer [31] and Scherlis [25] have argued independently in favor of working with program derivations as the basic entities in tasks relating to automatic programming. The advantages of the derivational analogy approach are quite evident in automatic programming because the of the frequent inappropriateness of direct solution transformation, but even in domains whether the latter is useful, one can create problems that demonstrate the need for preserving or reconstructing past reasoning processes.

3.2. The Process of Drawing Analogies by Derivational Transformation

Let us examine in greater detail the process of drawing analogies from past reasoning processes. The essential insight is that useful experience is encoded in the reasoning process used to derive solutions to similar problems, rather than just in the resultant solution. And, a method of bringing that experience to bear in the problem solving process is required in order to make this form of analogy a computationally tractable technique. Here we outline such a method:

1. When solving a problem by whatever means store each step taken in the solution process, including:
   - The subgoal structure of the problem
   - Each decision made (whether a decision to take action, to explore new possibilities, or to abandon present plans), including:
     - Alternatives considered and rejected
     - The reasons for the decisions taken (with dependency links to the problem description or information derived therefrom)
     - The start of a false path taken (with the reason why this appeared to be a promising alternative, and the reason why it proved otherwise, again with dependency links to the problem description. Note that the body of the false path and other resultant information need not be preserved,) only its end points are kept for future reference
     - Dependencies of later decisions on earlier ones in the derivation.
   - Pointers to external knowledge structures that were

accessed and that proved useful in the eventual construction of the solution

- The resultant solution itself
    - In the event that the problem solver proved incapable of solving the problem, the closest approach to a solution should be stored, along with the reasons why no further progress could be made (e.g., a conjunctive subgoal that could not be satisfied).
    - In the event that the solution depends, perhaps indirectly, on volatile assumptions not stated in the problem description (such as the cooperation of another agent, or time-dependent states) store all dependencies to such assumptions made by the problem solver.

2. When a new problem is encountered that does not lend itself to direct plan instantiation or other direct recognition of the solution pattern, start to analyze the problem by applying general plans or weak methods, whichever is appropriate to the situation.
3. If after commencing the analysis of the problem, the reasoning process (the initial decisions made and the information taken into account) parallels that of past problem situations, retrieve the full reasoning traces with the same initial segments and proceed with the derivational transformation process. If such traces are formed, consider the possibility of solution transformation analogy or, failing that, proceed with the present line of non-analogical reasoning.
    - Two problems are considered similar if their analysis results in equivalent reasoning processes, at least in its initial stages. This replaces the more arbitrary context-free similarity metric required for partial matching among problem descriptions in drawing analogies by direct solution transformation. Hence, past reasoning traces (henceforth *derivations*) are retrieved if their initial segments match exactly the first stages of the analysis of the present problem.
    - The retrieved reasoning processes are then used much as individual relevant cases in medicine are used to generate expectations and drive the diagnostic analysis. Reasoning from individual cases has been recognized as an important component of expertise [29], but little has been said of the necessary information that each case must contain. And no simple method has been proposed for retrieving the appropriate cases in a manner that does not rely on arbitrary similarity metrics. Here, I take the stand that reasoning from individual cases require that the stored analysis of these past cases contain a derivational history that justifies all the decisions taken to arrive at the conclusion. It is also neccessary to store pointers to external data that proved useful, list of alternative reasoning paths not taken, and failed attempts (coupled with both reasons for their failure and reasons for having originally made the attempt). Case-based reasoning is nothing more than derivational analogy applied to domains of extensive expertise.
    - It is important to know that although one may view derivational analogy as an interim step in reasoning from particular past experience as more general plans are acquired, it is a mechanism that remains forever useful, since knowledge is always incomplete and exceptions to the best formulated general plans require that the problem solver reason from past individual reasoning episodes.
4. A retrieved derivation is applied to the current situation as follows: For each step in the derivation, starting immediately after the matched initial segment, check whether the reasons for performing that step are still valid by tracing dependencies in the retrieved derivation to relevant parts of the old problem description or to volatile external assumptions made in the initial problem solving.
    - If parts of the problem statement or external assumptions on which the retrieved situation rests are also true in the present problem situation, proceed to check the next step in the retrieved derivation.
    - If there is a violated assumption or problem statement, check whether the decision made would still be justified by a different derivation path from the new assumptions or statements. If so, store the new dependencies and proceed to the next step in the retrieved derivation. The ideas of tracing causal dependences and verifying past inference paths borrow heavily from TMS [10] and some of the non-monotonic logic literature [18]. It should be noted that dependancies and therefore consistancy maintenence is local to each derivation. Hence verifying justifications is constrained to the information computing that proved relevance in solving analogically related problems. There is no notion of "global consistency" or global truth maintenence in this formulation. However, the role played by data dependencies in derivational analogy is somewhat different and more constrained than in maintaining global consistency in deductive data bases.
    - If the old decision cannot be justified by new problem situation,
        - evaluate the alternatives not chosen at that juncture and select an appropriate one in the usual problem solving manner, storing it along with its justifications, or
        - initiate the subgoal of establishing the right supports in order for the old decision to apply in the new problem[3] (clearly, any problem solving method can be brought to bear in achieving the new subgoal), or
        - abandon this derivational analogy in favor of another more appropriate problem solving experience from which to draw the analogy, or in favor of other means of problem solving.
    - If one or more failure paths are associated with the

[3] This approach only works if the missing or violated premise relates to that part of the global state under control of the problem solver, such as acquiring a missing tool or resource, rather than under the control of an uncooperative external agent or a recalcitrant environment. The discussion of strategy-based counterplanning gives a more complete account of subgoaling to rectify unfulfilled expectations [3, 6].

current decision, check the cause of failure and the reasons these alternatives appeared viable in the context of the original problem (by tracing dependency links when required). In the case that their reasons for failure no longer apply, but the initial reasons for selecting these alternatives are still present, consider reconstructing this alternate solution path in favor of continuing to apply and modify the present derivation (especially if quality of solution is more important than problem solving effort).

- In the event that a different decision is taken at some point in the rederivation, do not abandon the old derivation, since future decisions may be independent of some past decisions, or may still be valid (via different justifications) in spite of the somewhat different circumstances. This requires that dependency links be kept between decisions at different stages in the derivation. The lack of an explicit path of links between the decisions indicates that the problem solver believes them to be independent of each other.
- The derivational analogy should be abandoned in the event that a preponderance of the old decisions are invalidated in the new problem situation. Exactly what the perseverance threshold should be is a topic for empirical investigation, as it depends on whether there are other tractable means of solving this problem and on the overhead cost of reevaluating individual past decisions that may no longer be supported and may or may not have independent justification.

5. After an entire derivation has been found to apply to the new problem, store its divergence from the parent derivation as another potentially useful source of analogies, and as an instance from which more general plans can be formulated if a large number of problems of share a common solution procedure [8].

3.3. Efficiency Concerns

An important aspect of the derivational analogy approach is the ability to store and trace dependency links. It should be noted that some of the inherent inefficiencies that would be present if we were to enforce truth maintenence on very large dependency networks, such as would be required to span an entire knowledge base in maintaining a large deductive data base do not apply to this situation. Since the dependency links are internal to each derivation with external pointers only to the problem description and to any volatile assumptions necessitated in constructing the resultant solution, the dependency networks are quite small. Hence, the size of each dependency network is quite small, compared to a dependency network spanning all of memory. Dependencies are also stored among decisions taken at different stages in the temporal sequence of the derivation, thus providing the derivational analogy process access to causal relations computed at the time the initial problem was solved.

The analogical transformation process is not inherently space inefficient, although it may so appear at first glance. The sequence of decisions in the solution path of a problem are stored, together with necessary dependencies, the problem description, the resultant solution, and alternative reasoning paths not chosen. Failed paths are not stored, only the initial decision that was taken to embark upon that path, and the eventual reason for failure (with its causal dependencies), are remembered. Hence, the size of the memory for derivational traces is proportional to the depth of the search tree, rather than to the number of nodes visited. Problems that share large portions of their derivational structure can be so represented in memory, saving space and allowing similarity-based indexing. Moreover, when a generalized plan is formulated for recurring problems that share a common derivational structure, the individual derivations that are totally subsumed by the more general structure can be permanently masked or deleted. Those derivations that represent exceptions to the general rule, however, are precisely the instances that should be saved and indexed accordingly for future problem solving [14].

3.4. Concluding Remarks

Derivational analogy bears closer resemblance to Schank's reconstructive memory [27, 28] and Minsky's K-lines [21] than to rather more traditional notions of analogy. Although derivational analogy is less ambitious in scope than either of these theories, it is a more precisely defined inference process that can lead to an operational method of reasoning from particular experiential instances. The key notion is to reconstruct the relevant decision making processes of past problem solving situations and thereby transfer knowledge to the new scenario. The knowledge consists of decision sequences and their justifications, rather than individual declarative assertions. To summarize, let us describe the process of derivational analogy in terms of the criteria for analogical reasoning:

1. Two problems share significant aspects if their initial analysis yields the same reasoning steps, i.e., if the initial segments of their respective derivations start by considering the same issues and making the same decisions. An operational test for similarity is identity of the initial decision sequence.
2. Then the derivation of the retrieved solution may transferred to the new situation, in essence recreating the significant aspects of the reasoning process that solved the past problem.
3. Knowledge transfer is accomplished by reconsidering old decisions in light of the new problem situation, preserving those that apply, and replacing or modifying those whose justifications are no longer valid in the new situation.
4. Problems and their derivations are stored in a large episodic memory organized in a manner similar to Schank's MOPS [28], and retrieval occurs by replication of initial segments of decision sequences recalling the past reasoning process.

4. References

1. Balzer, R., "Imprecise Program Specification," Tech. report RR-75-36, USC/Information Sciences Institute, 1975.
2. Barstow, D. R., *Automatic Construction of Algorithms and Data Structures Using a Knowledge Base of Programming Rules*, PhD dissertation, Stanford University, Nov. 1977.
3. Carbonell, J. G., "Counterplanning: A Strategy-Based Model of Adversary Planning in Real-World Situations," *Artificial Intelligence*, Vol. 16, 1981, pp. 295-329.
4. Carbonell, J. G., "A Computational Model of Problem Solving by Analogy," *Proceedings of the Seventh International Joint Conference on Artificial Intelligence*, August 1981, pp. 147-152.
5. Carbonell, J. G., Larkin, J. H. and Reif, F., "Towards a General Scientific Reasoning Engine," Tech. report,

Carnegie-Mellon University, Computer Science Department, 1983, CIP #445.

6. Carbonell, J. G., *Subjective Understanding: Computer Models of Belief Systems,* Ann Arbor, MI: UMI research press, 1981.

7. Carbonell, J. G., "Experiential Learning in Analogical Problem Solving," *Proceedings of the Second Meeting of the American Association for Artificial Intelligence,* Pittsburgh, PA, 1982.

8. Carbonell, J. G., "Learning by Analogy: Formulating and Generalizing Plans from Past Experience," in *Machine Learning, An Artificial Intelligence Approach,* R. S. Michalski, J. G. Carbonell and T. M. Mitchell, eds., Tioga Press, Palo Alto, CA, 1983.

9. Clements, J., "Analogical Reasoning Patterns in Expert Problem Solving," *Proceedings of the Fourth Annual Conference of the Cognitive Science Society,* 1982.

10. Doyle, J., "A Truth Maintenance System," *Artificial Intelligence,* Vol. 12, 1979, pp. 231-272.

11. Duda. R. O., Hart, P. E., Konolige, K. and Reboh, R., "A Computer-Based Consultant for Mineral Exploration," Tech. report 6415, SRI, 1979.

12. Fikes, R. E. and Nilsson, N. J., "STRIPS: A New Approach to the Application of Theorem Proving to Problem Solving," *Artificial Intelligence,* Vol. 2, 1971, pp. 189-208.

13. Gentner, D., "The Structure of Analogical Models in Science," Tech. report 4451, Bolt Beranek and Newman, 1980.

14. Hayes-Roth, F., "Using Proofs and Refutations to Learn from Experience," in *Machine Learning, An Artificial Intelligence Approach,* R. S. Michalski, J. G. Carbonell and T. M. Mitchell, eds., Tioga Press, Palo Alto, CA, 1983.

15. Kant, E., *Efficiency in Program Synthesis,* UMI Research press, Ann Arbor, MI, 1981.

16. Kling, R. E., "A Paradigm for Reasoning by Analogy," *Artificial Intelligence,* Vol. 2, 1971, pp. 147-178.

17. Laird, J. E. and Newell, A., "A Universal Weak Method," *Proceedings of the Eight Joint Conference on Artificial Intelligence,* 1983, (submitted).

18. McDermott, D. V. and Doyle J., "Non-Monotonic Logic I," *Artificial Intelligence,* Vol. 13, 1980, pp. 41-72.

19. McDermott, D. V., "Planning and Acting," *Cognitive Science,* Vol. 2, No. 2, 1967, pp. 71-109.

20. McDermott, J., "XSEL: A Computer Salesperson's Assistant," in *Machine Intelligence 10,* Hayes, J., Michie, D. and Pao, Y-H., eds., Chichester UK: Ellis Horwood Ltd., 1982", pp. 325-337.

21. Minsky, M., "K-Lines: A Theory of Memory," *Cognitive Science,* Vol. 4, No. 2, 1980, pp. 117-133.

22. Newell, A. and Simon, H. A., *Human Problem Solving,* New Jersey: Prentice-Hall, 1972.

23. Nilsson, N. J., *Principles of Artificial Intelligence,* Tioga Press, Palo Alto, CA, 1980.

24. Polya, G., *How to Solve It,* Princeton NJ: Princeton U. Press, 1945.

25. Reif, J. H. and Scherlis, W. L., "Deriving Efficient Graph Algorithms," Tech. report, Carnegie-Mellon University, Computer Science Department, 1982.

26. Sacerdoti, E. D., "Planning in a Hierarchy of Abstraction Spaces," *Artificial Intelligence,* Vol. 5, No. 2, 1974, pp. 115-135.

27. Schank, R. C., "Language and Memory," *Cognitive Science,* Vol. 4, No. 3, 1980, pp. 243-284.

28. Schank, R. C., *Dynamic Memory,* Cambridge University Press, 1982.

29. Schank, R. C., "The Current State of AI: One Man's Opinion," *AI Magazine,* Vol. IV, No. 1, 1983, pp. 1-8.

30. Shortliffe, E., *Computer Based Medical Consultations: MYCIN,* New York: Elsevier, 1976.

31. Swartout. W. and Balzer, R., "An Inevitable Intertwining of Specification and Implementation," *Comm. ACM,* Vol. 25, No. 7, 1982.

32. Wilensky, R., *Understanding Goal-Based Stories,* PhD dissertation, Yale University, Sept. 1978.

33. Wilensky, R., *Planning and Understanding,* Addison Wesley, Reading, MA, 1983.

34. Winston, P., "Learning by Creating and Justifying Transfer Frames," Tech. report AIM-520, AI Laboratory, M.I.T., January 1978.

35. Winston, P. H., "Learning and Reasoning by Analogy," *Comm. ACM,* Vol. 23, No. 12, 1979, pp. 689-703.