CMU-CS-82-126

Learning by Analogy: Formulating and Generalizing Plans from Past Experience

Jaime G. Carbonell

19 June 1982

To appear in *Machine Learning, an Artificial Intelligence Approach*
R. S. Michalski, J. G. Carbonell and T. M. Mitchell (Eds.), Tioga Press,
Palo Alto, CA, 1982

This research was sponsored in part by the Office of Naval Research (ONR) under grant number N0014-79-C-0661, and in part by the Defense Advanced Research Projects Agency (DOD), ARPA order number 3597, monitored by the Air Force Avionics Laboratory under contract number F-33615-81-K-1539. The views and conclusions in this document are those of the author and should not be interpreted as representing the official policies, either expressed or implied, of the Defense Advanced Research Agency or the US Government.

Concluding Remark

Table of Contents

Learning by Analogy: Formulating and Generalizing Plans from Past Experience

Jaime G. Carbonell

Abstract

Analogical reasoning is a powerful mechanism for exploiting past experience in planning and problem solving. This paper outlines a theory of analogical problem solving based on an extension to means-ends analysis. An analogical transformation process is developed to extract knowledge from past successful problem solving situations that bear strong similarity to the current problem. Then, the investigation focuses on exploiting and extending the analogical reasoning model to generate useful exemplary solutions to related problems from which more general plans can be induced and refined. Starting with a general analogical inference engine, problem solving experience is, in essence, compiled incrementally into effective procedures that solve various classes of problems in an increasingly reliable and direct manner.

1. Introduction

Analogical reasoning has been a sparsely-investigated phenomenon in Artificial Intelligence [11, 20, 13, 31]. Nonetheless, analogy promises to be a central inference method in human cognition as well as a powerful computational mechanism. This paper discusses a computational model of problem solving by analogy based on an extension of means-ends analysis (MEA). My central hypothesis (based in part on Schank's theory of memory organization [28, 27]) is the following: When encountering a new problem situation, a person is reminded of past situations that bear strong similarity to the present problem (at different levels of abstraction). This type of reminding experience serves to retrieve behaviors that were appropriate in earlier problem solving episodes, whereupon past behavior is adapted to meet the demands of the current situation.

Commonalities among previous and current situations, as well as successful applications of modified plans can serve as the basis for generalization. Similarly, performing an inappropriate action in a new situation can provide information useful in reorganizing episodic memory. If the inappropriate action resulted from the application of a recently acquired general plan, an analysis of the type of error may trigger a discrimination process that constrains the range of applicability for that plan. In either case, a *reactive environment* that informs the problem solver of success, failure, or partial success is an absolute requirement for any generalization or discrimination process to apply.

Whereas humans exhibit a universal ability to learn from experience no matter what the task [22], AI systems are seldom designed to model this adaptive quality. Concept acquisition, i.e., inducing structural or attribute descriptions of non-procedural objects from examples, has received substantial attention in the AI literature [10, 8, 18, 29, 30], but with few exceptions, the techniques developed therein have not been transferred to learning in problem-solving scenarios.[2] Since the process of acquiring and refining problem solving and planning skills is indisputably a central component in human cognition, its investigation from an AI perspective is clearly justified.

This paper presents an analogical inference engine and investigates two fundamental hypotheses:

> **Hypothesis:** *Problem solving and learning are inalienable aspects of a unified cognitive mechanism.*

In other words, one cannot acquire the requisite cognitive skills without solving problems --- and, the very process of solving problems provides the information necessary to acquire and tune problem solving skills. The second hypothesis postulates a unified learning mechanism.

> **Hypothesis:** *The same learning mechanisms that account for concept formation in declarative domains, operate in acquiring problem-solving skills and formulating generalized plans.*

One method of verifying the second hypothesis is to develop a problem solving mechanism into which one can integrate the techniques developed in concept formation --- with a resultant system that learns from problem solving experience. The analogical problem solving method discussed below provides a framework for automated example generation that enables one to apply learning-from-examples techniques in order to acquire generalized plans. In essence, the objective is akin to Anzai and Simon's learning-by-doing method [2]. First, the basic analogical problem-solving method is discussed, and subsequently an experiential learning component is incorporated as an integral part of the general analogical inference process.

2. Problem Solving by Analogy

Traditional AI models of problem solving (e.g., GPS [21], STRIPS [9], and NOAH [24]) approach every problem almost without benefit of prior experience in solving other problems in the same or similar problem spaces.[3] Consider, for instance, two related problems:

[2] Exceptions include Anzai and Simon's Learning-by-Doing Paradigm [2], Mitchell's LEX system [19], STRIPS with MACROPS [9], and indirectly Lenat's AM [15].

[3] A *problem space* encodes the information necessary to solve a problem, including goals, initial state, and legal actions that may be taken in solution attempts. *Means-Ends Analysis* is a problem solving method that consists of selecting actions that reduce known differences between the current situation and a desired state. Both of these concepts are elaborated in the course of the present discussion. However, the reader not familiar with Means Ends Analysis is encouraged to review the technique in any standard AI text, such as Winston's *Artificial Intelligence* [32] or Nilsson's *Principles of Artificial Intelligence* [23], or read the much more thorough treatment in [21].

> **The monkey-and-bananas problem:** A (hungry) monkey is placed in a room with bananas suspended from the ceiling beyond its reach. A wooden box of sufficient size to serve as a platform from which the monkey can reach up to the bananas is placed elsewhere in the room.
>
> **The experimenter-and-bananas problem:** An experimenter wishes to set up the monkey-and-bananas problem. He has some bananas, a hook in the ceiling just beyond his reach, and a wooden box elsewhere in the experimental room, and, of course, a monkey.

A means-ends-analysis problem solver, such as GPS, will solve either problem, given sufficient time and a reasonable encoding of the permissible actions and their consequences. However, solving one problem does not provide any information useful in solving the other. One would think that practice solving a given type of problem should help in solving similar future problems. For instance, an intelligent monkey observing the experimenter move the box beneath the hook, hang the bananas, and return the box to its original location, may infer which parts of the experimenter's behavior it should replicate in order to reach the bananas. Similarly, if the experimenter tires of watching an unenlightened monkey repeatedly fail in its attempts to solve the problem, he should know how to take down the bananas by modifying parts of his earlier plan, rather than replanning from ground zero. In general, transfer of experience among related problems appears to be a theoretically significant phenomenon, as well as a practical necessity in acquiring task-dependent expertise necessary to solve more complex real-world problems. Indeed, the premise that humans transfer problem-solving expertise among closely related situations is inextricably woven into the pedagogical practices of our educational institutions.

The bulk of human problem solving takes place in problem spaces that are either well known or vary only slightly from familiar situations. It is rare for a person to encounter a problem that bears no relation to similar problems solved or observed in past experience. New abstract puzzles (such as Rubik's magic cube) are such exceptional problems, where initially the only tractable solution procedure is the application of standard weak methods [21] without benefit of (non-existent) past experience. Therefore, my investigations center on simplified versions of real-world problems, rather than more abstract, self-contained puzzles.

Now, let us turn to problem solving in familiar problem spaces. What makes a problem space "familiar"? Clearly, a major aspect consists of memory of past problems and their corresponding solutions that bear strong similarity to the new problem. Such knowledge, once acquired, can be exploited in the problem solving process. There is no other way to account for the fact that humans solve problems in familiar situations much faster, and with more self-assurance than in unfamiliar abstract situations. A computer model should exhibit the same skill-acquisition process; i.e., it should learn to adapt its problem-solving behavior by relying on past experience when available -- falling back on the application of standard weak methods when more direct recall-and-modification of existing solutions fails to provide an answer. How might a problem solver be augmented to exhibit such adaptive behavior? First, let us review the standard MEA process; then we see how the analogical transformation process augments MEA to exploit prior experience.

2.1. The Plan-Transformation Problem Space

Consider a traditional Means-Ends Analysis (MEA) problem space [21], consisting of:

- A set of possible problem states.
- One state designated as the *Initial State*
- One or more state(s) designated as *goal states* -- for simplicity, assume there is only one goal state.
- A set of operators with known preconditions that transform one state into another state in the space.
- A difference function that computes differences between two states (typically applied to compute the difference between the current state and the goal state).
- A method for indexing operators as a function of the difference(s) they reduce (e.g., the table of differences in GPS).
- A set of global path constraints that must be satisfied in order for a solution to be viable.[4] A path constraint is essentially a predicate on a partial solution sequence, rather than on a single state or operator. The introduction of path constraints in this manner constitutes a slight modification of the standard MEA problem space.

Problem solving in this space consists of standard MEA:

1. Compare the current state to the goal state
2. Choose an operator that reduces the difference
3. Apply the operator if possible -- if not, save the current state and apply MEA to the subproblem of establishing the unsatisfied precondition(s) of that operator.
4. When a subproblem is solved, restore the saved state and resume work on the original problem.

[4] For instance, a path constraint may disallow particular subsequences of operators, or prevent an operator that consumes K amount of a resource from applying more than N times, if there is only NxK amount of the resource available to the problem solver.

How can one exploit knowledge of solutions to previous problems in this type of problem space? First, consider the simplest case; knowledge consists only of solutions to previous problems. Each solution consists of a sequence of operators and intermediate states, including the initial and final states, together with the path constraints that the solution was designed to satisfy. One rather simple idea is to create "macro-operators" from sequences and sub-sequences of atomic operators that have proven useful as solutions to earlier problems. For instance, STRIPS with MACROPS exploited this idea [9] using its "triangle table" to store all partial sequences of operators encountered in a solution to a previous problem. However, the simple creation of macro-operators suffers three serious shortcomings. First, the combinatorics involved in storing and searching all possible subsequences of all solutions ever encountered becomes rapidly unmanageable. Searching for applicable macro-operators can become a more costly process than applying MEA to the original problem. Second, path constraints are ignored in this process. If the new problem must satisfy a different set of path constraints, most previous macro-operators may prove invalid. Third, no provision is made for substituting, deleting, or inserting additional operators into recalled solution sequences. These operations prove crucial in the analogical transform process described below. Therefore, let us think not in terms of creating more and more powerful operators that apply to fewer and fewer situations, but rather think in terms of gradually transforming an existing solution into one that satisfies the requirements of the new problem.

Consider a reminding process (a search for solutions to problems similar to the one at hand) that compares differences among the following:

1. The initial state of the new problem and the initial state of previously-solved problems

2. The final state of the new problem and the final state of previously-solved problems

3. The path constraints under which the new problem must be solved and path constraints present when previous similar problems were solved.

4. The proportion of operator preconditions of the retrieved operator sequence satisfied in the new problem situation. This measure is called the *applicability* of a candidate solution.

The difference function used in comparing initial and final states may be the very same function used for difference reduction in standard MEA. Here, I advocate using the difference function as a *similarity metric* to retrieve the solution of a previously-solved problem closely resembling the present problem. The difference function applied to path constraints is an augmented version of the problem-state difference function, as it must address operator-sequence differences in addition to state information. Hence, *reminding* in our problem-solving context consists of recalling a previously solved problem whose solution may transfer to the new problem under consideration. A more sophisticated method of computing similarities among episodic memory structures is based on a *relative-invariance hierarchy* among different components of recalled problem solutions, as discussed in [5].

Reminding is only the first phase in analogical problem solving. The second phase consists of transforming the old solution sequence into one satisfying the criteria for the new problem. How does this transformation process proceed? I submit that it is equivalent to problem solving in the space of solutions.[5]

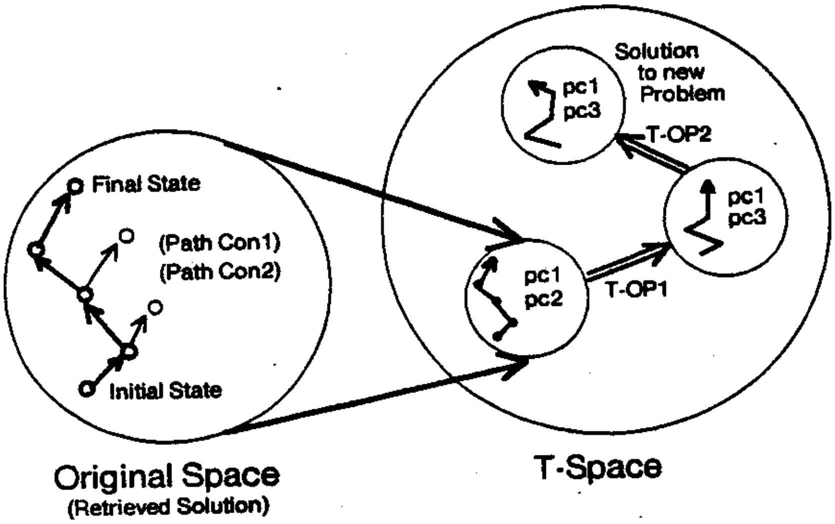

Figure 2-1: A solution path in the original problem space becomes a state in the *analogy transform problem space*.

Finding an appropriate analogical transformation is itself a problem solving process, but in a different problem space. The states of the transform problem space are solutions to problems in the original problem space. Thus, the initial state in the transform space is the retrieved solution to a similar problem, and the goal state is a solution satisfying the criteria for the new problem. The operators in the transform problem space are the atomic components of all solution transformations (e.g., substitute an operator in the solution sequence for another operator that reduces the same difference, but requires a different set of preconditions or entails different side effects, etc. -- see

[5] Here I apply my previous definition of a solution to be a sequence of operators and intermediate states together with the set of path constraints that sequence is known to satisfy. Thus, I advocate applying MEA to the space of potential solution sequences rather than the original problem space. However, the reminding process should generate an initial solution sequence close to the goal solution sequence, where closeness is determined by the difference metric above.

below). The differences that the problem solver attempts to reduce in the new problem space are precisely those computed by the similarity metric in the reminding process. In other words, progress towards a goal is determined by transitions in the solution space towards "solution sequences" corresponding to problems increasingly similar to the new problem. Intermediate states in the transform space need not correspond to *viable* solutions in the original (object) space, in that intermediate solution sequences may not be executable due to unsatisfied operator preconditions. The diagram in figure 2-1 gives an intuitive flavor of this problem-solving process. More precisely, the *analogy transform problem space* (T-space) is defined as follows:

- States in the transform space are potential solutions to problems in the original problem space (i.e., sequences of states and operators including the initial and final states, plus the path constraints under which those solutions were computed.)

- The initial state in the transform space is the solution to a similar problem retrieved by the reminding process.

- A goal state in the transform space is the specification of a solution that solves the new problem, satisfying its path constraints.

- An operator in the transform space (labeled a "T-operator" to avoid confusion) maps an entire solution sequence into another potential solution sequence. The following is a list of the most useful T-operators:

    - ***General Insertion.*** Insert a new operator into the solution sequence.

    - ***General deletion.*** Delete an operator from the solution sequence.

    - ***Subsequence Splicing.*** Splice a solution to a new subproblem into the larger established solution sequence. This T-operator is useful in the following situation: If an operator in the original problem sequence cannot be applied under the new problem specification because one of its preconditions is not satisfied, solve the subproblem of establishing that precondition. This subproblem may be solved either in T-space or in the original (object) space. If successful, splice the precondition-fulfilling subsequence into the original solution sequence.

    - ***Subgoal-preserving substitution.*** Substitute an operator in the original solution sequence by another operator (or sequence of operators) that reduces the same difference. This T-operator is particularly useful if either a precondition of an operator in the original sequence cannot be satisfied, or if the presence of a particular operator in the solution sequence violates a path constraint.[6]

    - ***Final-segment concatenation.*** Treat the solution sequence as a macro-operator

[6] Note that a subgoal-preserving substitution is much more restrictive than a general delete T-operator followed by a general insert T-operator. Therefore, this T-operator is more apt to yield useful transformations, a fact reflected in the ordering of operators under each appropriate entry in the difference table.

in the original problem space and apply MEA to reduce the difference between the old final state and the new final state. If successful, concatenate the solution to this subproblem at the end of the original solution sequence.

- ***Initial-segment concatenation.*** Apply the process above to find a path in the original problem space from the new initial state to the old initial state. If successful, concatenate the solution to this subproblem at the beginning of the original solution. [Note that in this case we *start* with the initial state for the new problem and seek a path to the initial state for the retrieved solution, whereas in the final segment-concatenation operator the inverse process applies.]

- ***Sequence meshing.*** Merge the operator sequences of two complementary solutions retrieved in the reminding process. The resultant solution sequence should differ from a complete solution to the new problem by the intersection of the differences between each retrieved solution and the new problem specification.[7] If the differences between the two retrieved solutions and the new problem specification form disjoint sets, sequence meshing yields a complete solution.

- ***Operator reordering.*** Reorder the operators in a solution sequence. Often a path constraint in the new problem specification can be satisfied by simple reordering of operators (when allowed by their preconditions) in the retrieved solution.

- ***Parameter substitution.*** Substitute the objects to which operators were applied in the retrieved solution by the corresponding objects in the new problem specification.

- ***Solution-sequence truncation.*** Eliminate unnecessary operators. Two significant special cases of this T-operator are ***initial-segment truncation*** and ***final-segment truncation.*** For instance, if the final state of an operator subsequence of the retrieved solution exhibits a smaller difference to a goal state of the new problem, use this subsequence as the new basis for mapping into the desired solution sequence.

- ***Sequence inversion.*** Reverse the operator sequence, inverting each individual operator, if a problem formulation is such that its goal state matches the initial state of a solved problem, and its initial state matches the goal state of that same previously solved problem. Inverting a process is not always possible, and seldom directly achievable. In the present case, the inverse of each operator must be found, and its preconditions satisfied, in order to apply global inversion. However, the general notion is attractive -- consider solving the problem of driving between two points in an unknown city. Once this problem is solved, the subsequent problem of returning to the departure site is easily solved by operator sequence inversion.

[7] Merging two partial operator sequences is an interesting and potentially complex problem in itself. Procedural networks, developed in the NOAH system [24], facilitate computations of operator interactions when meshing two plans. It is not always the case that two partial solution sequences can be merged effectively (e.g., each subsequence may violate necessary preconditions for the other subsequence). Non-algorithmic T-operators, such as ***sequence meshing,*** define their own internal problem space.

- The *difference metric* in the transform space (D_T) is a combination of the difference measures between initial states (of the retrieved and desired solution sequences), final states, path constraints, and degree of applicability of the retrieved solution in the new problem scenario. Hence, the values of D_T are 4-vectors, with the interpretation that all four component differences must be reduced (independently or jointly) in the transform space (T-space) problem-solving process.

  $$D_T = \langle D_O(S_{I,1},S_{I,2}), D_O(S_{F,1},S_{F,2}), \\ D_P(PC_1,PC_2), D_A(SOL_1,SOL_2)\rangle$$

  D_O is the difference function between states in the original space.

  D_P computes differences between path constraints (PC's).

  D_A measures the applicability of the old solution in the new scenario by determining the fraction of operators in the initial solution sequence (SOL_1) whose preconditions are not satisfied under the new problem specification.

  S_I denotes an initial state.

  S_F denotes a final (goal) state.

  The subscript 1 indexes the retrieved solution.

  The subscript 2 indexes the specifications on the desired solution to the new problem.

  D_T is reduced when any of its four components is independently reduced. The problem-solving process in T-space succeeds when D_T = <NIL, NIL, NIL, NIL>. Interesting search problems occur when, in order to reduce one component in the difference vector, one or more of the other components must be increased. For example, the insertion of new operators into the solution sequence may have the unfortunate side-effect of violating an established precondition of an operator in the original sequence. In this case reducing $D_O(I)$ or $D_O(F)$ results in increasing D_A. Our first-pass solution is to define a (linear) combination of the four components and choose the operator that maximally reduces this value, backtracking when necessary. Fortunately, it is often the case that differences in the 4-vector can be reduced in a componentwise-independent manner. Moreover, a modified version of the Δ-MIN method [4] may apply, focusing the backtracking process when backtracking proves necessary.

- A difference table for indexing the T-operators is needed. Entries in the difference table take the form `"To reduce <DIFFERENCE>, apply a member of <T-OPERATOR-SET>"`. The operators in the applicable set are usually ordered as a function of the heuristic measure of their utility in reducing the given difference. A sample difference table entry would be:

  - If the preconditions to an operator in SOL_1 are not satisfied (i.e., D_A is non-null), try ***subgoal-preserving substitution*** on the inapplicable operator, or try ***solution-sequence splicing*** to satisfy the violated preconditions.

- There are no path constraints in the transform space. Since we are mapping from one solution sequence to another, the intermediate states and T-operators do not necessarily correspond to actual operations performed on an external world, and therefore are not subject to its restrictions. This simplification is offset by the more complex difference metric in T-space.

2.2. Some examples

Consider a simple problem where analogical problem solving may prove quite appropriate:

> John is located in Pittsburgh and must travel to New York City. However, when he called the airlines, he discovered that all the flights were booked. John never took the intercity train (Amtrak) before, but knows it is a possible means of long-distance travel.

John's plan might be the following: Call Amtrak to make a reservation. Make sure he has sufficient money for the ticket. Find out where to buy the ticket; buy it; and later go to the station and board the train. Why is this a reasonable plan? How could John have synthesized his plan? We cannot really say that John had a "script"[8] for taking trains, as he had not previously traveled by train, nor had he acquired the requisite, detailed information enabling him to do so.

A reasonable way of formulating the plan is by analogy with taking an airplane (or perhaps an intercity bus). The first step is for John to be reminded of taking an airplane (thus recalling: making reservations, tickets being costly, often purchasing the tickets in advance, later traveling to the airport, etc.) Note that it is crucial for John to be reminded of an experience (or a general procedure) where he was fulfilling a similar goal (intercity travel) and ***not*** one where superficial similarities abound (e.g., taking a subway, where both means of conveyance are called "trains", they travel on tracks, have many stops, etc.). Subway travel would not suggest the potential necessity of making a reservation, nor would it suggest the requirement for a reasonable sum of money to purchase the ticket. Hence, a comparison of goal states, as suggested in our general method, is indeed a crucial component in the similarity judgements necessary for modeling a reasonable reminding process.

The solution transformation process proceeds by applying the ***subgoal-preserving substitution*** T-operator, substituting TRAIN-TRAVEL for AIR-TRAVEL, as both operators reduce the same difference. Then, the ***parameter-substitution*** T-operator replaces "airport" by "train station", "airline ticket" by "train ticket", etc. John must rely on his knowledge of how to satisfy the preconditions of AIR-TRAVEL, and hope that the same methods apply to TRAIN-TRAVEL. If this were not the case, further problem solving would be necessary.

[8] By "script" I mean a slight variation of Schank and Abelson's terminology [26, 7], i.e., a frozen plan: one or more normative sequences of planned actions whose purpose is to satisfy the preconditions of (and carry out) a high-level operator.

Now, let us reconsider the monkey-and-bananas and experimenter-and-bananas problems, in light of the analogical problem-solving model.

> A monkey watches a behavioral psychologist (i.e., the experimenter) pick up a wooden box and place it under a hook in the ceiling. Next, the experimenter climbs on the box, places some bananas on the hook, climbs off the box, and returns it to its original location. Then, the experimenter releases the (hungry) monkey and leaves the room. How does the monkey plan to reach the bananas? Can he benefit from having observed the experimenter?

As we mentioned earlier, a "smart monkey" ought to learn from his observations of the experimenter. Let us see how analogical problem solving applies here. For simplicity, assume the monkey does not have prior experience solving similar problems beyond his recent observation of the experimenter. The monkey's problem is: **initial state** = monkey on the floor, bananas on the ceiling, box in the room; **final state** = monkey in possession of the bananas; **path constraints** = physical abilities of the monkey. However, the solution to the experimenter's problem cannot be applied directly. (His problem was **initial state** = possession of the bananas, box in the room, experimenter on the floor; **final state** = Bananas on the ceiling, box *not* under the bananas; **path constraints** = physical abilities of the experimenter.)

Assuming the path constraints match, the differences between the initial states (and the differences between the final states) are so large as to preclude any reasonable attempt at direct analogical transformation. Therefore, the monkey must resort to standard MEA (in the original problem space). He selects the operator GET-OBJECT (applied to bananas). This operator suffers an unsatisfied precondition: The monkey cannot reach the bananas. Therefore, the active subgoal becomes: Reach the ceiling where the bananas are located. How may the monkey proceed at this juncture?

The entire problem can, of course, be solved by recursively applying standard MEA. However, there is a more direct solution method. If the monkey recalls his observation of the experimenter, he may realize that the problem of reaching the ceiling has already been solved (by the experimenter, as a subgoal to placing the bananas there -- although the monkey need not understand the experimenter's higher-level goals). The monkey can apply the ***parameter-substitution*** T-operator (substituting "monkey" for "experimenter"), and optionally the ***solution-sequence truncation*** T-operator (eliminating the need to return the box to its original location after having used it). This problem-solving process in T-space results in a plan that the monkey can apply directly to reach the bananas, and thus achieve his original goal of having the bananas.

The significant aspect of the experimenter-monkey-and-bananas example is that standard MEA and T-space MEA were combined into a uniform problem-solving process where standard MEA calls on analogical problem solving to solve a subproblem more directly. The converse process is also possible, and potentially significant. For instance, in the Amtrak example, if John could not have satisfied one of the preconditions for taking the train by analogy with the corresponding AIR-TRAVEL precondition, he could have resorted to standard MEA to solve this subproblem. Hence, *Analogical reasoning adds a powerful dimension to standard problem solving when prior experience can be brought to bear, but remains largely unobstrusive when no relevant prior knowledge suggests itself.*

It would be useful for the problem solver to remember his new problem-solving experiences to use as a basis for future analogical reasoning. These could be remembered directly or abstracted into episodic traces, much like Schank and Abelson's scripts [26, 7], and hierarchically organized as a function of the goals they fulfill.

An interesting observation concerns the recursive closure of analogical MEA.[9] If the t-operator sequence of an analogical problem solving transformation is remembered, the analogical MEA process can be applied to these transformations themselves. That is, one can construct an analogical mapping between two solution sequences by recyling a past analogical mapping among similar solutions -- or by transforming a past, almost useable mapping by recursive application of analogical MEA to the analogical mapping itself. A significant point is that no infinite regress requiring new "hyper-analogical" methods occurs. The same analogical transformation process that applies to object-level solution sequences applies directly to transforming analogical mappings.

3. Evaluating the Analogical Reasoning Process

In an informal experiment, not meant to withstand statistical significance tests, I gave the following problem to five undergraduate history-and-art students:

Prove that the product of two even numbers is even.

Somewhat to my surprise and dismay, none of the five was able to solve this simple algebraic problem, although all five made serious attempts. I had intended to give the subjects similar but more difficult problems in subsequent stages of the experiment, measuring whether they improved in speed or accuracy from their recently-acquired experience solving analogically-related problems. Nevertheless, the experiment proved useful in demonstrating the reliance of human problems solvers on analogical mechanisms, as discussed below. Continuing with the experiment, I explained the proof process carefully enough to insure that all five subjects understood it:

[9] This observation is due in part to Mitchell, personal communication.

> First, recall the definition of an even number: a number that is divisible by 2.
>
> Second, write down an expression that represents an even number: You may write "2N" where N is any integer, to represent a number divisible by 2.
>
> Next, multiply two even numbers, writing: 2N x 2M, where M is also any integer. Multiplying we get 4NM.
>
> Now, recall the representation of an even number: 2 x any integer. Therefore you can write 4NM = 2 x 2NM, which by closure of integers under multiplication matches the representation of an even number. Hence, the product of two even numbers is even.

At this point, all five subjects claimed they understood the proof, and moreover expressed some feeling of embarrassment for not having derived such an "obvious" proof themselves. Then, I suggested they try the following problem:

> *Prove that the product of two odd numbers is odd.*

With grim determination to redeem their previous poor performance all five attempted the problem and three of them succeeded. Briefly:

> Odd numbers can be represented as "even + 1" = 2N + 1 for any integer N.
>
> The product is: (2N + 1) x (2M + 1) = 4NM + 2N + 2M + 1 = 2(2NM + N + M) + 1, which is the representation of an odd number.[10]

This informal experiment strongly indicates that the second problem was solved by analogy from the solution to the first problem. The scratch papers collected from the subjects suggest direct attempts at transferring and modifying steps of the first solution. The insertion of an extra algebraic step[11] illustrates an application of the ***subsequence splicing*** T-operator. The global substitution of a representation for odd numbers in place of a representation for even numbers strongly suggests ***parameter substitution***. Moreover, the mere fact that three of five subjects were able to solve a problem more complex than the one where all five failed previously, argues very convincingly for an analogical process exploiting the previous solution (or some abstraction thereof). However, it should be noted that this type of experiment does not in itself demonstrate dominance of analogical reasoning in human problem solving, but rather it provides strong evidence for the existence of

[10] Interestingly, one subject chose to represent odd numbers as 2N + 3, which is correct but requires a bit of additional algebraic manipulation. When asked why she chose such a representation, her reply was "4 is a nice even number, and 7 is a nice odd number. The difference between them is 3. The next even number is 6; the next odd is 9; and the difference is *always* 3. So, I took 2N and added 3." What a graphic illustration of means-ends-analysis to solve the subproblem of mapping from a representation of even numbers to a representation of odd numbers! Of the two subjects who did not present an adequate proof, one erred in an algebraic manipulation step, the other erroneously chose 3N as his representation for odd numbers.

[11] I.e., distributing the product of the two odd numbers is required to fulfill a precondition for factoring the constant "2" from three of the four terms in: 4NM + 2N + 2M + 1.

analogical processes in cognitive activities. Demonstrating the conjecture that analogy is the central inference mechanism for human problem solving would require a much more thorough (and perhaps more controlled) set of psychological observations.

As a test of the computational feasibility of the analogical problem solving process, a simple version of MEA was programmed to operate on the transform space, and given a subset of the T-operators with a corresponding difference table. It solved the product-of-two-odds problem starting from the solution for two even numbers.[12] The initial computer implementation of analogical MEA is not of particular interest -- it demonstrates that the analogical problem solving process actually works, but does little else. The truly interesting issues will arise when:

- a much fuller implementation is available allowing comparisons among different problem solving methods over a representative corpus of problems,
- the learning from experience process discussed in the following section is fully integrated with the analogical transform process,
- and the analogical problem solver is integrated with a dynamically-changing long term memory model.

4. Learning Generalized Plans

The analogical transformation process provides a method of exploiting prior experience in a flexible manner. That is, it requires only that the new problem be structurally similar, rather than identical, to one or more previously solved problems.[13] Hence, simply storing solutions to new problems constitutes a form of learning --- as these can serve as a basis from which solutions to yet newer problems may be analogized. However, there are other aspects to learning that present more interesting challenges. To wit, if a type of problem recurs with sufficient frequency, a human planner is apt to formulate a generalized plan for dealing with future instances of that problem, rather than reasoning analogically from a particular member of that cluster of similar experiences. A generalized plan is, in essence, similar to Schank's notion of a script [26, 28cull77], i.e., a parameterized

[12] The program used 2N-1 to represent an odd number, since the SUB1 operator was inadvertently listed before ADD1 in the object-space difference table, and therefore the program had to splice in an additional algebraic step in the solution: (2N-1)(2M-1) = 2(2NM - N - M) + 1, which does not correspond to the 2N-1 representation for odd numbers, and therefore had to apply ***subsequence splicing*** to add two algebraic operators that transformed the expression into 2(2NM - N - M + 1) - 1. In fact, most of the computational effort was spent finding those two operators (adding and subtracting the same quantity, and refactoring the expression). This allocation of effort roughly corresponds to the substantial time spent by the subject who chose 2N + 3 as a representation with the resultant product being 2(2NM + 3N + 3M) + 9, which did not exactly match the original representation, and was eventually refactored into 2(2NM + 3N + 3M + 3) + 3.

[13] The MACROPS facility in STRIPS required corresponding initial states and goal states to be identical modulo parameterization of operators in order to reuse portions of past solution sequences [9].

branching sequence of events with expected goals and default actions.

4.1. Acquiring Generalized Solutions Procedures

How is a generalized plan acquired from past problem solving experience? Consider an inductive engine, such as those developed to formulate generalized concepts from sequences of positive and negative exemplars of the target concept, as discussed in [10, 29, 30, 8, 18]. Instead of acquiring disembodied concepts from an external teacher providing training sequences of exemplars labeled "positive" or "negative", in experiential learning the exemplars consist of past problems and their respective solutions. These solutions are grouped together as exemplars of a generalized plan by virtue of being derived from a common ancestor in the analogical transform process. Thus, as in learning from observation, the concepts to be acquired are not known *a priori* by an external teacher, but correspond to clusters of experientially related solutions to a common type of problem. The "type" is not artificially defined, but depends on the actual experience of the individual problem solver. More specifically, generalized plans are acquired by the following process:

- Whenever the analogical problem solver generates a solution to a new problem, that solution is tested in the external world. If it works, it becomes a member of the positive exemplar set, together with the prior solution from which it was analogized and other successful solutions to problems from the same analogical root.

- If the analogized solution fails to work when applied in the external world, the cause of the failure is stored and this solution becomes a member of the corresponding negative exemplar set.

- The positive and negative exemplar sets are given to an induction engine that generates a plan encompassing all the positive solutions and none of the negative exemplars. Thus, the training sequence is provided by past experience solving similar problems, rather than by an external teacher. And, the concept acquired is a generalized solution procedure rather than the description of a static object, as is typically the case in the concept acquisition literature. If the description language for the object-space operators is extended, additional generalization can occur (e.g., in selecting more general operators that cover disjunctive subsequences in the generalized solution plan).

- Moreover, negative exemplars are near-misses,[14] since the analogical process generated them by making a small number of changes to known positive instances (i.e., transformations to past solutions of the same general problem type, retaining the bulk of the solution structure invariant). Hence, near-miss analysis can point out the relevant discriminant features between positive and negative exemplars of the general planning structure under construction. In other words, the problem solver serves as an automated

[14] Winston [30] defines a *near-miss* as a negative exemplar that differs from positive exemplars in a small number of significant features. Near misses are crucial in isolating defining characteristics of a concept in the learning-from-examples paradigm.

example generator, producing near-misses as a side effect when failing to generate an effective plan.

- Finally, in cases where the analogical problem solver fails to generate a solution for the new problem (as opposed to generating an erroneous solution that becomes a negative exemplar for the generalized plan formation process), different information can be acquired. The situations where a solution was recalled and a plan was formed analogically (independent of whether the plan worked) serve as positive exemplars to reinforce and perhaps generalize the similarity metric used to search memory. The cases where a recalled solution could not be analogized into a candidate plan for the new problem suggest that the old and new problems differed in some crucial aspect not adequately taken into account in the similarity metric, and thus serve as negative reinforcement to refine and constrain the similarity criterion.

Graphically, the information flow in the learning process is illustrated in figure 4-1. The formula

Analogy: $S_i/C_i \dashrightarrow P_j/C_j$

should be interpreted as "The analogical transform process maps plan P_i applicable under conditions C_i into plan P_j applicable under conditions C_j." And, the formula

Environment: $P_j/C_j \dashrightarrow +$ (or -)

should read as "Plan P_j succeeded (or failed) when executed in the external environment under conditions C_j."

Figure 4-1 summarizes the process of acquiring generalized plans and updating the similarity criterion from experience. The analogized plans along with their conditions of applicability, form the input to a learning-from-examples engine. Successful solutions are classified as positive exemplars; unsuccessful ones are classified as near-miss negative exemplars. Moreover, the cases where the analogy transform process failed to yield a candidate plan become negative reinforcement instances to a parameter-tuning process, which is positively reinforced by those cases where a (successful or unsuccessful) plan was formulated. Updating the similarity criterion should make future memory searches for solutions to similar problems more responsive to the features that enable the analogical transform system to map a recalled solution into a potential solution for the new problem. Thus, we see that analogical problem solving interfaces naturally with a learning-from-examples method in that it provides an internal example generator requiring no external teacher.

Presently, I am extending the problem solving engine to extract and use information from the planning process itself (not just problem descriptions and corresponding solutions), such as viable alternatives not chosen, causes of failure to be wary of in similar situations, etc. The objective of this endeavor is to enable the learning-from-examples component to learn, or at least refine, the problem solving strategies themselves, in addition to forming generalized plans. Thus, general patterns of

The analogical problem solving process

Analogy: S_1/C_1 --> P_2/C_2	Environment: P_2/C_2 --> +
Analogy: S_2/C_2 --> P_3/C_3	Environment: P_3/C_3 --> +
Analogy: S_1/C_1 --> P_4/C_4	Environment: P_4/C_4 --> -
Analogy: S_3/C_3 --> P_5/C_5	Environment: P_5/C_5 --> -

Analogy: S_3/C_3 --> <no-plan>/C_6
Analogy: S_1/C_1 --> <no-plan>/C_7

Acquiring generalized plans from solutions attempts to similar problems

Input to a learning-from-examples process
Positive exemplars: P_1/C_1, P_2/C_2, P_3/C_3
Negative exemplars: P_4/C_4, P_5/C_5 (near misses)

Output from the learning-from-examples process
Generalized plan: P_G/C_G

Updating the similarity criterion used to recall relevant prior experience

Input to a parameter-tuning process
Present similarity metric
Positive reinforcement trials: C_1, C_2, C_3, C_4, C_5
Negative reinforcement trials: C_6, C_7

Output from the parameter-tuning process
Updated similarity metric

Figure 4-1: Acquiring generalized plans and updating the similarity metric

inference may be acquired from experience [6].

Parts of the plan generalization process are currently being implemented to test the viability of the proposed knowledge acquisition method, and preliminary results are encouraging. Although, much of the theoretical and experimental work in acquiring problem solving skills is still ahead of us, there is sufficient evidence to support the two original hypotheses: the integration of learning and problem solving methods into a unified cognitive mechanism, and the utility of the learning-from-examples technique for acquiring planning skills as well as more static concepts.

As our discussion has demonstrated, learning can occur in both phases of analogical problem solving: 1) the reminding process that organizes and searches past experience, and 2) the analogical transformation process itself. Additional issues in the experiential adaptation of the reminding process are discussed below.[15]

4.2. Episodic Memory Organization

Memory of solutions to previous problems, whether observed or directly experienced, must be organized by similarities in goal states, initial states, and means available (or path constraints present). Otherwise, there can be no reasonable reminding process when solving future problems of a similar nature. Hence, a hierarchical indexing structure on an episodic memory must be dynamically constructed and extended as the system gradually accumulates new experience. Given an effective memory model, the process of continuously expanding and structuring past experience becomes a relatively simple, but absolutely essential, aspect of learning that proceeds concurrent with analogical reasoning. Moreover, the memory model should retrieve general plans when these have been proven reliable to the exclusion of the original episodic memory traces, which then effectively "fade" from memory. "Fading" means that the memory indexing structure is altered so they are no longer easily recalled in the reminding process. (This notion is akin to Schank's "mushing" process [27] and Anderson's masking by declining relative activation [1].)

4.3. Episodic Memory Restructuring

It is conceivable that in the lifetime of an adaptive problem solver, the nature of the problems it is called upon to solve may change gradually. The change may manifest itself as decreased reliability of the difference function comparing new and old problem specifications, causing the reminding process to retrieve inappropriate solutions, or to miss relevant past experiences. Hence, a means of tuning the difference metric in a failure-driven manner is a requisite process for long-term adaptive behavior.

More specifically, the heuristic combining the four values in the D_T 4-vector may be tuned to yield appropriate values for certain classes of problems most commonly encountered. For instance, differences in path constraints are less meaningful to a problem-solver who has ample resources than to a more spartanly-endowed problem solver. If a graduate student later becomes a millionaire, the fact that he now commands more substantial resources should lessen the impact of resource-based path constraints in his problem solving. Consequently, the similarity metric will cease to consider past

[15] The reader is referred to Schank [27, 28], Lebowitz [14] and Kolodner [12] for various discussions on the type of basic episodic memory model implicit in this paper. The memory organization scheme must be structured according to similarity criteria instrumental to the task indexing and recalling past problem solving experience [5].

solutions of otherwise similar problems that were solved when operating under more severe resource constraints. This is not a particularly desirable state of affairs, as resource-limited solutions are certainly viable, if not always desirable, to a problem solver commanding more resources. Therefore, the reminding heuristic should no longer weigh path-constraint differences as heavily. (Note that reminding is a constrained search process, whereas analogical mapping or instantiating a general solution pattern are generative processes. Hence, the reminding process need only retrieve approximate, *plausible* solutions.) Returning to our example, if that same millionaire later files for bankruptcy, the relevance of resource-based path constraints assumes significant proportions once again. A pauper will not be able to solve most problems by emulating a millionaire. Thus, the path-constraint component of the similarity/difference metric should reestablish its central role in the reminding heuristic. In this manner, the relevance of each component in the similarity measure is subject to long-term fluctuation.[16]

How can the relative weights in the similarity heuristic be tuned? When the reminding process fails to retrieve a viable initial state to the T-space problem solver, but the problem is later solved in the original problem space, the solution can be compared to episodic memory. If a solution to a previous problem is found to be very similar, then the problem descriptions should also have been found similar by the reminding heuristic. The component contributing the largest difference is then reduced in importance. The converse process also applies. If a solution retrieved as similar does not lead to a solution in T-space, the difference(s) that could not be reduced by the T-space problem solver are made more important in the difference heuristic. These complementary processes regulating the difference metric are designed to make all changes very gradually to insure against potentially unstable behavior. This form of experiential parameter tuning is a new application of a technique dating back to Samuel [25].

4.4. T-Operator Refinement

If episodic memory is extended to contain T-space problem-solving traces, in addition to experienced events and solutions to past problems, then learning can occur in the T-operator domain. For instance, consider a T-operator present with high frequency in unsuccessful T-space solution attempts. It is conceivable that the entry (or entries) in the difference table indexing that T-operator are insufficiently constrained, suggesting the need for a discrimination process such as

[16] This process is analogous to Berliner's application coefficients in SNAC [3], whose values change gradually over the course of a game. Here change occurs more gradually over the lifetime of the problem solver, but I am proposing an adaptive rather than a pre-programmed contextual-weighting process. Note that whereas individual path constraints differ from problem to problem, I am discussing gradual changes in the relative significance of path constraints *vis a vis* other criteria in the similarity metric on average over many individual problem solving episodes.

the following:

1. Compare T-space solution attempts where the T-operator in question was present only in failure paths, with solution attempts where it was present in successful solution paths.

2. If there are multiple entries in the difference table for that T-operator, and some entries correspond only to failure instances of the operator, delete these entries, as the operator is being applied to reduce a difference it proved incapable of reducing.

3. If a single entry corresponds to many more failures than successes, the description of the difference being reduced may be too general and ought to be factored into a disjunctive set of more specific differences. Later experience can help isolate which of these sub-differences the T-operator is actually capable of reducing. Then, the more specific differences (those that the T-operator in question proved capable of reducing) replace the previous more general entry in the difference table. Other differences in the factored disjunctive set that (as experience shows) cannot be reduced by the T-operator are discarded. It should be noted that the operation of factoring an arbitrary concept into a disjunctive set of sub-concepts is, in general, not a tractable process. However, given a hierarchical memory model and a non-monotonic inference capability,[17] approximately correct factorings can be achieved.

4.5. The Acquisition of New T-Operators

If the reminding process retrieved one or more solutions, but the analogy transform process failed to map these into a solution satisfying the specifications of the new problem, and the original-problem-space problem solver found a solution, then we have a clear indicator that the T-space problem solver is missing some essential T-operators. One approach to remedy this situation is the following process:

1. Compare the solution computed by the problem solver in the untransformed space with the various attempted transformations in T-space.

2. Find the intermediate state in the failed T-space solution attempt that minimizes the difference metric (D_T) to the solution computed by standard MEA.

3. Hypothesize a T-operator instance to be the transformation from the closest state (reached in the T-space solution attempts) to the actual solution. Save this T-operator instance.

4. If later problem-solving impasses cause failure-driven creation of more T-operator instances, then the application of a learning-from-observations technique, such as the *conceptual clustering* method [17] may prove fruitful. If the exemplars are sufficiently similar, or form clusters of closely similar exemplars, new T-operators can be hypothesized according to the characteristic description of each conceptual cluster.

[17] Non-monotonic inference is a plausible inference technique based on tentative deductions and assumptions that may prove invalid as additional knowledge is acquired [16].

"Sufficiently similar" in this context means that the common structure shared by the cluster of T-operator instances is not present in other active T-operators. Hence, the new operator will perform transformations different from those of any previously existing T-operator -- i.e., the new operator may prove generatively useful.

5. The newly-created T-operator may then be added to the set of active T-operators (subject to the refinement process above if the new operator proves unreliable).

6. The entry in the difference table indexing the new T-operator is a bounded generalization of the differences that each T-operator instance reduced at the time it was created. If these differences do not share a common component not present in other entries, more than one (disjunctive) entry must be made in the difference table.

Thus, new T-operators can be acquired if the problem solver is given a set of problems for which the same (previously unknown), general T-space transformation was required. Moreover, the operator acquisition and discrimination processes are equally applicable to refining and extending sets of operators in the original untransformed problem space (if the problem solver can tap an external source of knowledge upon failure, or relax processing constraints upon resource-limited failure). Acquiring T-operators, however, requires learning from observation, rather than the better understood and generally simpler process of learning-from-examples used to acquire generalized plans.

The learning mechanisms discussed in this section can prove effective if and only if the reasoning system is capable of remembering, indexing and retrieving past experience, including aspects of its internal processing in previous problem-solving attempts (e.g., hypothesized T-operator instances). Therefore, the necessity for **both** dynamic memory organization processes and a problem solving mechanism capable of exploiting episodic memory is clearly manifest.

5. Concluding Remark

The primary objective of this paper has been to lay a uniform framework for analogical problem solving capable of integrating skill refinement and plan acquisition processes. Most work in machine learning has not addressed the issue of integrating learning and problem solving into a unified process. (However, Mitchell [19] and Lenat [15] are partial counter-examples.) Past and present investigations of analogical reasoning have focused on disjoint aspects of the problem. For instance Winston [31], investigated analogy as a powerful mechanism for classifying and structuring episodic descriptions. Kling [11] studied analogy as a means of reducing the set of axioms and formulae that a theorem prover must consider when deriving new proofs to theorems similar to those encountered previously. In his own words, his system "...derives the analogical relationship between two [given] problems and outputs the *kind of information* that can be usefully employed by a problem-solving system to expedite its search." However, analogy takes no direct part in the problem-solving process itself. Hence, the extension of means-ends analysis to an analogy transform space is, in itself, a new, potentially-significant problem-solving method, in addition to supporting various learning mechanisms in an integrated manner.

References

1. Anderson, J. R. and Greeno, J. G., "Acquisition of Problem-Solving Skill," in *Cognitive Skills and Their Acquisition*, J. R. Anderson, ed., Hillsdale, NJ: Erlbaum Assoc., 1981.

2. Anzai, Y. and Simon, H. A., "The Theory of Learning by Doing," *Psychological Review*, Vol. 86, 1979 , pp. 124-140.

3. Berliner, H., "On the Construction of Evaluation Functions for Large Domains," *Proceedings of the Sixth International Joint Conference on Artificial Intelligence,* 1979 , pp. 53-55.

4. Carbonell, J. G., "Δ-MIN: A Search-Control Method for Information-Gathering Problems," *Proceedings of the First AAAI Conference,* August 1980 .

5. Carbonell, J. G., "Metaphor: An Inescapable Phenomenon in Natural Language Comprehension," in *Knowledge Representation for Language Processing Systems*, W. Lehnert and M. Ringle, eds., New Jersey: Erlbaum, 1982.

6. Carbonell, J. G., "Towards a Computational Model of Metaphor in Common Sense Reasoning," *Proceedings of the Fourth Annual Meeting of the Cognitive Science Society,* 1982 , Ann Arbor, MI.

7. Cullingford, R., *Script Application: Computer Understanding of Newspaper Stories,* PhD dissertation, Yale University, Sept. 1977.

8. Dietterich, T. and Michalski, R., "Inductive Learning of Structural- Descriptions," *Artificial Intelligence,* Vol. 16, 1981 .

9. 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.

10. Hayes-Roth, F. and McDermott, J., "Knowledge Acquisition from Structural Descriptions," *Proceedings of the Fifth International Joint Conference on Artificial Intelligence,* 1977 , pp. 356-362.

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

12. Kolodner, J. L., *Retrieval and Organizational Strategies in Conceptual Memory: A Computer Model,* PhD dissertation, Yale University, Nov. 1980.

13. Korf, R. E., "Toward a Model of Representation Changes," *Artificial Intelligence,* Vol. 14, No. 1, 1980 , pp. 41-78.

14. Lebowitz, M., *Generalization and Memory in an Integrated Understanding System,* PhD dissertation, Yale University, Oct. 1980.

15. Lenat, D., *AM: Discovery in Mathematics as Heuristic Search,* PhD dissertation, Stanford University, 1977.

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

17. Michalski, R. S., and Stepp, R. E., "Learning from Observation: Conceptual Clustering," in *Machine Learning, An Artificial Intelligence Approach,* R. S. Michalski, J. G. Carbonell and T. M. Mitchell, eds., Tioga Press, Palo Alto, CA, 1982.

18. Mitchell, T. M., *Version Spaces: An Approach to Concept Learning,* PhD dissertation, Stanford University, December 1978.

19. Mitchell, T. M., Utgoff, P. E. and Banerji, R. B., "Learning by Experimentation: Acquiring and Refining Problem-Solving Heuristics," in *Machine Learning, An Artificial Intelligence Approach,* R. S. Michalski, J. G. Carbonell and T. M. Mitchell, eds., Tioga Press, Palo Alto, CA, 1982.

20. Moore, J. and Newell, A., "How can MERLIN Understand?," in *Knowledge and Cognition,* L. Gregg, ed., Hillsdale, NJ: Erlbaum Assoc., 1974, pp. 253-285.

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

22. Newell, A. and Rosenbloom, P., "Mechanisms of Skill Acquisition and the Law of Practice," in *Cognitive Skills and Their Acquisition,* J. R. Anderson, ed., Hillsdale, NJ: Erlbaum Assoc., 1981.

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

24. Sacerdoti, E. D., *A Structure for Plans and Behavior,* Amsterdam: North-Holland, 1977.

25. Samuel, A. L., "Some Studies in Machine Learning Using the Game of Checkers," in *Computers and Thought,* E. A. Feigenbaum and J. Feldman, eds., McGraw Hill, New York, 1963, pp. 71-105.

26. Schank, R. C. and Abelson, R. P., *Scripts, Goals, Plans and Understanding,* Hillside, NJ: Lawrence Erlbaum, 1977.

27. Schank, R. C., "Reminding and Memory Organization: An Introduction to MOPS," Tech. report 170, Yale University Comp. Sci. Dept., 1979.

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

29. Vere, "Inductive Learning of Relational Productions," in *Pattern-Directed Inference Systems,* Waterman and Hayes-Roth, 1978, eds., New York: Academic Press, 1978.

30. Winston, P., *Learning Structural Descriptions from Examples,* PhD dissertation, MIT, September 1970.

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

32. Winston, P., *Artificial Intelligence,* Reading, MA: Addison Wesley, 1977.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER
CMU-CS-82-126

2. GOVT ACCESSION NO.
ONR

3. RECIPIENT'S CATALOG NUMBER

4. TITLE (and Subtitle)
LEARNING BY ANALOGY: FORMULATING AND GENERALIZING PLANS FROM PAST EXPERIENCE

5. TYPE OF REPORT & PERIOD COVERED
Interim

6. PERFORMING ORG. REPORT NUMBER

7. AUTHOR(s)
JAIME G. CARBONELL

8. CONTRACT OR GRANT NUMBER(s)
N0014-79-C-0661

9. PERFORMING ORGANIZATION NAME AND ADDRESS
Carnegie-Mellon University
Computer Science Department
Pittsburgh, PA. 15213

10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS

11. CONTROLLING OFFICE NAME AND ADDRESS

12. REPORT DATE
June 12, 1982

13. NUMBER OF PAGES
27

14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)

15. SECURITY CLASS. (of this report)
UNCLASSIFIED

15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

Approved for public release; distribution unlimited

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

Analogical reasoning is a powerful mechanism for exploiting past experience in planning and problem solving. This paper outlines a theory of analogical problem solving based on an extension to means-ends analysis. An analogical transformation process is developed to extract knowledge from past successful problem solving situations that bear strong similarity to the current problem. Then, the investigation focuses on exploiting and extending the analogical reasoning model to generate useful exemplary solutions to related problems from which more general plans can be induced and

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

refined. Starting with a general analogical inference engine, problem solving experience is, in essence, compiled incrementally into effective procedures that solve various classes of problems in an increasingly reliable and direct manner.