What Should I Do Now?
Using Goal Sequitur Knowledge to Choose the Next Problem Solving Step [1]

Michael Redmond
School of Information and Computer Science
Georgia Institute of Technology
Atlanta, Georgia 30332-0280
E-mail: redmond@pravda.gatech.edu

Abstract

Many problems require multi-step solutions. This is true of both planning and diagnosis. How can a problem solver best generate an ordered sequence of actions to resolve a problem? In many domains, complete pre-planning is not an option because the results of steps can vary, thus a large tree of possible sequences would have to be generated. We propose a method that integrates the use of previous plans or cases with use of knowledge of relationships between goals, and the use of reasoning using domain knowledge to incrementally suggest the actions to take. The suggestion process is constrained by heuristics that specify the circumstances under which an instance of a particular reasoning goal can follow from an instance of other reasoning goals. We discuss the general approach, then present the suggestion methods and the constraints.

1 Introduction

There are many problems for which a multi-step solution must be generated. Such problems occur both in planning and in diagnosis. For instance, in automobile troubleshooting, a possible sequence of actions includes clarifying the complaint, verifying the complaint, generating hypotheses in some order, testing hypotheses, interpreting the test results, carrying out repairs, and testing the repairs. Complete pre-planning of troubleshooting steps may be inefficient. The number of possible choices and the variety of possible results of the actions can lead to a large, very bushy tree of possible paths. Generation of the complete trouble-tree for the given car and problem would be an expensive task to do, and might not even be possible. In addition, one action may not directly follow from the previous action. The question raised is how to best generate an ordered sequence of actions to resolve a problem.

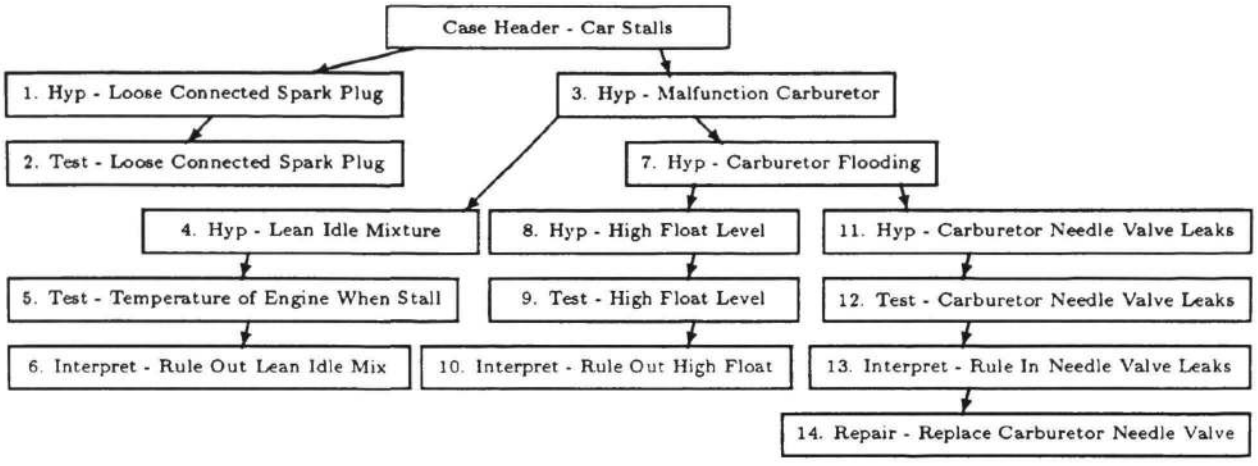
For example, for a stalling car, a possible sequence of actions is shown in Figure 1. The problem solver first hypothesizes a loose spark plug and a test of the hypothesis finds it not to be true. The problem solver hypothesizes that the carburetor is malfunctioning, then refines that guess to the more specific hypothesis of the idle mixture being lean. This is tested and the result suggests that the idle mixture is probably not the problem. Next, the problem solver generates a hypothesis that the carburetor is flooding, refines that to a hypothesis that the float level has become set too high, and tests for that. However, the test result indicates that that is not the problem. The problem solver generates another refinement, that the carburetor needle valve is leaking, allowing fuel in when it should not. This is tested, and is found to be true. A repair is done, and is tested, and results in the elimination of the problem.

These actions are not independent. Results of early actions influence future actions, and results cannot be predicted with certainty. If the spark plugs turned out to be loose, a repair would be done at that point and the problem solving would be complete if that is the only problem. If the needle valve is not leaking, further steps would be necessary beyond those in Figure 1. If, as a by-product of the test of the float level being high, it was determined that the fuel level in the carburetor was not too high, a different hypothesis would have been pursued opportunistically, instead of continuing to pursue the carburetor flooding hypothesis.

This example illustrates several points. First, it shows a situation where complete pre-planning would be inefficient. Not only can the number of possible choices and the variety of possible results of the actions lead to a large, very bushy tree of possible paths, but in many problems much of the tree would not be used. Second, it illustrates that when choosing the next action to take, the next action may not follow from the

[1] This research has been supported by the Army Research Institute for the Behavioral and Social Sciences under Contract No. MDA-903-86-C-173, and by DARPA contract F49620-88-C-0058 monitored by AFOSR. The author wishes to thank Janet Kolodner for her advice and guidance, and Tom Hinrichs, Steve Robinson, and Joel Martin for helpful comments on earlier versions of the paper.

most recent action. For example, step 7 follows from step 3. In addition, following a trouble-tree would not suggest taking advantage of unexpected opportunities, such as following up of the by-product of a test, as seen above.



The nodes represent the different goal instances that have been pursued. They are numbered in the temporal order in which they occurred. The links represent the relationships of which goal instance followed from which goal instance.

Figure 1: A Multi-step Solution in Automobile Troubleshooting.

The problem to be addressed in this paper is three-fold:

1. How can a problem solver efficiently generate successive goals and actions in a multi-step solution?
2. What knowledge is needed to generate the succeeding goals and actions?
3. How should the generation process be controlled and suggestions selected?

We address this problem in the task domain of automobile troubleshooting. Our program, CELIA (Cases and Explanations in Learning; an Integrated Approach), solves problems by generating and achieving reasoning goals. As in the example above, often later goals cannot be generated until earlier ones have been achieved. In addition, the program learns by understanding and explaining the statements and actions of a teacher.[2] The same process that generates goals during problem solving generates them during learning, where they act as expectations of what the teacher will do or say next.

As we will show, our approach integrates the use of four important types of knowledge to generate goals: previous solutions or cases (as in *Case-based Reasoning* [Kolodner and Simpson 1984]), knowledge of relationships between reasoning goals (in this case, knowledge of the troubleshooting process), and causal knowledge, including structural and functional knowledge of the domain. We will discuss generation of suggested actions, and control of the process, and then present an example.

2 Generation of New Subgoals and Actions

We have found four types of knowledge useful for generating subgoals and actions:

1. Case knowledge.
2. Causal knowledge of components, mainly functional knowledge.
3. Structural knowledge of component parts, including part/whole and adjacency relationships.
4. Knowledge of how to do the task, or the relationships between reasoning goals.

[2]Not natural language.

Case Knowledge: Case-based Reasoning (CBR) is a method of using previous episodes to suggest solutions to new problems. CBR is an important problem solving technique because it allows a reasoner to solve problems efficiently when previous similar experiences are available and complete knowledge is not present. In this type of problem, a case provides an ordered set of actions that have worked in the past. Thus remembering a part of a case, or *snippet*, suggests the next action. A next action can be suggested by the case the reasoner is currently reasoning from, or if context changes, by another case that becomes more relevant.

Causal knowledge: A next action or subgoal can also be suggested by causal knowledge. A causal link from a previous action can make the suggestion. Some aspect of the previous goal, for example the test done, the test result, the hypothesis generated, the fix done, something ruled out, is the starting point for the reasoning. The reasoning can proceed from that point forward toward effects of that aspect, or backward toward causes of that aspect.[3] The furthest point of progress in the causal reasoning is suggested as the value for the instance of the reasoning goal to be suggested. The causal reasoning uses both functional and structural knowledge of the domain.

Structural knowledge of components: Part/whole and topological knowledge of components involved in a previous action can suggest the next action. For example, a problem with the electrical system might be due to a problem with the battery. If the most recent instance of a generate hypothesis goal was that the electrical system is faulty, the next appropriate goal instance might be the generation of a hypothesis that the battery is faulty.

Knowledge of how to solve problems in the domain: How problems are normally solved is also important to generating goal and action sequences. Our method uses heuristic knowledge in the form of a set of the types of reasoning goals, or *goal types*, which can follow from each reasoning goal type, called *goal sequitur* knowledge, or sequitur knowledge for short.[4] [5] For example, hypothesis generation goals can be followed by tests of hypotheses, or by further hypothesis generation.

In general, there are many possible succeeding goals a problem solver might generate at any time. Good problem solvers generate goals that can lead them toward their final destination in the most opportune way. In the remainder of this paper we will present a way to choose the next goal or step wisely. We will show that the fourth type of knowledge listed above, knowledge of the problem solving task, is primary to this endeavor, providing guidance for moving toward a solution.

Our method uses three main types of heuristics for this task: *suggestor* heuristics suggest new steps, *restrictor* heuristics constrain the behavior of the suggestors, and *selector* heuristics choose the best of the suggested next steps.

We begin by presenting the four main types of suggestor heuristics:

1. Case Sequential Access
2. Case Direct Access
3. Causal link
4. Refinement (Part/Whole)

The suggestor heuristics, or suggestors, indicate ways to generate possible instances of the consequent goal type.[6] Running these heuristics results in a set of possible succeeding reasoning goals and actions.

[3] Reasoning is uncoupled from the reasoning goal involved. Given a hypothesis that the fuel mixture is too lean, reasoning proceeds from the state that the fuel mixture is too lean (which may or may not be true), not from the *hypothesis* that the fuel mixture is too lean. Thus the causal reasoning does not depend on the reasoning goals involved in the domain, or vary depending on those involved.

[4] *Goal types* are general types of reasoning goals such as clarifying the complaint, verifying the complaint, generating hypotheses, testing hypotheses, interpreting the test results, carrying out repairs, and testing the repairs. They could also be considered subtasks of troubleshooting. Goal instances are specific instantiations of goal types, such as the specific test used to test a specific hypothesis.

[5] As *non sequitur* means an inference or conclusion that does not follow from established premises or evidence, we use sequitur to refer to a goal or action that follows from a previous goal or action.

[6] The borrowing of the logical terms antecedent and consequent should not be taken as an indication that the second goal logically follows from the first. The consequent only plausibly follows from the antecedent.

2.1 Case Suggestors

As noted above, a case provides an ordered set of actions that have worked in the past. Thus remembering a case appropriate to the current context suggests the next action. Multiple parts of multiple cases can be useful in solving a particular problem. Useful parts can be accessed directly, by retrieving the relevant part of a relevant case, or sequentially, by continuing to follow a previous case while it continues to be relevant. The two suggestors that use parts of previous cases are based on these two methods.

2.1.1 Sequential Access

If the results of running a step in the new situation match those obtained when it was run in the previous case, the next step in sequence in that case can be suggested. The **Continue-following-link** suggestor does this.

2.1.2 Direct Access

If the results are different, however, the **Case-snippet** suggestor uses direct access to part of a different case that can provide a suggestion of what to do next. Retrieval involves matching the current situation to the case part, or snippet's goal and context. In our system, CELIA, retrieval via direct access is first restricted to snippets that are centered around the goal type being considered. Then a weighted similarity metric is used, with matching occurring for all features within the context. The context includes the internal context, the results of actions taken up to that point in problem solving, so the retrieved piece is influenced by the results of goals pursued so far in this problem.

2.2 Causal Link Suggestors

Causal link suggestors use domain knowledge of function and structure to reason either forward or backward from a clause in the preceding goal instance in order to suggest the main clause for the consequent goal instance.

Variations in these heuristics include:

- Whether reasoning is forward or backward from the initial clause. For instance, reasoning backwards can lead toward suggesting hypotheses that could be root causes. Reasoning forward can lead to suggesting tests of hypotheses based on their potential effects.
- Which aspect of the previous goal instance to use as the initial clause. For instance, when reasoning from a test of a hypothesis a useful starting point is the test result. When reasoning from the interpretation of a test useful starting points include things ruled in or ruled out.
- Whether the initial clause is returned as a result when no progress is made in the causal chaining. When the consequent goal type is the same as the antecedent goal type, this is not appropriate.
- Whether to return a contradiction of the linked clause, or just the linked clause. For instance, a test for a contradiction of something that follows from a hypothesis can be a good test of the hypothesis.

2.3 Refinement Suggestors

Refinement suggestors use part/whole knowledge to suggest a new goal instance through refinement of the preceding goal instance. Either

- Its component is below the previous goal instance's component in the partonomy, (a *leak in the fuel line* is more refined than a *leak in the fuel system*). The previous goal instance's component is refined to a component that is part of the previous component. or
- The component is the same and the new predicate is more specific, (the *ECM not being grounded properly* is more refined than a *malfunction in the ECM*). This requires use of knowledge of the functions of the involved components. If the predicate is 'malfunction', then those predicates that are involved in obstacles to the component's function are considered as refinements of the previous predicate.

Variations in these heuristics include:

- Whether a clause that is equally as refined is acceptable. When the consequent goal type is the same as the antecedent goal type, this is not appropriate.
- Which aspect of the previous goal instance to use as the initial clause. For instance, refining the interpretation of a test, useful starting points include things ruled in or ruled out.

3 Controlling Suggestions: Restrictors

There are, in general, large numbers of possible next steps that could be generated by the methods above. Restrictors constrain the suggestion process so that effort is not expended trying to generate actions in directions that will not prove fruitful. In general, restrictors rule out goal sequences that are sometimes possible, but are not appropriate in the particular current circumstance. For instance, a test should not follow from a hypothesis that has already been tested, and a hypothesis should not follow from a hypothesis that has already been tested. However, a hypothesis can follow from a hypothesis that has already been refined, it could be another refinement. These examples suggest two of the restrictor heuristics.

No-sibs restricts a goal following from a previous goal to contexts in which no action has already followed from the antecedent.

Only-same-type-sibs restricts a goal following from a previous goal to contexts in which either no action has already followed from the antecedent, or contexts in which only actions fulfilling the same goal type have already followed from the antecedent.

Because some goal types should only follow from the most recent instance of some other goal types, restrictors are necessary for that purpose. For example, an interpretation of a test should follow from the most recent test instead of some previous test. This is clearly not the case for all goal sequences. For example, a number of hypotheses could be advanced, then tested in order, thus the test would not follow from the most recent hypothesis.

Most-recent restricts a goal following from another goal to contexts in which the previous goal was the most recent instance of that goal type.

Also needed are restrictors that constrain what can follow from the interpretation of a test. After a test result has been interpreted, what follows depends on the interpretation. If the hypothesis that is being pursued has been ruled in, either the hypothesis can be refined further, or a repair can be made. If nothing has been ruled in, and something ruled out, it is possible that the complaint should be further clarified. The following two restrictors are used.

Prev-ruled-out restricts a goal following from a previous action to contexts in which the previous actions included ruling out some condition.

Prev-ruled-in restricts a goal following from a previous goal to contexts in which the previous actions included ruling in some condition.

4 Selectors

Even with restriction, several steps might plausibly follow the current situation. Selector heuristics choose the best of those generated. Selectors work in two stages. Before suggestors are run, some selectors specify allocations of computational effort to the different suggestors associated with each of the possible future goal sequences. Then, after generation of plausible next steps, the best next step is chosen based on the rest of the selectors, and the amount of the allocation used.

The selectors that influence allocations include:

1. Allocate more effort to generating possibilities following from more recent goals pursued.
2. Allocate more effort to generating possibilities following from a leaf node of problem solving.
3. Allocate more effort to generating possibilities following from a problem solving node closer to the most recent goal pursued.

These allocations serve to limit the processing suggestors can do before cutting off search. This is important because it keeps the slowest heuristics, such as causal chaining, which can be intractable, from slowing the process down too much.[7]

When suggestions have been made, the choice of what goal and action to take is based on the percentage of allocated effort used in conjunction with the following preferences:

1. Favor possibilities generated using continue-following-link, then case-snippet, then other methods such as causal chaining.
2. Favor possibilities generated using goal sequences judged more likely in our analysis of the diagnostic task.
3. Favor possibilities generated from reasoning goals with more restrictor heuristics. These are less likely to have inadvertently escaped restriction.
4. Favor possibilities generated from reasoning goals with fewer suggestor heuristics. These are less likely to be low quality 'shots in the dark'.

The combined effect of the selectors is to favor continuing following along from the most recent goal, using a previously retrieved case snippet, or a newly retrieved case snippet. The preference is not absolute, however. It does not make the easiest suggestor heuristics dominant, because the allocated effort can vary widely based on the factors discussed above.

5 Example

We will illustrate the process of choosing the next action using the example shown in Figure 2. This is an English-ized version of a sequence of problem solving steps generated by our program CELIA.[8] The problem solver first clarifies the complaint, then verifies the complaint to make sure that the problem can be recreated. The problem solver hypothesizes that the carburetor is malfunctioning, then refines that guess. The idle speed is considered, and rejected. The idle mixture is considered, tested, and repaired, and yet the problems remain. A further hypothesis of the throttle dashpot being out of place is generated, and tested.

1. Clarify the complaint
2. Verify the complaint
3. Generate a hypothesis - carburetor malfunction
4. Generate a hypothesis - low idle speed
5. Test hypothesis - temperature of engine when stalling occurs (warm)
6. Interpret Test - idle speed not a problem; idle mixture possible problem
7. Repair - Adjust idle mixture screw
8. Test Repair - engine still stall? (yes)
9. Interpret Test - idle mixture not the problem
10. Generate a hypothesis - throttle dashpot out of place
11. Test hypothesis - distance between throttle dashpot stem and throttle lever small? (no)

Figure 2: Example Multi-step Solution in Automobile Troubleshooting.

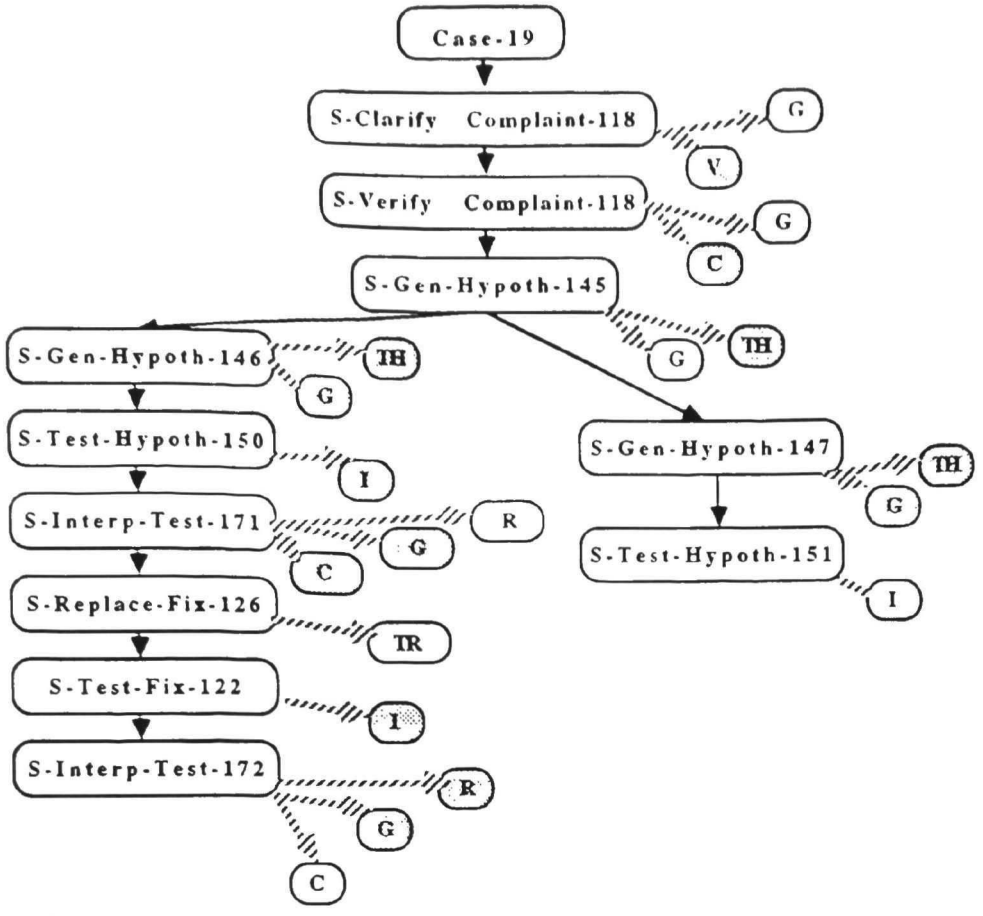
After step 11, the problem solving can be illustrated by the large nodes of the tree shown in Figure 3. Nodes represent goals that have been pursued so far. Links represent the sequencing relationships between the goals. S-Verify-Complaint-118 corresponds to the first step, S-Gen-Hypoth-145 corresponds to step 3 - Generating a hypothesis, in this instance a carburetor malfunction. The most recently completed action is included in S-Test-Hypoth-151. At this point the next action must be generated.

The small ovals in Figure 3 show types of possible succeeding goals that can follow from the parts of the problem solving to this point. The key for the different goal types is given.

The set of possibilities can be reduced significantly using the restrictor heuristics. Shaded ovals in Figure 3 show the effects of the restrictors. These are the directions restrictors have determined not to be fruitful.

[7] Causal chaining is constrained both by the strategy of trying to form a connection between actions rather than trying to form a connection over the large space between complaints and root causes, and by selection allocations.

[8] Actually, it was generated as predictions of what an expert would do by the learning component of CELIA using the same methods as described for the problem solving component. The problem solving component has not yet had the equivalent upgrade from the previous version.



The nodes represent the different goals that have been pursued so far. The links represent the sequitur relationships between the goals. The small ovals connected to nodes with striped arrows represent the possible goal types that can follow from the goal types of the nodes.

Figure 3: Remaining Possible Next Goal Types after restriction.

For each of the remaining possibilities there are several applicable suggestors, these are able to generate 26 possibilities including:

Goal Type	Instance	Following from Piece	Using Suggestor
G-Interp-Test	(Incorrect (Position Throttle-Dashpot))	S-Test-Hypothesis-151	Case-snippet
G-Gen-Hypoth	(High (Contains Carburetor-Float-Bowl Fuel))	S-Verify-Complaint-118	Case-snippet
G-Test-Repair	(Small (Dist Throttle-Dashpot-Stem Throttle-Lever))	S-Replace-Fix-126	Case-snippet
G-Test-Repair	(Low (Position Idle-Mixture-Screw))	S-Replace-Fix-126	Un-Improve
G-Test-Repair	(Increase (Position Idle-Mixture-Screw))	S-Replace-Fix-126	Equivalent
G-Replace-Fix	(Lean (Position Idle-Mixture-Screw))	S-Interpret-Test-171	Fault-Determination
G-Gen-Hypoth	(Hole Carburetor-Barrel)	S-Gen-Hypoth-145	More-Refined
G-Gen-Hypoth	(Clogged Carburetor-Pipe-To-Venturi)	S-Gen-Hypoth-145	More-Refined

From among these, the selector heuristics choose the first action, the interpretation of the test ruling out the hypothesis of the throttle dashpot being out of place. This suggestion was chosen due to several factors:

1. It was generated from the most recent previous goal and actions. Therefore, the suggestor which generated it was allocated a high amount of processing, of which not much was used in retrieving the case snippet that suggested the interpretation.
2. It was generated from a case snippet. Therefore it was favored at selection time.
3. Tests (of hypotheses or of repairs) need to be interpreted, so the judged likelihood of an interpretation of a test following from a test of a hypothesis is high, favoring this suggestion.

6 Related Work and Conclusions

A number of other efforts share some flavor with our approach. Koton [1988] combines use of a number of reasoning methods. First, associations formed from generalizations of cases are tried, then cases, and lastly model knowledge. However, the strict ordering of methods used is less flexible. More importantly, her approach does not generate steps for a multi-step solution, but rather a classification. Carbonell [1986] generates steps for a multi-step solution using a previous case. Domain knowledge is used in adapting the solution, but *one* case will either provide a whole solution or have to be abandoned or adapted. Parts of multiple cases cannot be used. Allen and Langley [1989] generate multi-step solutions using a combination of generalizations, cases, and domain knowledge (in the form of operators). However, they do not retain relations between problems and subproblems, so their DAEDALUS system cannot use an entire previous plan from memory.

Our approach combines the use of several types of knowledge and reasoning techniques. It takes advantage of knowledge about the relationships between goal types to provide constraint on the problem of coming up with the next action to do. The problem solving is flexible and can take advantage of the results of previous actions when deciding what to do next, while remaining goal directed. The approach has three phases – restrictors limit the number of possibilities to be considered, suggestors generate possible next actions, and selectors chose the action to take. There are several advantages to the approach. It combines multiple reasoning methods in a flexible manner. Problem solving is flexible enough to use whatever knowledge is available, using cases when appropriate cases can be found, domain knowledge when it can be useful. It is a flexible way of using parts of multiple cases in forming a solution that is a synthesis of steps. Problem solving can change directions when the results of the problem solving make that necessary. A major side benefit is that many of the suggestor heuristics can benefit when further knowledge is added to the system, in the form of new cases or new domain knowledge. Our system, CELIA, is a learning system, and is designed to acquire such knowledge. This makes problem solving more effective without having to learn new heuristics.

References

Allen, J. A. & Langley, P. (1989). Using concept hierarchies to organize plan knowledge. In *Proceedings of the Sixth Annual International Workshop on Machine Learning.*

Carbonell, J. (1986). Derivational analogy: A theory of reconstructive problem solving and expertise acquisition. In Michalski, R., Carbonell, J., & Mitchell, T., (Eds.). *Machine Learning: An Artificial Intelligence Approach, Volume II.* Morgan Kaufmann, Los Altos, CA.

Kolodner, J. & Simpson, R. J. (1984). A case for case-based reasoning. In *Proceedings of the Sixth Annual Conference of the Cognitive Science Society*, Hillsdale, NJ. Lawrence Erlbaum Associates.

Koton, P. (1988). *Using experience in learning and problem solving.* PhD thesis, Massachussetts Institute of Technology, Cambridge, MA.

Lancaster, J. & Kolodner, J. (1987). Problem solving in a natural task as a function of experience. In *Proceedings of the Ninth Annual Conference of the Cognitive Science Society*, Hillsdale, NJ. Lawrence Erlbaum Associates.

Lancaster, J. & Kolodner, J. (1988). Varieties of learning from problem solving experience. In *Proceedings of the Tenth Annual Conference of the Cognitive Science Society*, Hillsdale, NJ. Lawrence Erlbaum Associates.

Redmond, M. (1989a). Combining case-based reasoning, explanation-based learning and learning from instruction. In *Proceedings of the Sixth Annual International Workshop on Machine Learning*, San Mateo, CA. Morgan Kaufmann.

Redmond, M. (1989b). Combining explanation types for learning by understanding instructional examples. In *Proceedings of the Eleventh Annual Conference of the Cognitive Science Society*, Hillsdale, NJ. Lawrence Erlbaum Associates.

Redmond, M. (1989c). Learning from others' experience: creating cases from examples. In *Proceedings of the Second Workshop on Case-Based Reasoning*, San Mateo, CA. Morgan Kaufmann.