A Case-Based Approach to Creativity in Problem Solving[1]

Janet L. Kolodner & Theresa Louise Penberthy
School of Information and Computer Science
Georgia Institute of Technology
Atlanta, GA 30332-0280
email: jlk@gatech.edu, louise%pravda@gatech.edu

Abstract

One of the major activities creative problem solvers engage in is exploration and evaluation of alternatives, often adapting and merging several possibilities to create a solution to the new problem. We propose a process that models this activity and discuss the requirements it puts on representations and reasoning processes and present a program that solves problems by following this procedure.

Introduction

Often, when people solve problems, they create new solutions by adapting and combining previous solutions they have learned about or created themselves. Sometimes a new solution can be created by merely "tweaking" an old one in routine ways. Often, however, problem solving is less routine, requiring exploration of several alternatives, perhaps adapting and merging several possibilities gleaned from experience.

Consider, for example, a dietician in a nursing home planning meals for its residents. The normal routine is to come up with a standard breakfast and adapt it for those with special diets. For example, a patient with no teeth might be served runny eggs and ground-up toast. A more creative solution, however, might be to give the person without teeth a yogurt milkshake from time to time. The excerpt below shows how one person produced that solution.

```
adapting breakfast for a patient with no teeth,
standard way is to grind the toast in the blender, and leave
the eggs runny

==> evaluate: but that's boring, has lousy texture, and tastes bad.
what can someone without teeth eat that tastes better?

==> elaborate:  perhaps something liquid

breakfast, blender, liquid ==> instant breakfast

==> evaluate:  easy to make, texture right, but taste is
```

[1]This research is supported in part by DARPA under contract number F49620-88-C-0058, and monitored by AFOSR. Mike Redmond and Steve Robinson read preliminary drafts of the paper.

```
    uninteresting.  also, nutrition is wrong -- it should come from
    real foods rather than being artificially created

blender, liquid, real food, nutritious ==> yogurt milkshakes,
made with fruit juice, wheat germ, and yogurt (I drank them a lot as
supplements when I was pregnant)

==> evaluate:  easy to make, texture is good:  in addition to being
    liquid, there's a slight crunch;  nutrition might still be
    insufficient, might need more iron or protein for older people

..., extra nutrition ==> a friend who is a health nut
..., health nut ==> add brewer's yeast for extra protein and vitamins

..., extra nutrition, eating while pregnant ==>
blender, liquid, extra protein, pregnant, ... ==> add a raw egg

==> evaluate: can't use raw eggs any more because of possibility of
salmonella poisoning

==> solution:  yogurt milkshakes, with fruit juice, wheat germ,
yogurt, brewer's yeast

evaluate ease-of-prep:  easy
evaluate texture:  liquid, slight crunch
evaluate taste:  flavorful
evaluate nutrition:  good
```

This reasoning sequence is much like several other examples we have collected of creative problem solving. The reasoner starts with a partial, incomplete problem specification, and through a series of example retrievals and evaluations, eventually both defines the problem more clearly and creates a solution. Several solutions are considered, and the final one has elements of many considered before it. We refer to the process as **exploratory reasoning**.

Exploratory reasoning seems to have many of the elements of *brainstorming*. When brainstorming, a person or group of persons attempts to solve a problem by thinking of the most unusual, farfetched, and "off-the-wall" solutions they can. This helps them expand their thinking beyond the usual and routine, and to consider the problem in a new light. This process continues, as Shouksmith (Shouksmith, 1970) points out, until eventually the problem-solving efforts must persistently focus on refining a proposed solution until it is usable.

Our research goal, and the one reported on in this paper, has been to derive a model of the exploratory processes involved in creative problem solving. We are considering creative problem solving as an extension of more mundane reasoning. Creative reasoning, we think, should not require a different architecture or set of processes. Rather, common processes should be extended and packaged for creative reasoning.

The exploratory processes that we have observed in creative problem solving are similar to the processes involved in *case-based reasoning* (Hammond, 1989; Kolodner & Simpson, 1984; Simpson, 1985). In case-based reasoning (CBR), problems are solved by remembering old situations similar to a new one and adapting the solutions to those problems to fit the new situation rather than solving problems from scratch each time. More mundane case-based reasoning attempts to apply old solutions directly rather than trying out many alternatives, as seen in our protocols.

A case-based reasoner includes in its architecture a memory of cases, retrieval algorithms that can retrieve the best partially matching cases from memory on demand, an adapter that can adapt an old solution to fit a new situation, and a goal scheduler to keep track of the reasoning. Our proposal extends CBR in the following ways:

1. Many cases are considered; many solutions are proposed, and solutions are often made up of combinations of features from several cases.
2. Evaluation of proposed solutions is a primary process.
3. Problem solving is incremental; problem solutions as well as descriptions are updated based on evaluations.
4. Evaluation includes willingness to accept what might not be "right;" odd proposals are considered for what they can contribute rather than being disregarded outright because they won't work.

The process we propose for the exploratory part of creative reasoning is as follows:

- **Retrieve** a set of cases (initially, use the original problem specification as a guide)
- For each case:
  - **Evaluate** the solution proposed by the case for its applicability to the new problem
  - **Evaluate** the solution proposed by the case for its adaptability to the new problem
  - Based on evaluations, **update the problem solution** and
  - **update the problem specification** appropriately
- Repeat until a satisfactory solution is created or found

In the remainder of this paper, we discuss each of these processes, describing the ways it must be extended for exploratory problem solving.

Memory and Retrieval Processes

As we look at the examples of exploratory reasoning that we have collected, we notice that there is a broad selection of cases remembered. Some are routine and some are novel. Each addresses in some way the goals of the problem solver, but none was created with exactly the new goal in mind

(e.g., yogurt milkshakes have always been drunk with sound natural teeth). In other words, they address the problem but this new problem was probably not anticipated at the time those cases were experienced.

Memory retrieval theories espoused in case-based reasoning suggest that indices associated with cases stored in memory indicate what is important about a case, and in most theories, that they also direct and restrict search through the memory (Hammond, 1989; Kolodner, 1983; Schank, 1982). Attempts to designate what makes one case better than another for case-based reasoning suggest that the goal of the reasoner is most important in choosing a best case and, furthermore, that the better the goal of the new case matches one being pursued in the new situation, the better that case will be for case-based reasoning (Kolodner, 1989).

The remindings we see in examining our creative problem solving protocols suggest a few other things. First, the fact that the problem solver's current goal could not have been anticipated at the time an old case was recalled indicates that *a priori* or anticipatory indexing is not sufficient to explain retrieval. Second, the protocols suggest that there often will be no cases that address the primary goals of the reasoner. Third, it seems that stranger cases are preferred over more normal ones.

From the above analysis, several things can be inferred about memory and retrieval. First, retrieval needs to be more flexible than an "index-following" scheme. It needs to allow seeing an old case in a new light. Features that were not salient at the time a case was experienced might be important for retrieval. This means cases should be retrievable both on salient features (that probably were indexed) and on features that were not originally salient, and hence were probably not indexed. PARADYME (Kolodner, 1989) suggests one way to do this: all partial matches are retrieved by a parallel algorithm, and then a set of preference heuristics chooses the best of those, preferring those that match on features marked as salient but allowing other cases to be recalled if no cases with marked salient features are found, and preferring those that can be used to address the problem solver's current goal over others.

The second change these protocols suggest is that ranking of retrieved cases based on which can best address the reasoner's current goal may not always be appropriate. Brainstorming requires being open to remembering things that may not address the current goal but instead might address some higher level or sibling goal. More investigation is needed of the kinds of preferences necessary to rank cases for exploratory reasoning.

Third, there needs to be a way of ascertaining whether recalled cases and derived solutions are ordinary or special.

Evaluation of Applicability and Adaptability

The solutions to cases that are remembered during exploratory reasoning are evaluated for their applicability and adaptability to the problem at hand. When evaluating, the reasoner must ask several questions: whether the old solution or some part of it is applicable to the new situation; whether it or some part of it is obviously inapplicable; whether it or some part of it could be adapted to fit the new situation. When yogurt milkshake is considered as a solution in the protocol above,

for example, it is inserted into the solution in progress because it achieves many of the problems goals. Later reasoning and exploration assumes it to be part of the solution and aims at refining it to solve the problem better.

Evaluation of an old solution might also cause the reasoner to add additional constraints to the problem description. Some will be desirable, and therefore added to the problem description. Others will be seen to be poor, and ruled out in the problem description. When one of us was trying to decide what to do with leftover white rice, for example, she considered making fried rice. However, she rejected that possibility because she had had Chinese food recently, and preferred not to have it again. The problem specification was fixed to rule out any dish of Chinese cuisine as a result of that consideration. Conversely, when she considered making rice pudding, though she rejected that also because it was sweet, she decided to make a baked dish with milk. The problem specification was augmented to prefer a baked dish with milk, such as a casserole.

Evaluation, then, is a key to exploration, since its results are used to both further define a problem and create a solution. But any solution that is remembered could be quite large, and any part of it might be evaluated. The big issue is how to provide focus the evaluation procedures so that they ask appropriate questions of a proposed solution.

Based on an examination of the task of recipe creation, we can propose several places from which these questions derive. First is **function-directed** evaluation. In the recipe creation domain, the purpose is to create something that can be eaten. Thus some evaluative questions arise from the concept of edibility. It is important, for example, to examine the taste and appeal of a dish to see if it is edible. Another set of evaluative criteria are **constraint-related**. Does the proposed solution fit the problem specification? If a dish is to be vegetarian, for example, evaluation procedures must ask if it is vegetarian or could be made that way easily.

A third set are derived from reasoning that occurred in creating previous solutions. We will call them **derivation-driven**. Old solutions themselves provide a rich and important source of questions if the considerations taken into account in creating them are saved. Consider, for example, the task of trying to decide if tofu can be substituted for cheese in tomato tart (a cheese pie flavored with dijon mustard and garlic and with tomatoes on top). The reasoner must be able to evaluate the original ingredient, and determine which of its characteristics are shared by the proposed substitute, which are not, and whether the differences matter. One way the right evaluative questions can be derived is by recalling another case where tofu was to be substituted for cheese. Concerns in that case are likely to be concerns in the new one too. For example, if in the previous case the texture of tofu was compared with the texture of the original ingredient, the reasoner might then ask about texture in the current case.

A fourth set of evaluative questions are **outcome-related**. Some outcome features are well-known and might be asked all the time. Others might be derivable only from other cases. A failure in a similar dish, for example, causes the reasoner to ask whether a similar failure might occur in the new one. If such a failure might occur, repairs suggested by the evaluation of the failure in the previous case will be helpful in fixing the new problem specification or solution.

Of course, if evaluative questions derive from previous cases, then cases must maintain their adaptation or derivation history. They must know where their solutions came from, what was taken into account in creating them, and evaluations of their outcomes.

Updating the Solution in Progress and Problem Specification

Evaluation of old solutions provides guidelines for filling in the solution in progress. A portion of an old solution that fits the new constraints added to the progressing solution. More interesting, however, is that evaluation can also result in update of the problem specification.

In ill-defined problem solving situations, the problem solver may not have all the necessary information at the beginning of problem solving. In these cases, the problem specification must be treated as a dynamically-changing entity. The results of evaluation can help with this. Ruling out Chinese cuisine as a result of remembering and evaluating a Chinese dish provides one example of this. In another instance, remembering a dish that was hard to prepare caused the reasoner to add "easy-to-prepare" to the problem specification. Both of these are cases of refining or adding to the descriptors in a problem specification.

The most interesting examples of changing the problem specification, however, are those in which the goals of the problem solving seem to be changed as a result of a "good" solution that is found or created. For example, in the white rice problem (mentioned above), rice frittata, a breakfast dish, came to mind. While it would certainly use white rice, it is not a dinner dish. Rather than throwing it out because it violated the goals of the problem solving, she changed her mind about the goals: a breakfast dish would be acceptable. In this way, the opportunity to fulfill more important goals is not quashed by the inability to fulfill less important ones.

There are several questions that arise here. First, if essential features of the problem can be changed during problem solving, then how can the goodness of a solution be evaluated? Certainly not based entirely on the problem specification. Second, is it "legal" to change all parts of a problem specification while solving the problem, or are some parts sacred?

We have addressed these two issues by making an assumption that some parts of a problem specification are more sacred than others. To date, we differentiate **primary** from **secondary** specifications, where primary ones are less likely to be changeable than secondary ones. This designation plays three roles during problem solving: First, when retrieving examples from memory, those that match on primary features are considered better matches than those matching on secondary features. Second, when evaluating applicability of a solution, it is judged more applicable if it achieves primary goals than if it achieves secondary ones. Third, when the problem specification is being updated, secondary features are allowed to be updated if an otherwise good solution is found, but primary features cannot be changed without good reason.

Our Implementation

We've implemented a program that solves problems by exploring a library of alternatives, evaluating relevant ones, and based on those evaluations, respecifies its problem and creates a solution. It is based on a protocol of a person attempting to decide what to do with leftover white rice.

The program's initial problem is to come up with a dinner dish that uses leftover white rice. It begins by remembering three dishes: fried rice, yeasted bread with cooked rice baked into it, and rice pudding. It evaluates each suggestion. Fried rice is rejected because the program knew that

Chinese food had been served too recently. It adds a note to the problem specification that Chinese cuisine is ruled out. It is also reminded of rice fritata, a breakfast dish. Considering rice fritata, it notes that it is a breakfast dish rather than a dinner dish, but it does achieve the goal of using some of the rice, and it is well-liked, so the program decides rice fritata should be made for breakfast.

Because the fritata doesn't use all the rice, however, the program continues its reasoning. It considers and rejects the yeasted bread because it takes too much time. It adds a note to the problem specification that the solution should be quick to prepare. The rice pudding is rejected because it is sweet. but based on desirable characteristics of rice pudding (baked, casserole-like, uses milk), it finds macaroni and cheese. Macaroni and cheese doesn't use rice, but it can be adapted to rice and cheese, which does use rice. Together, these two dishes satisfy the program's original primary goal. It proposes making rice fritata for breakfast and rice and cheese casserole for dinner.

While the program is still in its formative stages, it does many of the things discussed earlier in the paper. As it retrieves candidate cases, it augments the problem specification based on its evaluation of the cases. At a minimum, it evaluates alternatives for the appropriateness of their main ingredients and preparation method. When cases it is evaluating "know" how they were created or why they were selected, it also asks evaluative questions based on previous concerns. It updates its solution in progress based on these considerations. Additionally, the problem specification is augmented with desirable features of each recalled case and constrained by undesirable features, as described above. This augmentation is not permanent; when another case is chosen as the basis for remindings, the problem specification is altered to fit it instead.

Discussion and Conclusion

Our claim has been that a primary component of creative problem solving is the process of exploring and evaluating alternative solutions, often merging several solutions into one as a result. Goel & Pirolli (1989) have also observed creative designers redefine and elaborate their problem specifications, explore alternatives, and merge possibilities. However, they do not propose processes. We have tried to make more concrete the processes for doing the things they have studied, and to show how already-known processes can be extended for these tasks.

A key aspect of the creative process is the ability to ask questions. Others studying creativity have made a similar claim. According to Schank & Leake (1986), creativity lies in the ability to ask creative questions, and to use those questions to apply a preset *explanation pattern* (XP) in an unusual way to come up with interesting answers. While we agree that the ability to ask creative questions contributes to creative solutions, we propose that they also allow the reasoner to modify the problem specification, an important task when solving open-ended problems.

The most obvious application of creative reasoning is in those domains that are seen as inherently creative: art, for example, or meal planning, or architecture, and so on. But the approach we have introduced here can help reasoners in many domains.

Any problem solver can reach an impasse as it works to solve a problem. Exploratory reasoning provides techniques to breach such an impasse. By elaborating the original problem specification in more than one way, a problem solver can find different paths to solutions.

In a domain where the problem may be poorly defined, perhaps incomplete or overconstrained, the exploratory process can help elaborate the specification of the problem and provide a more complete basis for problem solving.

There many issues that still need to be addressed, the most important of which, we think, is control of evaluative processes. While we have insight into the derivation of evaluative questions, we still don't know how a reasoner chooses among the many paths that reasoning could take.

An understanding of creative problem solving processes has several important implications. If we understand creative processes, which parts are hard and which are easy, we will be able to create the right kinds of tools to help problem solvers with their tasks. We will find out which processes can be relegated to a machine and which will need to continue to be done by people. This understanding will also help us in building the kinds of tools and developing the kinds of curricula that can best be used to train creative problem solvers of the future.

Bibliography

Goel, V. & Pirolli, P. (1989). Motivating the Notion of Generic Design Within Information Processing Theory: The Design Problem Space. *AI Magazine*, Vol 10, No. 1.

Hammond, K. J., *Case-Based Planning: Viewing Planning as a Memory Task.* San Diego: Academic Press; 1989.

Kolodner, J. L., Selecting the best case for a case-based reasoner. In: Proceedings of the Eleventh Annual Conference of the Cognitive Science Society, University of Michigan, Ann Arbor, Michigan, 1989.

Kolodner, J. L., and R. Simpson Jr., A case for case-based reasoning. In: Proceedings of the Sixth Annual Conference of the Cognitive Science Society. Hillsdale, N.J.: Lawrence Erlbaum Associates, 1984.

Kolodner, J. L., Reconstructive memory: A computer model. Cognitive Science 7, 281 – 328. 1983.

Schank, R. C., *Dynamic Memory: A Theory of Reminding and Learning in Computers and People.* Cambridge: Cambridge University Press, 1982.

Schank, R. C., and D. B. Leake, Computer understanding and creativity. Information Processing 86, pp 335 – 341.

Shouksmith, G., *Intelligence, Creativity and Cognitive Style.* New York: Wiley-Interscience, 1970.

Simpson, R. L., A computer model of case-based reasoning in problem solving: An investigation in the domain of dispute mediation. Tech. Rep. GIT-ICS-95/18, Georgia Institute of Technology, Atlanta, GA, 1985.