A Case-Based Model of Creativity

Scott R. Turner

Artificial Intelligence Laboratory
University of California Los Angeles
Los Angeles, CA 90024
srt@cs.ucla.edu

Abstract

Creating new solutions to problems is an integral part of the problem-solving process. This paper presents a cognitive model of creativity in which a case-based problem-solver is augmented with a set of creativity heuristics. New solutions are discovered by solving a slightly different problem and adapting that solution to the original problem. This model has been implemented in a computer program called MINSTREL.

1 Introduction

Creativity is the pinnacle of human achievement. Intellectual heroes such as Albert Einstein, William Shakespeare, and Thomas Edison are revered for the originality and beauty of their ideas. But genius is not the sole province of creativity. We all create on a daily basis: we fix cars using spare change and bailing wire, invent jokes based on the latest domestic crisis, and make up bedtime stories for our children. The ability to invent original, useful solutions to problems is a fundamental process of human thought.

This paper presents a model of creative reasoning as an extension to problem-solving. When problem-solving fails, the creative reasoner invents new solutions by combining old knowledge in new ways. This model has been implemented in a computer program called MINSTREL, which creates stories about King Arthur and his Knights of the Round Table.

2 Case-Based Problem-Solving

In case-based reasoning, problems are solved by recalling similar past problem situations and applying the solutions from those situations to the current problem (Kolodner 1987; Hammond 1988; Slade 1991). The problem-solver *recalls* a past problem similar to the current problem, *adapts* the solution from the old problem to the current problem, and then *assesses* the results.

The advantage of case-based problem-solving is that the task of adapting and assessing the recalled solution is simple, because the recalled problem is similar to the current problem. The disadvantage is that case-based problem-solvers cannot solve a problem which is *not* similar to a previous problem. When a new problem is encountered the problem-solver has no way to invent or discover a new solution, and problem-solving fails.

3 Creativity

What needs to be added to case-based problem-solving to make it capable of creativity? Consider this example of creativity concerning the seven-year old niece of the author:

> One day, while visiting her grandparents, Janelle was seated alone at the dining room table, having milk and cookies. Reaching for the cookies, she accidently spilled her milk on the table. She decided to clean up the mess herself.
>
> Janelle went into the kitchen, but there were no towels or paper towels available. She stood for a moment in the center of the kitchen thinking, and then she went out the back door.
>
> Janelle returned a few minutes later carrying a kitten. The neighbor's cat had given birth to a litter about a month ago, and she had been over to play with the kittens the previous day. Janelle brought the kitten into the dining room, where he happily lapped up the spilled milk.

This story illustrates three important features of the creative process:

(1) *Creative solutions are both original and useful* (Koestler 1964; Shouksmith 1970). Janelle's solution is original because it has significant differences from all other solutions Janelle knows, and useful because it solves her problem.

(2) *Creativity is driven by the failure of problem-solving* (Weisberg 1986). Janelle could not use the plan she knew for cleaning up a spill because no towel was available. In this case, problem-solving failed because a plan precondition could not be achieved.

(3) *Old knowledge is used to create new solutions* (Koestler 1964; Weisberg 1986). Janelle knew that kittens like milk and that she could find a kitten next door. The creative problem-solver must be able to discover appropriate old knowledge and apply it in new ways to the current problem.

Janelle's example illustrates that a creative problem-solver must be failure-driven, able to use old knowledge in new ways, and able to produce novel and useful solutions.

4 Discovering New Solutions

Case-based problem-solving finds solutions by recalling past problems similar to the current problem. To discover new solutions, the problem-solver must (1) recall a past problem *different* from the current problem and (2) adapt the solutions from the recalled problem to the current problem.

MINSTREL implements this process as three part Transform-Recall-Adapt Methods (TRAMs). "Transform" takes a problem and changes it into a slightly different problem. "Recall" takes the new problem description and tries to recall similar past problems from episodic memory. "Adapt" takes the recalled problem solutions and adapts them to the original problem.

TRAM-Cross-Domain-Solution is an example of a Transform-Recall-Adapt Method. TRAM-Cross-Domain-Solution suggests that a new solution to a problem can be found by translating the problem into a new domain, solving the problem in that domain, and then translating the solution back into the original domain.

MINSTREL uses TRAM-Cross-Domain-Solution to create the following story scene:

> One day while out riding, Lancelot's horse went into the woods. Lancelot could not control the horse. The horse took him deeper into the woods. The horse stopped. Lancelot saw Andrea, a Lady of the Court, who was picking berries.

The original specification for this problem is "create a scene in which a knight accidently meets a princess." TRAM-Cross-Domain-Solution maps this specification into the modern domain and recalls this story:

Walking The Dog

> John was sitting at home one evening when his dog began scratching at the door and whining for a walk. John decided to take the dog for a walk. While they were out, John ran across his old friend Pete, whom he hadn't seen in many years. John realized that he would never have run into Pete if his dog hadn't wanted a walk.

This story is adapted for the original problem by mapping the story back into the King Arthur domain, creating a scene in which Lancelot's horse leads him to Andrea. The resulting scene is creative because TRAM-Cross-Domain enabled the problem-solver to discover a solution different from the known solutions.

There are two major benefits to Transform-Recall-Adapt Methods.

First, TRAMs discover new solutions by looking at *slightly* different problems. A slightly different problem shares many of the same constraints as the original problem, and its solution is likely to have some applicability to the original problem. Looking at slightly different solutions also simplifies the adaptation process, since fewer changes will be needed to adapt the solution to the original problem.

Second, TRAMs further simplify the adaptation task because the transformation itself can be used to guide adaptation. Knowing the transformation that led to a solution, the problem-solver can apply a specific adapation to reverse the effects of the problem transformation. In the case of TRAM-Cross-Domain-Solution, the problem-solver knows to adapt the solution by translating back to the original problem domain. In MINSTREL, each TRAM contains both a problem transformation and a specific adaptation to apply to recalled solutions.

TRAMs are added to the model of case-based problem-solving by augmenting the Recall and Adapt steps. During problem-solving, a TRAM is selected from this pool and applied to the current problem. If the TRAM succeeds in discovering a solution to the problem, problem-solving succeeds. If problem-solving fails, the current TRAM is discarded, another selected from the pool of available TRAMs, and the cycle repeated. The first TRAM selected from the pool is always TRAM-Problem-Solving, a special TRAM that implements normal problem-solving by passing the original problem unchanged to the recall step.

5 Imaginative Memory

The central step of the Transform-Recall-Adapt Method is recalling a solution from episodic memory. But the process of recalling something from episodic memory can *itself* be viewed as a problem. What happens if creative problem-solving is used to implement recall?

To do this, the Recall step of each TRAM is modified to recursively call the creative problem-solving process. To prevent this from leading to endless recursion, TRAM-Problem-Solving is left unchanged, and continues to recall directly from episodic memory.

To recall something, the problem-solver uses creative problem-solving with the problem specification "Find something in episodic memory that matches these features." TRAM-Problem-Solving is the first TRAM used, and passes the recall features unchanged to episodic memory. If an episode that matches the recall features is found, problem-solving succeeds, and memory behaves as expected.

Something more interesting happens when recall fails. If TRAM-Problem-Solving cannot find an episode in memory that matches the recall features, problem-solving fails and a new TRAM is selected. This TRAM modifies the recall features and recursively calls the problem-solving process with the new recall features.

The first TRAM used on this recursive call is again TRAM-Problem-Solving. If the new features recall an episode, the episode is returned to the previous recursion of problem-solving, where it is adapted to the original problem by the previous TRAM, and recall succeeds. *But because the recalled episode was changed by the Adapt portion of the previous TRAM, it is no longer the episode that was found in memory.* Recall has succeeded in a strange way: by recalling an episode that does not exist in episodic memory. Episodic memory has become imaginative.

Treating recall as problem-solving also enables the problem-solver to apply multiple transformations to a problem when necessary. Each time recall fails another TRAM will be applied to the recall features. In this way, a number of TRAMs can be successively applied to a problem. Each TRAM changes the problem in only a small way, but the cumulative effect may be large, enabling the creative problem-solver to discover new solutions

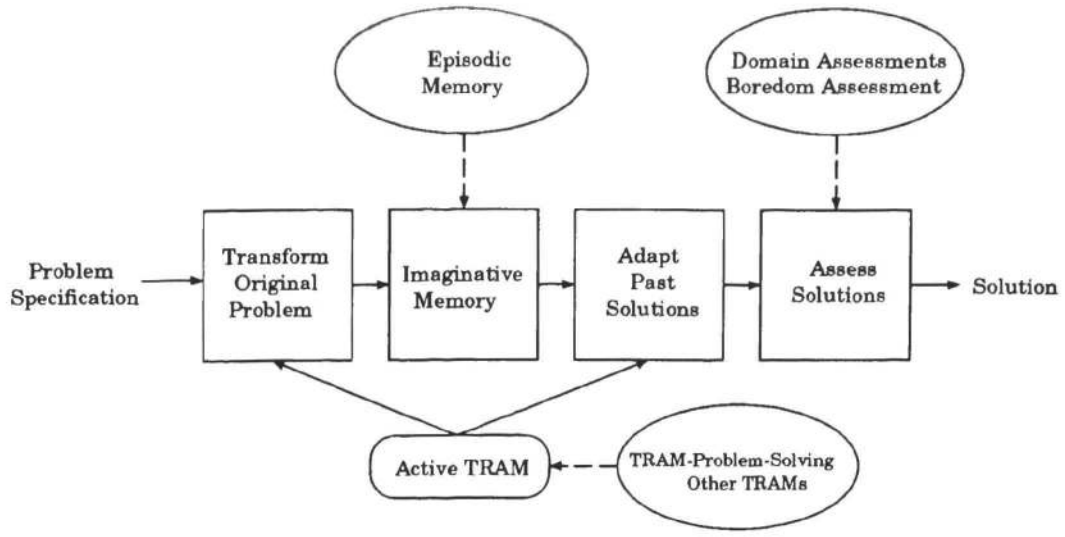

Figure 6.1 MINSTREL's Process Model of Creativity

significantly different from known solutions.

There are two advantages to embedding creativity in the recall process. First, it implements the creative process at a low cognitive level, making creativity transparently available to any cognitive process that makes use of episodic memory for reasoning. Second, it enables the recursive use of creativity heuristics, greatly increasing the power of even simple TRAMs.

6 A Model of Creativity

Figure 6.1 show how Transform-Recall-Adapt Methods and imaginative memory are integrated into case-based problem-solving. An active TRAM controls the Recall and Adapt steps. The recall step uses imaginative memory instead of episodic memory.

The problem-solving cycle begins when a problem description enters the recall step. Initially TRAM-Problem-Solving is in control of the recall and adapt steps. The original problem description is used to recall previous similar problem-solving situations from episodic memory. If recall succeeds, the recalled situations are passed to the adapt step. Under TRAM-Problem-Solving, no adaptation is needed because the recalled solutions are very similar to the original problem, so the recalled solutions are passed along to the assess step. In the assess step, all active assessments are applied to the recalled solutions, and if a solution passes all the assessments, it is output as a solution to the original problem. This is simply the normal problem-solving cycle.

If TRAM-Problem-Solving fails, either because no solutions were recalled or because the recalled solutions failed assessment, TRAM-Problem-Solving is discarded and a new TRAM selected as the active TRAM. The selection of a TRAM is based on the type of problem being solved and the TRAMs previously used.

The selected TRAM transforms the original problem specification, creating a new problem specification. The new problem specification is passed to imaginative memory. If recall succeeds, the recalled solutions are passed to the adapt step, where the active TRAM applies a specific adaptation which reverses the problem transformation. If recall fails, then imaginative memory recursively applies a second TRAM, and the problem-solving repeats at the inferior level.

Adapted solutions are passed to the assessment step. If a solution passes all assessments, problem-solving succeeds. If all solutions fail, the active TRAM is discarded, a new TRAM selected, and the Recall-Adapt-Assess cycle repeats.

7 Creativity in MINSTREL

The primary creative task in storytelling is scene creation – inventing story events to achieve a particular storytelling goal. This section illustrates how MINSTREL invents three scenes in which "a knight kills himself."

Initially, MINSTREL's episodic memory contains two story scenes:

Knight Fight

A knight fights a troll with his sword, killing the troll and accidently injuring himself.

The Princess and the Potion

A lady of the court drank a potion to make herself ill.

Using these scenes and three TRAMs, MINSTREL invents three ways for a knight to commit suicide. Figure 7.1 shows an abbreviated trace of MINSTREL creating these three scenes and generating English descriptions. Except for the deletions marked with "[...]", this trace appears exactly as produced by MINSTREL.

```
==========================================
          MINSTREL Invention
==========================================
Initial specification is #{ACT.105}:
 (A KNIGHT NAMED JOHN DID SOMETHING *PERIOD*
  JOHN DIED *PERIOD*)

RAS Cycle: #{ACT.105}.
 Executing TRAM:GENERALIZE-CONSTRAINT.   [1]
 Generalizing :OBJECT on #{STATE.112}.
 Recalling #{ACT.113}: #{KNIGHT-FIGHT}.

Minstrel invented this story:
 (A KNIGHT NAMED JOHN FOUGHT HIMSELF BY MOVING
  HIS SWORD TO HIMSELF IN ORDER TO KILL HIMSELF
  *PERIOD* JOHN DIED *PERIOD*)
[...]
 [TRAM Recursion: #{ACT.111}.]
    Executing TRAM:SIMILAR-OUTCOMES.     [2]
    Recalling #{ACT.136}:        NIL.
    [TRAM Recursion: #{ACT.136}.]
       Executing TRAM:GENERALIZE-CONSTRAINT.
       Recalling #{ACT.138}: #{PRINCESS-POTION}.

Minstrel invented this story:
 (A KNIGHT NAMED JOHN FOUGHT A DRAGON BY MOVING
  HIS SWORD TO IT IN ORDER TO KILL HIMSELF
  *PERIOD* JOHN DIED *PERIOD*)
[...]
 [TRAM Recursion: #{ACT.112}.]
    Executing TRAM:INTENTION-SWITCH.     [3]
    Recalling #{ACT.126}:        NIL.
    [TRAM Recursion: #{ACT.126}.]
       Executing TRAM:SIMILAR-OUTCOMES.
       Recalling #{ACT.128}: #{KNIGHT-FIGHT}.

Minstrel invented this story:
 (A KNIGHT NAMED JOHN DRANK THE POTION IN ORDER
  TO KILL HIMSELF *PERIOD* JOHN DIED *PERIOD*)
[...]
```

Figure 7.1 Suicide Trace

7.1 TRAM-Generalize-Constraint

TRAM-Generalize-Constraint suggests that a new solution to a problem can be found by removing a solution constraint, solving the new problem, and then adding the constraint back to the new solution. For scene creation, the problem constraints are the features of the specification.

In this example, there are four constraints: (1) the actor is a knight, (2) the object of the state is the actor, (3) the type of the state is health, and (4) the value of the state is dead. TRAM-Generalize-Constraint suggests recalling scenes in which one of these constraints has been generalized.

At **[1]** in Figure 7.1, this TRAM generalizes constraint (2), creating the specification "a knight killed something." This recalls the "Knight Fight" episode. "Knight Fight" is adapted to the current problem by replacing the troll with the knight, creating a scene in which a knight kills himself with his sword. TRAM-Generalize-Constraint has used previous knowledge about how knights kill monsters to create a scene in which a knight kills himself.

Two issues which must be addressed to implement TRAM-Generalize-Constraint are (1) how a feature is generalized, and (2) how a feature is adapted back into the created scene.

MINSTREL uses two methods to generalize a feature. First, the feature can be completely removed from the problem specification. This is the broadest possible generalization and is used to select the feature to be generalized. However, it is so broad that it often leads to the recall of scenes which are difficult to adapt to the original problem. A better generalization would ensure some similarity between the original feature and the instantiations of the generalization. One such generalization is based on *class hierarchies*.

Classes group concepts that have many similarities. The "human" class groups a variety of characters – princesses, knights, kings, and hermits – that have many similar features. By generalizing problem features within classes, MINSTREL is more likely to find a useful reminding.

TRAM-Generalize-Constraint uses class hierarchies to create generalizations. In the suicide example, the "knight" feature is generalized to "monster", because both monsters and knights are in the "Violent Characters" class. This generalization recalls the "Knight Fight" episode, in which a knight fights a troll in order to kill the troll.

The second issue in TRAM-Generalize-Constraint is adapting the recalled episode to the original specification. This is achieved by replacing the generalized feature value with the original feature value throughout the recalled episode. In the suicide example, the troll in "Knight Fight" is replaced with the knight, resulting in a scene in which a knight kills himself by fighting himself with his sword.

7.2 TRAM-Similar-Outcomes

TRAM-Similar-Outcomes suggests that if an action results in a particular outcome, it might also result in similar outcomes. If riding a horse can carry a knight to the castle, then riding a horse might also carry a knight to the woods.

At **[2]** in Figure 8.1, TRAM-Similar-Outcomes recognizes that being injured is similar to being killed, and transforms the scene description to "a knight purposely injures himself." If MINSTREL can recall a

scene which fits this description, it can be adapted to the current problem by guessing that an action which results in injury might also result in death.

However, the description "a knight purposely injures himself" does not recall either of the episodes in MINSTREL's episodic memory. Since recall fails, imaginative memory recurses and applies a new TRAM at the inferior level.

At this new level, MINSTREL applies TRAM-Generalize-Constraint to the description "a knight purposely injures himself" and generalizes the "knight" feature. This description recalls "The Princess and the Potion" in which a lady of the court drinks a potion to make herself ill. This scene is adapted by TRAM-Generalize-Constraint by replacing the "lady of the court" feature with "a knight," resulting in a scene in which a knight makes himself ill by drinking a potion.

This adapted scene is returned to the previous level and is adapted by TRAM-Similar-Outcomes by replacing the illness with death. This results in a scene in which a knight kills himself by drinking a potion, filling the original description "a knight kills himself." Note that in the course of inventing this scene, MINSTREL has also invented the idea of poison – a potion that kills.

The main issue in TRAM-Similar-Outcomes is determining when two outcomes are interchangable. MINSTREL has two methods for deciding this question.

First, MINSTREL assumes that if an action can result in a partial relative change of a state (i.e., drinking a potion results in a partial negative change in health) then the action can also result in a complete relative change of the state (i.e., drinking a potion can make one's health completely negative).

Second, MINSTREL assumes that two outcomes are interchangable if it can recall other scenes in which they are interchangable. MINSTREL assumes that if two outcomes are interchangable in any scene then they are interchangable in every scene.

7.3 TRAM-Intention-Switch

TRAM-Intention-Switch suggests that if the effect of an action in a scene was intentional it might just as well have been unintentional, and vice versa.

At [3] in Figure 8.1, TRAM-Intention-Switch creates the new specification "a knight accidently kills himself." Recall on this new specification fails, because MINSTREL's episodic memory does not contain any episodes in which a knight accidently kills himself.

Problem-solving is used recursively, and TRAM-Similar-Outcomes creates the new scene description "a knight accidently injures himself." This recalls "Knight Fight," in which a knight is injured while killing a troll. Both TRAM-Similar-Outcomes and TRAM-Intention-Switch adapt this recalled scene, resulting in a scene in which a knight commits suicide by intentionally losing a fight with a troll.

Since intended and unintended outcomes are always interchangable, TRAM-Intention-Switch is simple and useful in a wide variety of situations.

8 Status

MINSTREL is written in Common Lisp and contains about 11,000 lines of code and representation. MINSTREL implements twenty-nine TRAMs, twenty-eight author-level storytelling plans, and has an initial memory of approximately ten story scenes. MINSTREL creates a small number of one-half to one page stories based on three story themes, invents novel story scenes as illustrated in the suicide example, and can do simple mechanical invention.

9 Conclusions

Case-based problem-solving cannot invent new solutions to problems. Transform-Recall-Adapt Methods are heuristics that creates solutions by transforming problems into new problems. When integrated into the problem-solving cycle, TRAMs give the problem-solver the ability to invent new solutions and makes creativity transparently available to any cognitive process that uses episodic memory. The power and efficacy of TRAMs has been demonstrated in MINSTREL, a computer program that can tell stories and invent novel story scenes.

References

Hammond, Kristian 1988. Case-Based Planning. In Proceedings of the Case-Based Reasoning Workshop, May 1988.

Koestler, A. 1964. *The act of creation.* MacMillan, New York.

Kolodner, Janet L. 1987. Extending Problem Solver Capabilities Through Case-Based Inference, In Proceedings of the 4th Annual International Machine Learning Workshop

Shouksmith, George 1970. *Intelligence, Creativity and Cognitive Style.* B.T. Batsford, Ltd., London.

Turner, Scott and Reeves, John 1987. The RHAPSODY Manual, Technical Note UCLA-AI-N-85-87, Artificial Intelligence Laboratory, Computer Science Department, University of California Los Angeles.

Slade, Stephen, 1991. Case-Based Reasoning. *Artifical Intelligence Magazine* 12(1):42-55.

Weisberg, Robert W. 1986. *Creativity: Genius and Other Myths.* W.H. Freeman and Company, New York.