The Institute for the
Learning Sciences
Northwestern University

# QUESTION ASKING, ARTIFICIAL INTELLIGENCE, AND HUMAN CREATIVITY

**Alex Kass**

**Technical Report # 11 • March 1991**

# Question Asking, Artificial Intelligence, and Human Creativity

Alex Kass

March, 1991

The Institute for the Learning Sciences
Northwestern University
Evanston, IL 60201

This research was supported in part by the Defense Advanced Research Projects Agency, monitored by the Air Force Office of Scientific Research under contract F49620-88-C-0058 and the Office of Naval Research under contract N00014-90-J-4117, by the Office of Naval Research under contract N00014-89-J-1987, and by the Air Force Office of Scientific Research under contract AFOSR-89-0493. The Institute for the Learning Sciences was established in 1989 with the support of Andersen Consulting, part of The Arthur Andersen Worldwide Organization. The Institute receives additional support from Ameritech, an Institute Partner, and from IBM.

# Introduction

The central claim of this paper is that the key to creativity lies in the ability to ask oneself useful questions. To support and elaborate this claim I shall discuss two computer programs that I have developed, which take two rather different approaches to studying the role of question-asking in the creative process:

- The first program is an artificial intelligence system called **ABE** (**A**daptation-**B**ased **E**xplainer). ABE attempts to develop creative hypotheses by adapting stored explanations to new situations. ABE's ability to tweak its stored explanations is based on a set of hypothesis-adaptation questions that the program knows how to ask itself.

- The second program, called **Sounding Board**, is a piece of educational software that attempts to teach users how to develop creative solutions to problems by teaching them to ask themselves useful questions. Sounding Board is not itself creative, but the combination of Sounding Board and a user can be more creative than the user alone. Sounding Board can help a user put the facts and rules he knows to creative use.

These two programs attack creativity from different angles. ABE asks itself questions in an attempt to be creative, while Sounding Board asks the user questions in an attempt to make the user more creative. These are two different ways of addressing the same fundamental research issues: What role do questions play in the creative process, how can a computer program ask useful questions (of itself or of others), and what classes of questions is it useful for a system to ask?

## AI, Education, Rules, and Questions

Albert Einstein once said, "Imagination is more important than knowledge." He was half right. The essence of his statement is that knowing a lot of facts and rules (the sort of thing generally taught in schools is less important than an ability to use creatively the facts and rules you know. This is an important truth. Traditional education has stressed domain-dependent facts and rules, such as the temperature at which water freezes, and the procedure for converting Fahrenheit to centigrade, way beyond their importance, often to virtual exclusion of teaching anything else. This approach encourages rote learning, and often produces students who are not prepared to apply what they know, or to think for themselves. Within AI the story is pretty much the same; representing knowledge, in an expert system, for instance, generally means representing facts and rules that an expert knows about a particular domain. But this approach to AI has proven quite problematic. For one thing, most experts are quite bad at articulating a set of rules that guide their behavior. This makes it difficult to build rule-bases that reproduce expert behavior, and calls into question the assumption that rules are central to what makes an expert good at what he does. Furthermore, a narrow focus on domain rules tends to result in brittle computer systems whose behavior does not degrade gracefully in novel situations, and which are utterly unable to transfer knowledge between domains.

So if Einstein had said that imagination was more important than knowing many domain rules, he would have been right on target. But the research described below suggests that he was wrong to contrapose imagination and knowledge because imagination *depends* on a kind of knowledge, though not the kind of knowledge typically taught in schools. What imagination depends on is knowing good questions to ask. Some of these questions are domain-dependent, but many are quite general. Solving new problems and developing new theories depend on knowing to ask questions like the following:

- What solved problem does this unsolved problem remind me of?

- Which of the standard assumptions might be wrong?
- What can replace a missing ingredient of the old solution?
- What are the standard causes of things like the thing I'm trying to explain?
- Who might be able to perform an action like the one I am considering?

In addition to these general questions, each domain has many domain-dependent questions. For instance, some questions that might be relevant to ask when trying to come up with creative marketing ideas are as follows:

- What do we offer that our competitors do not?
- What are our customer's needs?
- What success stories can we tell our customers?

The domain-dependent knowledge necessary to answer these questions is important, but it is even more important to know the questions themselves; knowing the questions to ask at least tells what facts and rules you need to find out, but knowing the rules doesn't tell you what questions to ask. Furthermore, the same expert who has trouble telling a system-designer what rules govern his behavior may have less trouble telling him what questions he asks himself when confronted with a problem.

Although this emphasis on question-asking is still not practiced by the majority of researchers, it isn't exactly a new idea either, especially in education. Most smart teachers quickly come to realize that it is more important to turn students into intellectually-curious question-askers than to stuff them full of facts. Knowing facts doesn't necessarily lead to being able to put those facts to good use, but if the child's question-asking propensities are well developed then the child will seek out the relevant facts and rules himself.

The problem is that even when educators realize that they want to teach question-asking skills it isn't clear how to do so. The failure is a theoretical one. What does it mean to teach someone to ask themselves useful questions? What does it mean for an intelligent system to ask itself questions? Abstract statements such as, "Imagination is more important than knowledge," or (more accurately) "Teaching someone to ask good questions is more important than teaching correct answers," are a long way from concrete educational, or system-building suggestions. Concreteness is where computers can really help. In our work we attempt to address the issues of what it means for an intelligent system to ask itself questions, and what it means to teach people to ask themselves creativity-enhancing questions. Specific computational models help us put our suggestions in concrete terms.

# Questions and Creative Explanation

## Overview of this section

This section illustrates how a system that knows how to ask itself a useful set of questions can develop novel hypotheses. It describes the question-asking behavior investigated in the adaptation-based explanation project. We begin with a description of the problem that adaptation-based explanation attempts to address, and a comparison of the adaptation-based approach to other approaches to the explanation problem. The basic approach is described and motivated in the main text, and a set of 21 hypothesis-adaptation questions is enumerated in an appendix at the end of the report. Each question is accompanied by an example showing when asking the question would be useful, along with a sketch of the algorithm employed to ask it

## The problem: Developing creative hypotheses

When someone understands a boring story, most of his work has been completed once he has figured out what the words mean. But when someone understands an interesting story, the real work begins after the language has been parsed. Understanding interesting stories is not a language issue, it's an explanation issue. The difficult part of understanding such stories is developing creative hypotheses about *why* the events that the story describes took place. For example, consider the following:

> **The Swale Story**
> Swale was a star three-year-old racehorse.
> Swale won the Belmont Stakes.
> A few days later, he died.

This story is only three sentences long. There's nothing very tricky about the language in it. Nonetheless, it's the kind of story that an intelligent, curious reader may find quite interesting. It's difficult to *really* understand, because it may cause him to start wondering about what might have happened to Swale.

The ability to come up with hypotheses about what is really going on in a story is a hallmark of human intelligence. The biggest difference between truly intelligent readers and less intelligent ones (*especially* including less intelligent computer programs) is the extent to which the reader can go beyond merely understanding the explicit statements being communicated. It's not hard to build a computer program that can read the Swale Story and can then tell you who died, and what he did before he died. A program with a little inferential capability could also answer questions about how fast Swale probably was, what his life was probably like before he died, etc. The ability to perform simple inference is often required to do text-level tasks, such as lexical disambiguation and resolving pronoun references. But the ability to perform text-level tasks does not constitute evidence of much human-level intelligence; in particular, there isn't much creativity involved. Smart readers do something that is actually rather creative when they read: they dig for explanations. This means more than just inferring those things that the author intended to communicate implicitly; even the author of the Swale story might not know why Swale died. Achieving a creative level of understanding means developing hypotheses about questions for which there may be no conclusively correct answer at all.

## The adaptation-based approach

We have written a program that exemplifies a particular method of developing creative hypotheses called *adaptation-based explanation.* An adaptation-based explainer generates new explanations by retrieving stored explanations from memory and adapting them to new situations. In other words, it applies case-based reasoning to the task of constructing explanations. (For some examples of previous work on case-based reasoning see (Schank, 1983), (Kolodner, 1985), (Simpson, 1985), (Hammond, 1986), and (Sycara, 1987)). How would an adaptation-based explainer develop hypotheses about events such as Swale's death? The answer depends on what kinds of explanations it had already built to understand previous events; an adaptation-based explainer is very dependent on the library of stored explanations it has to work with. The system might be an expert on racehorse death in the sense of having many explanations of previous racehorse deaths stored in its memory. For instance, a veterinarian who specialized in racehorses would be expected to have many such explanations. For such an expert, understanding a story about racehorse deaths is just like understanding stories about going to restaurants for the rest of us. It's done by rote application of a structure in memory. The expert requires no creativity to understand racehorse deaths, and it learns little from them.

It is relatively easy to design a system that can handle stories that exhibit no substantial difference from stories that have been understood before. All that's required is to recognize the similarity. The interesting cases are the novel ones. Does that mean that previous explanations are of no use at all? It does not. Even explanations that are not completely applicable may have sub-parts that can be usefully employed to explain the racehorse's death. Even if a system cannot use the entire explanation it gets reminded of, why shouldn't it use as much as possible? When an adaptation-based explainer isn't reminded of any explanations that can be applied directly to a particular situation, it attempts to modify any explanations of which it does get reminded. It builds new variations that do apply. For example, when hearing about Swale, someone might be reminded of a man who killed his wealthy wife in order to collect the life insurance money. The explanation based directly on this reminding is not applicable to Swale, since, among other things, horses don't have spouses. Nevertheless, this failed explanation can serve as a good starting point for developing the rather interesting idea that perhaps Swale's owner killed him for the property insurance money[1] *if the program knows to ask itself a question such as the following:* ***"Since Swale didn't have a spouse, is there someone else who might have killed him for insurance money?"*** An adaptation-based explainer is one that knows how to ask these sorts of questions.

## Schema application, causal reasoning, and story understanding

Adaptation-based explanation is a schema-based theory of story understanding. As such, it owes a great deal to previous schema-based theories, particularly script/frame theory (see (Schank, 1977) and (Minsky, 1975)). It should serve as a useful introduction to adaptation-based explanation to review the purpose, as well as the limitations of script/frame theory.

The simplest theory of inference is one in which a system has a large corpus of inference rules which it chains together to create explanations. There are many problems with this theory, but the most

[1]To satisfy the reader's idle curiosity I should point out that Swale was under-insured, although this fact was not mentioned in the early newspaper accounts. That would make this explanation seem rather implausible.

important is that it is too inefficient. It is reasonable for a completely novel inference chain to take a long time to build; complete novelty is difficult for people. But inference-chaining systems treat *every* problem as if it were completely novel. The idea of storing frozen inference chains, such as scripts, in memory is a response to that problem.

Scripts don't make thinking easier — they make thinking unnecessary. That's the wonderful thing about scripts. Causal reasoning is a lot of work, so if you know a script that applies to a situation, you don't have to do much causal reasoning. To the extent that a script represents an explanation at all, it is an explanation of the following simple form: Event *X* occurred because script *S* applied in the situation, and one of the events predicted by *S* is *X*. There is nothing in the script that represents the causal relations between the various events.

For example, if you have a suitable restaurant script, you don't need to think about why the waitress brings a menu every time a customer is seated in the restaurant. Thinking through routine events like that every time is a waste of cognitive resources. If we had to do this all the time, we'd never have the time to think any more important thoughts. By freezing expectations in a knowledge structure, the inferencing needed to understand a phenomenon the first time it was experienced can be avoided during subsequent iterations. When actions go as expected, all you need to do is follow the script. Furthermore, the kind of information that the script needs to contain in order for you to apply it is very limited. All you need to know is what to expect and when to expect it. You specifically *don't* need to know *why*.

Things get more interesting when unforeseen deviations from the script occur. What happens when you go to a new restaurant, and the waitress doesn't bring a menu? How do you adjust? Can you just skip over this line in the script as if it were just ceremonial, or is it crucial? Can another action substitute, and if so what? The restaurant script (as formulated in (Schank, 1977), for example) supplies no clues because the restaurant script doesn't tell you why the menu-bringing line appeared in the script in the first place. Usually you don't need to know, but when things don't go as expected, you do. The fact that scripts lack this kind of information makes them brittle knowledge structures that are not adaptable.

When a story doesn't quite match a script, it is difficult to formulate a question that will lead to a new variation on the script, because the script doesn't provide the causal clues to lead the question-asker in the right direction. For instance, one might want to ask what other action might the waitress be performing to satisfy the goals that bringing a menu usually satisfies. The problem is that the standard restaurant script doesn't say what that goal is. The script just tells what to expect, not why. Therefore, it isn't possible to formulate that question at a functional level of specificity. That is why a system whose knowledge is represented in script form is not good at asking hypothesis-adaptation questions. If a system is to be equipped to do anything other than abandon a script that doesn't exactly match the current situation, it needs a richer representation of the causal underpinnings of observed events.

## Adding adaptation to script/frame theory

Adaptation-based explanation is an extension of script/frame theory for handling input that is atypical enough to raise explanation questions. The key to building that more flexible understander is to relax the assumption, which script theory relied on, that schemas will be retrieved and applied in situations that precisely match the situations that those schemas were built to describe. Instead, one must assume that schemas will often be retrieved in situations that are only *sort of* like those they were originally built to handle.

Broadening the range of situations in which a schema will be applied introduces the need to evaluate the newly-produced explanations for problems, and to do some creative adaptation to fix any problems that are found. In an adaptation-based theory of explanation, much of the burden of producing an appropriate explanation is moved out of the schema-retrieval module into the adaptation module (which we call the *tweaker*). The emphasis is shifted from pulling a perfectly appropriate structure out of memory, to being able to work with whatever the best structure pulled out of memory happens to be. Since this adaptation process is more time-consuming than simply applying structures straight out of the schema library, it makes sense to add a storage module as well, allowing the system to save and reuse the new variations of its structures that the adaptation module produces.

In the augmented theory, "retrieve and apply" evolves into "retrieve, apply, evaluate, adapt, and store". Actually, the application, evaluation and adaptation steps are contained in a loop; when a new variation is produced, that new variation is instantiated and evaluated again. If there are still problems with the explanation it can be further adapted. The augmented theory is essentially an application of the case-based methodology to the problem of constructing explanations. (Other case-based reasoning work on adaptation work that has influenced my thinking includes (Hammond, 1986) and (Kolodner, 1985). Some other work that specifically focuses on adapting explanations includes (Koton, 1988) and (Simmons, 1988).)

### Explanation Patterns are adaptable knowledge structures

Understanding stories that raise open-ended explanation questions requires knowledge structures that are specifically designed to encode causal explanations. In order to adapt an old knowledge structure to a new situation, rather than just applying it, the system has to have explanations for why the elements of the structure are as they are. Explanations Patterns (or XPs for short) are structures that explicitly encode causal coherence. This causal annotation is what makes explanatory memory structures adaptable, whereas those that don't encode causal reasoning are brittle.

XPs are explanation structures that are stored in memory. They contain variables so that they can be instantiated to explain new cases. In some ways, XPs are a lot like scripts and their more general successors, called Memory Organization Patterns (MOPS) (Schank, 1982). But XPs serve a different function from scripts and MOPs. The central organizing principle underlying scripts is a temporal sequence of scenes. MOPs add the abstraction and sub-part hierarchies, but are still essentially temporally ordered expectations. XPs, on the other hand, are intended to help process surprising events by giving a causal explanation of the event. (We will use the term *explanation* to refer to an XP that has been instantiated for a specific case.) The central organizing principle is the inference chain rather than temporal sequencing. The causal network leads from a set of explanatory premises through some intermediate beliefs to an explanatory conclusion, which is what the XP explains. The explanatory premises are those beliefs for which no further explanation is given within the XP, although there may be other XPs which can be used to explain them.

For example, the **JOPLIN XP** explains the death of a famous young rock star in terms of a drug overdose. The essential idea of that XP is as follows:

- Success leads to both stress and wealth;
- being a rock star leads to having many drug-using friends;
- having wealth and having drug-using friends leads to easy access to drugs.
- Stress leads to a desire for stress reduction;
- access to drugs combined with desire for stress reduction leads to drug use;
- drug use leads to drug overdose; overdose leads to death.

Computer output of the **JOPLIN XP**, which more closely resembles the internal representation of that knowledge structure appears in Table 1

Because XPs contain an explicit representation of the causal relationships between their components, they are very adaptable knowledge structures. If some component of an XP is inappropriate when the XP is applied in a new context, the system has a relatively easy time determining which parts of the XP must be fixed in order to repair the problem, and which other parts of the XP are affected by the potential repair. This makes it possible to know what questions to ask and how to search for answers.

Table 1

Computer output describing the JOPLIN XP

Assumptions:

?X? was A ROCK STAR.

?X? was young.

Chain of reasoning:

?X? had a HIGH degree of WEALTH LEVEL BECAUSE

?X? was A ROCK STAR [SOCIAL-CAUSE].

?X? had a VERY HIGH degree of STRESS LEVEL BECAUSE

?X? was A ROCK STAR [SOCIAL-CAUSE].

?X? had a HIGH degree of INFLUENCEABILITY BECAUSE

?X? was young [SOCIAL-CAUSE].

?X? had access to RECREATIONAL-DRUGS BECAUSE

?X? had a HIGH degree of WEALTH LEVEL [PRECONDITION-SATISFACTION].

?X? desired that: ?X? was drugged out BECAUSE

?X? had a HIGH degree of INFLUENCEABILITY [SOCIAL-CAUSE] AND

?X? had a VERY HIGH degree of STRESS LEVEL [MENTAL-CAUSE].

?X? TOOK DRUGS BECAUSE

?X? desired that: ?X? was drugged out [GOAL-SAT] AND

?X? had access to RECREATIONAL-DRUGS [PRECONDITION-SATISFACTION].

?X? had a drug overdose BECAUSE

?X? TOOK DRUGS [PHYSICAL-CAUSE].

?X? died BECAUSE

?X? had a drug overdose [PHYSICAL-CAUSE].

## A creative explainer that can adapt XPs

An Adaptation-based explainer consists of (a) a knowledge base containing domain-depended facts and rules, (b) a corpus of hypothesis-adaptation questions (or adaptation strategies, as we sometimes refer to them), and (c) a set of five main program modules, the tasks of which we describe in this section.

The explanation process is triggered when an understanding system recognizes an anomaly of the sort raised by the Swale Story. The specifics of how anomalies are detected depend greatly on what sort of task the understander is engaged in. This detection process is discussed in (Leake, 1990), and is really beyond the scope of this report. To understand the context in which an adaptation-based explainer works, it is merely necessary to assume that an anomaly has been detected. In other words, assume that some event that is not predicted by any currently active memory structure has occurred, that this anomaly has somehow caught the attention of the system, and that the understander has invoked the explainer in order to figure out what might have caused the event.

### Retrieval and Application

The first two steps in developing an explanation are called **XP retrieval** and **XP application**. They involve extracting an appropriate XP from the system's large library, and instantiating it with the variable bindings generated by the current situation. These first two steps in the explanation process are analogous to the structure retrieval and application phases performed by script-based story understanders, such as SAM (Cullingford, 1978) and FRUMP (DeJong, 1977). The structures are different and the retrieval criteria are different, but otherwise, the XPs are treated like scripts during these initial two steps.

If the anomalous event is well explained by the retrieved XP, creative explanation isn't really required. In situations such as this the explanation process is pretty much reduced to script application. But since the needs of a more flexible understanding system require that we assume that many experiences will not completely match any structure retrieved from memory, it is crucial that the explanation process include a post-application step in which the system identifies any weakness that the explanation may have.

### Evaluation

The process of **Explanation evaluation** is a complex one. Explanations can fail in many different ways: they can be inconsistent, incomplete, based on invalid assumptions, or can simply contain the wrong type of knowledge for the system's current purposes. (Leake, 1990) discusses the process of evaluating explanations in great detail. For our purposes, merely assume that the process can be performed. Our only real concern with regard to evaluation is what the output of an evaluator would look like. It is the description how the XP is inadequate which drives the **XP adaptation** phase.

### Adaptation

**XP adaptation** (or *tweaking* for short) is where the real creativity comes in. The tweaker addresses the problems found during evaluation by producing a new variation on the retrieved XP that does not suffer from the problem that the evaluator identified. There are many different tweaking strategies, each appropriate to a particular class of XP failures. Each strategy corresponds to a different question that it may be useful to ask about the failed explanation. For any given failure, the tweaker may know a number of relevant question, and it may have to try asking itself more than one question before producing a variation that satisfies the evaluator.

Actually, the interaction between the evaluator and the tweaker is an iterative one. When the tweaker is done performing an adaptation, the resulting XP must be re-applied and then re-evaluated. The system must ensure that the problem which originally caused the evaluator to call for tweaking has been addressed satisfactorily, and to check that no other problems of importance to the evaluator have been introduced. After each step of tweaking the evaluator has three choices:

- It can decide that the new variation is satisfactory, and should be adopted;
- it can decide that the system should give up on building a variation on the retrieved XP that fits the current case;
- or it can request further tweaking.

### Storage

Finally, when the tweaker succeeds in creating an acceptable explanation, a module responsible for **XP storage** is employed to index the new variant in memory so the results of the tweaker's labor will be available for building new explanations in the future. Script appliers didn't need storage routines because they never changed their structures, but the existence of an adaptation step makes a storage step necessary as well.

The Pseudo-code in Table 2 summarizes the high-level adaptation-based explanation algorithm that we have implemented.

Table 2

High-Level description of adaptation-based explanation algorithm

**LOOP1**

**RETRIEVE XP**

**LOOP2**

**APPLY XP**

**EVALUATE EXPLANATION**

**TWEAK XP**

**END LOOP2**

**END LOOP1**

**STORE NEW XP**

- The outer loop involves retrieving an XP from the system's library and then running the inner loop on that retrieved XP. This outer loop iterates until a satisfactory explanation has been produced, or until the retriever cannot find any more relevant XPs.
- The inner loop (i.e. the apply, evaluate, tweak cycle) iterates until the evaluator decides either that a satisfactory explanation has been produced, or that it isn't worth trying to further tweak the current XP.

## ABE's Questions

In general, the algorithms a system must employ to ask itself hypothesis-adaptation questions can be quite complex. A typical strategy involves four main stages: First, the system must isolate the part of the XP to be altered; then it must search its knowledge base for possible answers to its question; next it must check each potential answer to see if it makes sense in the context of the XP; and finally, if it succeeds in answering the quesiton, it must build a new explanation based on the answer. The search method checks make up the main portions of the question-asking algorithms.

In my doctoral dissertation (Kass, 1990) I propose 21 adaptation questions and describe the algorithms required to ask them in great detail. Those 21 questions are enumerated in Table 3. They are discussed somewhat more extensively, with examples illustrating where they are appropriate, and how they operate, an appendix.

Table 3

21 Hypothesis-Adaptation Questions

1. What other actions are sort of like action Y?
2. What other actions does *X* (or agents like *X*) typically perform?
3. What other actions typically cause events like *Z*?
4. Who would have wanted to bring about one of the consequences of the action?
5. What types of agents typically perform this action?
6. Who could the agent that was originally proposed have caused to perform the action on his behalf?
7. What implements are commonly used to perform this action?
8. What type of implements would this agent typically be expected to have available to him?
9. What implements are typically available in the location where the action took place?
10. Why would the negative effects of the action be particularly unimportant to the agent involved?
11. Why would the positive effects of the action be particularly important to the agent involved?
12. Which of the negative effects of his decision might the agent not have known about, and why?
13. Are there positive effects of the action which might have motivated the agent, but which were not mentioned by the original explanation?
14. Do I know an explanation that could provide a sub-explanation relevant to this premise? If not can I build the sub-explanation on the fly?
15. Do I know an explanation that could explain how event *A* might have caused event *B*? If not can I build the sub-explanation linking the two on the fly?
16. Is it possible that one of the beliefs in the knowledge base that led to contradiction with the XP might be inaccurate for some reason?'
17. Which members of the original agent category would have desired one of the goal states achieved by the action in question?
18. Which members of the original action category typically involve an implement that is observable in the current situation, or that is related to the agent in question?
19. Can the constraint be generalized to make it compatible with the current slot-filler while still maintaining the causal coherence of the explanation?
20. What generalization of slot-filler *A* would be compatible with *B* while still maintaining the causal coherence of the explanation?
21. Can the problematic belief be removed while still maintaining the causal coherence of the explanation?

There are surely more strategies than this waiting to be discovered by future researchers. The point of the project was not to enumerate every possible question that a system might want to ask itself when attempting to adapt an explanation to a new context. Rather, the point was to demonstrate the utility of developing such questions, and to provide enough examples, in enough detail, to make it clear what it is involved when a computer asks itself this type of question.

## Sample adaptations

We developed the adaptation-based theory of explanation by analyzing a set of sample anomalies and associated explanations. Most of the examples you will see in this report revolve around unexpected deaths and disasters. We chose this domain principally because it shows up in the news often, so that it is easy to gather data. It also has the advantage that many of our informal subjects found death-and-destruction events sufficiently interesting and straightforward that they could recall stored explanations, and could produce new ones.

In order to make the operation of the adaptation-based explanation process clear in a concrete way we present here a high-level description of five sample adaptations that were performed by the ABE program. Each example presented here consists of a story that the system attempts to process, an XP that is in some way relevant, a failure that occurs when applying the retrieved XP to the current story, and a variation of the XP that the adapter could produce.

## The Len Bias Story:

College basketball star, Len Bias, died one day after being drafted by the Boston Celtics.

He was the first pick in the NBA draft.

**An XP that might be relevant — The Jim Fixx XP:**

Someone who regularly engages in recreational jogging also has a hereditary heart defect. The stress on the heart from exertion caused by jogging combines with the defect to cause that person to have a heart attack and die.

**A failure that occurs when applying The Jim Fixx XP to the Len Bias Story:** Len Bias wasn't known as a recreational jogger.

**The type of failure this is:** An agent of some action in the explanation is not known to perform that action.

**A question that's appropriate to that failure:** What other actions does the agent typically perform, that could have caused whatever the original action caused in the XP ?

**A result of asking that question about this example:** By substituting playing basketball for recreational jogging, the adapter can build an excellent explanation based on this XP that is appropriate for Bias. A defective heart combined with the exertion from playing basketball, caused a fatal heart attack in Len Bias.

---

One of the strategy-types available to the adapter involves replacing a single slot-filler within a belief in the XP with an alternate slot-filler. The replacement must be more appropriate to the given situation, and should also preserve most of the causal relationships that the original slot-filler participated in.

In this example the tweaker is faced with a situation in which an agent is hypothesized to have performed an action that is not stereotypically associated with him. The tweaker addresses the situation by executing a strategy that replaces the original action with one that is stereotypically associated with the agent.

## The Swale story:

Swale was a star three-year-old racehorse.

Swale won the Belmont Stakes.

A few days later, he died.

**An XP that might be relevant — Spouse insurance XP:**

Some agent is greedy and is married. The agent doesn't love his/her spouse. The spouse has a lot of life insurance. The agent is the beneficiary of the life insurance because spouses are generally the beneficiary of life insurance. Because the spouse has life insurance the agent knows that he/she will get money if spouse dies. Because of greed and lack of love, agent kills spouse.

**A failure that occurs when applying the spouse insurance XP to Swale's death:** Swale lacks a spouse.

**The type of failure this is:** An object slot (spouse) referenced in the XP does not in reality exist.

**A question that is appropriate to this type of failure:** Who would have wanted to bring about the effects of the action?

**A result of asking that question about this example:** By searching for other agents with similar motivations, the adapter can build from this explanation the reasonable hypothesis that Swale's owner got greedy and killed him to collect the property insurance.

---

This example is included to illustrate the point that just as it sometimes helps to replace the filler of an action slot, at other times, the action should be kept as is, and instead it may help to replace the filler of the agent slot.

The **SPOUSE INSURANCE XP** doesn't quite make sense when applied to Swale since racehorses don't have spouses. But when a strategy is invoked which considers who it would make sense to conjecture in the role of killing a racehorse for the insurance money, a plausible explanation is the result.

## Another XP relevant to Bias — The Janis Joplin XP:

A young rock star is very successful. Success leads to wealth and stress. Being a rock star leads to having lots of drug-using friends. Stress leads to a desire for lowering stress. Drug-using friends and wealth leads to access to drugs. Access to drugs and desire for lowering stress leads to taking drugs. Taking drugs leads to a drug overdose. Drug overdose leads to death.

**A failure that occurs when applying the Janis Joplin XP to the Len Bias story:** Len Bias was not a rock star.

**The type of failure this is:** A slot-filler does not fit one of the packaging descriptions (*i.e.* constraints) that the XP specifies.

**A question that's appropriate to that failure:** What generalization of the the original slot-filler might be more appropriate?

**A result of asking that question about this example:** Generalize the Joplin XP to apply to any star performer. This example is included for two reasons: First, to emphasize the point that there is often more than one XP that can produce reasonable explanations for a given example; and second, to illustrate a case in which generalization is an appropriate course of action.

---

The previous example, in which the **FIXX XP** was invoked to explain Len Bias's death, resulted in an explanation that would be plausible if it were not discovered that drugs were involved. But in light of that knowledge, another XP — the **JOPLIN XP** — suggests itself as more appropriate. Much of this explanation applies well to a famous basketball player, but it was originally encoded as applying to rock stars. A tweaking strategy that generalizes the rock star constraint to make the XP applicable to any star performer can be invoked to fix the situation.

## The Pan Am Story:

Pan Am flight 103 exploded in mid air, killing all aboard.

It was en route to NYC from Frankfurt, West Germany, via London, England.

**An XP that might be relevant — The terrorist bombing XP:**

Someone who is engaged in intense political conflict with the people of a particular nation may kill citizens of that nation by planting bombs in crowded areas in that nation.

**A failure that occurs when applying the terrorist bombing XP to the Pan Am crash:** Does not specify enough about who did the bombing. Note that this might not be a problem if the understander were a Pan Am engineer who just wants to know whether it was a design flaw that caused the crash, but it would be a problem if the understander were the government agency responsible for retaliating against the perpetrators.

**The type of failure this is:** A slot-filler is insufficiently specified to satisfy the goals of the understander.

**A question that is appropriate for this type of failure:** What members of the agent category specified in the original XP are known to have an effect of the action as a goal.

**A result of asking that question about this example:** By searching for a more specific description of someone who might have had appropriate motivation to perform the action, the adapter can conjecture that perhaps an Iranian terrorist planted the bomb on the American Pan Am jet in order to retaliate for America's destroying an Iranian airliner.

---

Some adaptation strategies serve the function of making an explanation more specific. For instance, in this example, a generic terrorism XP is applied to explain a plane crash; the explanation that results, that a bomb set off by a terrorist caused the crash, is perfectly reasonable, but it doesn't say much about who the terrorist was. In some situations this might not matter, but in others it would (for instance, the *who* question is crucially important to those responsible for bringing the perpetrators to justice).

Developing variations of the stored explanation that can satisfy the system's need for a conjecture that is less vague requires a set of strategies whose job it is to turn vague slot-filler descriptions into more specific ones.

## A Suicide Bomber Story

A teenage girl exploded a car bomb at a joint post of Israeli troops and pro-Israeli militiamen in southern Lebanon.

The bomber and a number of Israeli soldiers were killed by the blast.

**An XP that might be relevant:**

**The terrorist bombing XP:** Someone who is engaged in intense political conflict with the people of a particular nation may kill citizens of that nation by planting bombs in crowded areas in that nation.

**A failure that occurs when applying the terrorist bombing XP to this car-bombing:** This doesn't explain an important part of the anomaly — why someone would do something that resulted in her own death.

**The type of failure this is:** An action that the explanation claims occurred is not sufficiently motivated by the XP. The explanation seems to claim that a more important goal was sacrificed in order to achieve a less important one.

**A question that is appropriate for this type of failure:** Why might the negative side effects of the action be less important than expected to the agent involved?

**A result of asking the question about this example:** By employing this strategy the adapter can hypothesize that besides having the above political conflict motivation, the bomber was terminally ill, and therefore did not value her own life as highly as most people would.

---

This example points out that sometimes a failed explanation doesn't have any contradiction that needs to be fixed, but instead is incomplete, and needs to be reinforced with sub-explanations. These sub-explanations can strengthen the explanation by providing causal support for a belief that was unexplained by the original XP.

## An Annotated Example of ABE in Action

This section presents an annotated transcript of a the ABE computer program processing a particular story. The objective is to clarify through example some of the details that the program goes through.

The point of ABE is to exercise the hypothesis-adaptation questions and their associated strategies. Therefore, other interesting issues, such as XP selection and evaluation of explanations are glossed over. In fact, evaluation isn't performed by the program itself at all. Instead, the user is asked to evaluate the high-level outline of what happens in this run of ABE is as follows:

- The program is fed an anomaly to explain — in this case it is asked to explain the unexpected death of college basketball star, Hank Gathers.
- It retrieves a set of XPs relevant to explaining deaths and disasters.
- It applies the highest-ranking of the retrieved XPs to the particulars of the Gathers case.
- It presents the explanation to the human evaluator, who decides that the explanation requires tweaking.
- The program queries the human evaluator about just what is wrong with the explanation.
- The system then retrieves hypothesis-adaptation questions that might be applicable to the problem the user has identified.
- The retrieved questions are ranked, and the highest-ranking question is asked. Its associated strategy is queued for execution.
- After the strategy is executed, the resulting explanation is resubmitted to the human evaluator. This version of the explanation is accepted by the evaluator.
- The system updates the knowledge base and adds the XP to the XP library.

*In the transcript that follows the actual computer output is in* `small, fixed-width type`*, while the comments are in* normal type.

To start the ABE program running the user calls the EXPLAIN function, passing it anomaly structure, which represents the anomaly the user wants an explanation for:

```
-> (explain gathers-anom)

Explainer: The anomaly being processed is:
       HANK GATHERS was in EXCELLENT physical condition.
       HANK GATHERS died.
 > was a star basketball player at the Loyola Marymount University.
 > He died suddenly while playing in an important college basketball game.
```

In this example the user requested an explanation of the pre-defined anomaly, named the GATHERS-ANOM, about the sudden death of a college basketball player. The anomaly itself is represented as two beliefs. The system prints out it's paraphrases of those beliefs, along with the text of the story associated with that anomaly. Internally the beliefs are represented in an abstract, conceptual representation from which the program produces crude, english-like paraphrases to help the user follow the system's progress, and evaluate its explanations.

The first step the system performs is to see if it has explanation patterns with indices which match any part of the anomaly. It then ranks those that do match according to how many constants within the anomaly description are specifically matched by the index pattern associated with each tweak.

```
XP-Retriever: Found 7 XPs to consider for this anomaly:
      [5]    FIXX-XP
      [4]    JOPLIN-XP
      [4]    MAFIA-REVENGE-XP
      [4]    KAMIKAZE-XP
      [4]    KILL-FOR-INSURANCE-XP
      [4]    SEX-XP
      [4]    TERRORIST-BOMBING-XP
```

In this case, the retriever extracts 7 XPs as potentially relevant. The FIXX XP is ranked highest because it is a more specific match with the anomaly. The other XPs are indexed as general explanations for death, while the FIXX XP is indexed as an explanation for the death of someone who appears to be in superior physical condition.

The next step, after XP retrieval is to apply the highest ranking XP to see what sort of explanation it produces in the context of the new anomaly. This involves instantiating the XP's variables by matching beliefs in the XP against beliefs in the anomaly and, if necessary, other beliefs culled from the story. Once an explanation is produced it is displayed for the user to evaluate.

```
Applier: Attempting to apply FIXX-XP to GATHERS-ANOM  Found a match for:
                (from xp)     ?X? died
                (from story)  HANK GATHERS died

Evaluator: Evaluating explanation: FIXX-XP.EXPL-1

FIXX-XP.EXPL-1
Premise: HANK GATHERS DID RECREATIONAL JOGGING
Premise: THE HEART OF HANK GATHERS is in category HEREDITARY DEFECTIVE HEARTS
THE HEART OF HANK GATHERS was weak BECAUSE
 THE HEART OF HANK GATHERS is in category HEREDITARY DEFECTIVE HEARTS.
HANK GATHERS ran BECAUSE
 HANK GATHERS DID RECREATIONAL JOGGING.
HANK GATHERS had a heart-attack BECAUSE
 THE HEART OF HANK GATHERS was weak AND
 HANK GATHERS had a VERY HIGH degree of EXERTION LEVEL.
HANK GATHERS had a VERY HIGH degree of EXERTION LEVEL BECAUSE
 HANK GATHERS ran.
HANK GATHERS had a VERY HIGH degree of PHYSICAL CONDITIONING BECAUSE
 HANK GATHERS had a VERY HIGH degree of EXERTION LEVEL.
HANK GATHERS died BECAUSE
 HANK GATHERS had a heart-attack.
```

In this case, the FIXX XP has only one variable, which represents the deceased. This variable is bound by matching the variablized portion of the XP (which mentions a dying action) against the unvariablized portion of the anomaly (which mentions the same action).

Next, the user is allowed to evaluate the explanation which is generated by applying the system's chosen XP to the new situation. The user communicates his evaluation to the system through a series of menu choices.

```
- Please choose one of the following:
     1> Adopt this hypothesis.
        Install XP in memory + record entailed beliefs.
     2> Tweak this XP.
     3> Abandon this hypothesis for now.
     4> Abort the explainer.

Enter Choice -> 2
```

The user has indicated here that he wants the system to try to tweak the XP. Therefore, the system presents a couple more menus which allow the user to identify what type of problem he wishes to identify, and which of the beliefs within the XP the problem applies to.

```
What failure type?
- Please choose one of the following:
     1> Agent-action-mismatch : physical-disability
     2> Agent-action-mismatch : stereotype-violation
     3> Agent-action-mismatch : mental-disability
     4> object-action-mismatch : physical-disability
     5> object-action-mismatch : stereotype-violation
     6> implement-action-mismatch : physical-disability
     7> implement-action-mismatch : stereotype-violation
     8> incomplete-explanation : covers-part-of-anomaly
     9> incomplete-explanation : unmotivated-action

Enter Choice -> 2

Identify the belief that is a problem
- Please choose one of the following:
    1> HANK GATHERS DID RECREATIONAL JOGGING
    2> THE HEART OF HANK GATHERS is in category HEREDITARY DEFECTIVE HEARTS
    3> HANK GATHERS died
    4> THE HEART OF HANK GATHERS was weak
    5> HANK GATHERS had a heart-attack
    6> HANK GATHERS had a VERY HIGH degree of EXERTION LEVEL
    7> HANK GATHERS had a VERY HIGH degree of PHYSICAL CONDITIONING
    8> HANK GATHERS ran

Enter Choice -> 1
```

The user indicates that the problem is a stereotype violation involving a mismatch between an agent and an action. In particular, the part of the explanation that proposes that HANK GATHERS did recreational jogging is objected to because Gathers was not known as a recreational jogger.

Once the problem is identified the tweaking process begins. The first step within the tweak process is to retrieve all the tweaks relevant to this type of stereotype violation.

```
 Tweaker: Attempting to adapt FIXX-XP.EXPL-1
 Tweaker: Failure is: XP Failure:
            Agent-action-mismatch : stereotype-violation
  Retriever: Retrieving XPs indexed under failure:
  Retriever: found - SUBSTITUTE-AGENT:MENTIONED-IN-XP
      DELETE-BELIEF
```

```
SUBSTITUTE-AGENT:MENTIONED-IN-STORY
SUBSTITUTE-AGENT:STEREOTYPICAL-AGENT
SUBSTITUTE-ACTION:CAUSAL-INDEX
SUBSTITUTE-ACTION:AGENT-THEME
SUBSTITUTE-ACTION:RELATED-ACTION ; . . .
```

The system knows quite a few questions to ask about a stereotype violation involving a mismatch between an agent and an action. It should be noted that the names assigned to the questions by the program are slightly different than the names in this paper, but the mappings should be sufficiently clear.

After retrieval the next step is to filter out any strategies whose tweak input filters are not satisfied, and then to rank the rest according to how specifically they match the failure which is the cause for tweaking. (The specifics of the scoring method are beyond the scope of this paper. See (Kass 90) for details.)

```
Filter: SUBSTITUTE-AGENT:MENTIONED-IN-XP is filtered out
      because only one agent is mentioned in the xp.  ; . . .

  Ranker: Ranking remaining strategies.

  Current ranking priorities:
   CW = 1.0    HW = 1.0    SW = 1.0

  TCE   THR   TMS
   0     0     3   Score: 3.0 - SUBSTITUTE-ACTION:AGENT-THEME
   0     0     2   Score: 2.0 - SUBSTITUTE-ACTION:RELATED-ACTION
   0     0     1   Score: 1.0 - DELETE-BELIEF
  Choice point reached: 3 choices available.
  Each choice involves: Trying another tweak on FIXX-XP.EXPL-1
```

After the remaining questions are ranked, the highest ranking is executed while the others are kept in reserve in case the first one doesn't pan out. The program decides to begin by asking whether there is an action that is typically associated with the agent that could have caused the effects of the implausible action.

```
==== Running a tweak:
          Substitute a theme that is associated with the agent
          XP being tweaked:     FIXX-XP
          Anomaly to be explained:
                HANK GATHERS died
          Bindings: X is bound to HANK GATHERS.
          Explanation Failure being fixed: XP Failure:
              Agent-action-mismatch : stereotype-violation
====
Tweaker: Problems is with HANK GATHERS as the AGENT of
           DID RECREATIONAL JOGGING
```

The first step in executing this tweaking strategy is to collect the arguments that the tweak's search-and-inference routine will need. In this case, that means simply determining which agent node (the HANK GATHERS node) will be the starting point for the search for stereotypically associated actions.

The next step is to search for actions associated with Gathers.

```
Tweaker: 5 actions associated with HANK GATHERS are being considered:

-
   Premise: HANK GATHERS had theme WON ATHLETIC AWARDS
-
   HANK GATHERS PLAYED BASKETBALL BECAUSE
   HANK GATHERS is in category COLLEGE BASKETBALL STAR AND
   COLLEGE BASKETBALL STAR is in category BASKETBALL PLAYERS AND
   BASKETBALL PLAYERS had theme PLAYED BASKETBALL.
  Premise: HANK GATHERS is in category COLLEGE BASKETBALL STAR
  Premise: COLLEGE BASKETBALL STAR is in category BASKETBALL PLAYERS
  Premise: BASKETBALL PLAYERS had theme PLAYED BASKETBALL
-
  HANK GATHERS PRACTICED BASKETBALL SHOTS BECAUSE
   HANK GATHERS is in category COLLEGE BASKETBALL STAR AND
   COLLEGE BASKETBALL STAR is in category BASKETBALL PLAYERS AND
   BASKETBALL PLAYERS had theme PRACTICED BASKETBALL SHOTS.
  Premise: HANK GATHERS is in category COLLEGE BASKETBALL STAR
  Premise: COLLEGE BASKETBALL STAR is in category BASKETBALL PLAYERS
  Premise: BASKETBALL PLAYERS had theme PRACTICED BASKETBALL SHOTS
-
  HANK GATHERS TOOK PERFORMANCE DRUGS BECAUSE  ; . . .
-
  HANK GATHERS LIFTED WEIGHTS BECAUSE  ; . .
```

The system comes up with 5 actions that it can associate with Gathers. One of these is directly indexed as associated with the GATHERS node (winning awards) while the others are the result of inference chains, involving the hierarchy of categories which Gathers is in.

After the set of prospective replacement actions is collected, the system checks to see which might actually make appropriate substitutions by seeing whether any of the actions can form inference chains linking to aspects of the original XP that the original action linked to.

```
 Checking a candidate Replacement: HANK GATHERS WON ATHLETIC AWARDS
 Trying to link up to: ("HANK GATHERS ran")
 Does NOT link up with the rest of the XP.
 Cannot create a tweaked explanation based on HANK GATHERS WON ATHLETIC AWARDS
Checking a candidate Replacement: HANK GATHERS PLAYED BASKETBALL
 Trying to link up to: ("HANK GATHERS ran")
 This chain DOES link up with the rest of the xp:
   Inference: HANK GATHERS PLAYED BASKETBALL -> HANK GATHERS ran  ; . . .
```

In this example, the relevant aspect of the original action (recreational jogging) is that it led to running, which led to exertion. The exertion combined with the heart defect led to a heart attack, which led to death. So the system checks to see if the replacements lead to any of these effects. Most of them don't. For example, winning awards does not cause running, physical exertion, heart attacks, or death at least, not by any short casual chain that the system could discover. However, playing basketball does, so it can form the basis of a tweak.

Once a suitable replacement action is found, the system builds a new variation on the FIXX XP in which playing basketball replaces recreational jogging.

```
Creating a new tweaked explanation.

 Adding an index to FIXX-XP-1:
     ?X? PLAYED BASKETBALL
 Adding: ?X? ran BECAUSE
           ?X? PLAYED BASKETBALL.
 Adding: ?X? PLAYED BASKETBALL BECAUSE
           ?X? had theme PLAYED BASKETBALL.
 Adding: ?X? had theme PLAYED BASKETBALL BECAUSE
           ?X? is in category COLLEGE BASKETBALL STAR AND
           COLLEGE BASKETBALL STAR had theme PLAYED BASKETBALL.
 Adding: Premise: ?X? is in category COLLEGE BASKETBALL STAR
 Adding: COLLEGE BASKETBALL STAR had theme PLAYED BASKETBALL BECAUSE
           COLLEGE BASKETBALL STAR is in category BASKETBALL PLAYERS AND
           BASKETBALL PLAYERS had theme PLAYED BASKETBALL.
 Adding: Premise: COLLEGE BASKETBALL STAR is in category BASKETBALL PLAYERS
 Adding: Premise: BASKETBALL PLAYERS had theme PLAYED BASKETBALL

 Deleting a belief: ?X? DID RECREATIONAL JOGGING
 Deleting supporter inference:
  Premise: ?X? DID RECREATIONAL JOGGING
  Deleting supported inference:
  ?X? ran BECAUSE
    ?X? DID RECREATIONAL JOGGING.
 Deleting orphaned beliefs.
```

Making the replacement involves (1) adding in the new belief, along with the inference chain which causes the system to believe it, (2) adding the inferences which link the new belief to its implications in the XP, and then, (3) deleting the old belief from the XP, along with any other beliefs which were around solely to support the belief being deleted.

Once the new XP is built the next step is to repeat the evaluation procedure, allowing the user to see the new explanation and to decide whether it needs any more tweaking.

```
  Evaluator: Evaluating explanation: FIXX-XP-1.EXPL-1

FIXX-XP-1.EXPL-1
Premise: HANK GATHERS is in category COLLEGE BASKETBALL STAR
Premise: THE HEART OF HANK GATHERS is in category HEREDITARY DEFECTIVE HEARTS
Premise: COLLEGE BASKETBALL STAR is in category BASKETBALL PLAYERS
HANK GATHERS had theme PLAYED BASKETBALL BECAUSE
 HANK GATHERS is in category COLLEGE BASKETBALL STAR AND
 COLLEGE BASKETBALL STAR had theme PLAYED BASKETBALL.
HANK GATHERS had theme PLAYED BASKETBALL BECAUSE
 COLLEGE BASKETBALL STAR is in category BASKETBALL PLAYERS AND
 BASKETBALL PLAYERS had theme PLAYED BASKETBALL.
HANK GATHERS PLAYED BASKETBALL BECAUSE
 HANK GATHERS had theme PLAYED BASKETBALL.
HANK GATHERS ran BECAUSE HANK GATHERS PLAYED BASKETBALL.
THE HEART OF HANK GATHERS was weak BECAUSE
 THE HEART OF HANK GATHERS is in category HEREDITARY DEFECTIVE HEARTS.
```

```
HANK GATHERS had a heart-attack BECAUSE
 THE HEART OF HANK GATHERS was weak AND
 HANK GATHERS had a VERY HIGH degree of EXERTION LEVEL.
HANK GATHERS had a VERY HIGH degree of EXERTION LEVEL BECAUSE HANK GATHERS ran.
HANK GATHERS had a VERY HIGH degree of PHYSICAL CONDITIONING BECAUSE
 HANK GATHERS had a VERY HIGH degree of EXERTION LEVEL.
HANK GATHERS died BECAUSE HANK GATHERS had a heart-attack.

- Please choose one of the following:
     1> Adopt this hypothesis. Install XP in memory + record entailed beliefs.
     2> Tweak this XP.
     3> Abandon this hypothesis for now.
     4> Abort the explainer.

Enter Choice -> 1
```

This time around the user finds the explanation satisfactory and requests that it be adopted.

When an explanation is adopted, it is added to the XP library. Since the distinction between the new variation and the original was that the new one applies to people who play basketball, playing basketball is added as an index for the new XP. That way, the next time a basketball player dies unexpectedly, this new variation will be preferred over the one which involves recreational jogging.

In addition, some new hypotheses are added to the knowledge base, such as the belief that Gathers had a weak heart.

```
Explainer: Explanation process successful. Storing FIXX-XP-1.EXPL-1
 Adding #{XP 20 FIXX-XP-1} to the XP library.
 Indices:     ?X? PLAYED BASKETBALL -    ?AGENT? died
 Adding new hypotheses to the knowledge base:
 THE HEART OF HANK GATHERS is in category HEREDITARY DEFECTIVE HEARTS
 THE HEART OF HANK GATHERS was weak
 HANK GATHERS had a heart-attack
Explainer: Explanation process complete.
```

# Questions and Creative Problem Solving

## Educational Software as a Means of Studying Cognitive Issues

In this section we present a different kind of question-asking program. This one aims its questions at a user rather than at itself. The program's goal is to teach the user to solve problems more creatively rather than to be creative itself.

Education is a promising application area for computational theories of cognitive theories (and particularly for computational theories of question asking), but education can be more than that as well. Educational software *of a certain type* can serve as an important alternative methodology to AI software for developing and testing cognitive theories. The Sounding Board program is a program of that type. It is intended to serve the twin goals of demonstrating a useful, innovative approach to educational software, and of developing an alternative computational paradigm for studying the role of question asking in the reasoning process.

The reason that a computational alternative to the AI methodology is desirable is that some of the strengths of the AI methodology also lead to difficult limitations. AI has been successful as a methodology for studying issues in cognitive science for three main reasons:

1. because the task of developing computer algorithms places the focus on ***process* models** rather than on descriptive models, and provides a perspicuous language for describing and comparing such models.

2. because the **discipline of developing implementable algorithms** — specific enough to be implemented on a computer — forces the researcher to examine many details that might otherwise be overlooked. (Many non-trivial processing details *seem* trivial when introspecting because they require little *conscious* thought. Only when one is actually pushed to specify how a computer will perform the process does the complexity of the task become apparent);

3. because the **running computer program** one has built allows one, at least occasionally, to observe surprising behavior that points out strengths and weaknesses of the theory which would be difficult to envision before the program was built.

However, the requirement for detailed specification that provides much of the strength of the AI methodology also limits it. The limitations becomes more and more acute as AI theories become more complex and more knowledge-intensive. In particular, the third advantage, the use of the working program to demonstrate the strengths and weaknesses of the theory is more difficult to achieve in practice, particularly for intensively knowledge-based theories such as the adaptation-based theory of explanation that we discussed in the previous section.

There are really two relevant problems:

1. Getting a knowledge-based program to perform intelligently on an open-ended set of examples requires more than an adequate theory; **a knowledge-based theory of cognition requires a *very* big knowledge base** as well. Consider the adaptation-based theory of explanation: each question that an adaptation-based explainer knows corresponds to a method it can use to search its knowledge base for the domain knowledge necessary to perform an adaptation.

   The domain knowledge is not what the theory is about, but the system needs to have an awful lot of it in order to work robustly. Achieving human-level abilities requires thousands upon thousands of uninteresting facts and rules. So when the program fails to perform at a

human level of performance, it isn't clear why. Is it because of a flaw in the theory, or a gap in domain knowledge?

Furthermore, because the theory of how the knowledge is used will have a big effect on the theory of knowledge representation, the knowledge-base must be a custom-built one. It is unlikely, at least for quite some time, that a new AI theory will be able to run with an off-the-shelf knowledge base. Each new theory generally implies something new about how the knowledge base should be represented; therefore, a new knowledge-base is required for most new theories. For this reason, it is very rare that a knowledge-based AI program ever works on more than a few examples. It is not uncommon for doctoral dissertations to be published based on programs that worked on as few as one or two. The ABE program performed only the five samples discussed in the paper and trivial variations. Even though we believe the theory to be much more general we don't have the knowledge base to prove it.

2. **A theory of cognition is never a theory of *all* cognition, but rather a theory of a few component processes.** For every issue that a researcher addresses he must leave many unaddressed. For every process that he implements fully he must leave others unimplemented. But how then to test the program? The typical strategy is to *kludge* the components that are not the current focus of theoretical attention; in other words, to implement them in a way that is not general — a way that works for the examples the researcher has in mind, but for those only. But this also limits the generality of the overall program. Even if the components that are fully implemented are general, there is no way to run them on an open-ended set of examples because the weak links in the chain, the kludged sections about which theoretical claims are not even being being made, will trip the program up.

So we see that when viewed as a method for studying cognition AI offers at least three significant advantages, but also runs into at least two roadblocks. The obvious question is how one might maintain the significant advantages without having to rely on an inadequate knowledge base, and without having to include kludged modules in the system.

The solution we propose here is to get a human being into the loop. In the standard AI paradigm, the idea is that the system is supposed to operate on its own, as an intelligent system. Human intervention is considered cheating. However, since this approach becomes less practical as the problems being addressed become more difficult and knowledge-intensive, it might make more sense to consider computational models that include interaction with a human user. The idea is to consider the human and the program together as an intelligent system, with the user and the program each making a well-circumscribed contribution to the overall task. The program can structure the overall problem-solving process and provide some of the knowledge (such as the questions to ask), while the user can perform the parts of the processing that aren't part of the theory and can supply much of the domain-specific knowledge.

This notion of dividing the labor between a program and a user also suggests an approach to education. This conception is different from the standard conception of education, in which the student is required to rediscover solutions that the teacher already knows. In this alternative conception of teaching, the student and teacher each play a role in solving problems better than either could have solved individually; the teacher suggests questions that might allow the user to put his domain knowledge to use effectively.

In the following section we discuss a program, called Sounding Board, which implements this user/computer team-up in a very simple fashion. Despite its simplicity, the Sounding Board shows nicely the ways in which the approach can be useful educationally, and suggests how such programs can serve as an alternative way to implement computational models of cognition. After describing the Sounding Board Program we will discuss a somewhat more ambitious program which we are starting

to implement now: A case-adaptation coach that teaches the student to perform the sort of case-adaptation that ABE performed.

## Coaxing versus Tutoring

A central assumption underlying the traditional model of a tutor's job is that the tutor is more knowledgeable about the subject than the student. One implication of this assumption is that the tutor naturally comes to view his role to be one of knowledge transmission. Since the assumption is that the tutor knows the answers while the student doesn't, it follows that the tutor's job is to increase the proportion of the answers which the student does know. The tutor would consider himself supremely successful if the student eventually came to know as much as the tutor himself.

Bringing the student up to the teacher's level of ability through knowledge transfer is not the only role that a teacher can play. Indeed, it may not be the most useful, especially for computer-based teaching. For example, a coach's role often needs to be quite different from that, for the simple reason that the coach may be working with someone whose abilities at the performance task already equal or even exceed his own. The coach's principal value is not that he is better than the student (he may in fact be, but this is neither necessary nor sufficient). His value derives from knowing ways to help any student improve his own abilities. Roger Bannister's track coach never ran a four-minute mile, and Mike Tyson's boxing coach probably holds no illusions about being able to take Tyson's title away, yet these coaches still have something important to offer. Rather than adopting the goal of bringing the pupil up to the coach's level of ability, the coach adopts the goal of extending the abilities that the student already has. Part of the coach's job may involve transferring knowledge the way a tutor does, but much of his job looks more like coaxing than like tutoring. The coach helps by prompting the student at an appropriate moment to think about what he is doing somewhat differently.

Classroom teaching tends to be more like tutoring than it is like coaxing. This is, perhaps because the coaxing model is thought to be appropriate only for developing physical, as opposed to intellectual skills. Alternatively, it may be because teachers believe that when a pupil actually is less able than the teacher the tutoring mode is more appropriate. But our claim here is that neither of these arguments is correct. Education might be a lot more effective if teachers thought less in terms of transferring knowledge and more about coaxing the student to use a certain kind of thought process. Toward that end we are designing computer-based teachers that focus on coaxing rather than tutoring.

(Goldstein, 1977) introduced, and John Brown and his associates (Burton, 1982) have adopted, the useful idea of "coaching" as a model of computer-based education. While Sounding Board follows their notion of a subtler, less intrusive form of intervention, we wish, in advocating the "coaxing" aspect of coaching, to suggest something more radically different from the traditional model of tutoring. Coaxing means asking a student the kinds of questions that will cause him to apply *his own* mental abilities in a new way, rather than concentrating on transferring specific knowledge. It's not just the style of intervention, but the purpose of the intervention, that we have in mind. Graduate advisors are good examples of intellectual coaches. Two aspects of advanced research force the advisor to concentrate on coaxing rather than tutoring. First of all, advising a dissertation project involves helping a student solve a problem for which the answer is currently unknown. Furthermore, by the time the student becomes advanced, he will know more about the specifics of his little corner of the problem than anyone else, including the advisor. The advisor's main role, therefore, cannot be to supply answers. Rather, the advisor must act as a problem-solving catalyst. The second reason that graduate advisors don't act like tutors is that the goal of graduate training is not merely to teach the student the answers to specific questions; much more important is to teach the general skills of asking and answering the types of questions that are important in the particular field of research. The

advisor can coax the student to try a different a path. He can offer questions which might be fruitful to think about; and he can suggest broad problem-solving strategies that worked in the past, but actually implementing the strategies and formulating the answers must remain the student's responsibility.

Coaxing is clearly more appropriate than tutoring when the "answers" are not known, and when the student may have as much domain knowledge as the teacher, but most educational computers will not be training Ph.D. students. However, even in settings where tutoring is viable, such as grade schools, high schools, and professional training settings, coaxing still has much to recommend it, particularly for computer-based teaching. The realization of computer programs that are as intelligent, or even as knowledgeable, as a reasonable school teacher is still a long way off. The tutor model depends on the tutor being a lot smarter than the student; and while this may be true of human teachers, we'd be a lot better off not having to count on it for computer-based teachers.

But even if we imagine for a moment that we could build programs with sophisticated student and domain models, there are still pedagogical motives for preferring coaches. The coaxing model makes the student, rather than teacher, the central participant in the educational experience. The student is encouraged to be more active, and therefore, is likely to be more committed to the experience. Also, coaxing places emphasis on teaching a methodology for skill improvement, as opposed to teaching specific domain knowledge. In the long term, telling the student something he doesn't know is less useful than teaching him to ask the kind of question that will cause him to search for the answer himself.

It should be noted here that the shift of emphasis away from merely transferring domain knowledge toward teaching a method of thinking about problems does *not* imply that one cannot find a way to transmit specific domain knowledge through coaxing; it just takes a somewhat more creative teacher to do so in that mode. For example, a program could coax a student to search for information in an on-line library of case studies — and it could do so without knowing the details of the cases' contents. The rest of this paper is devoted to describing two different coaxing programs. This is the idea behind the case-adaptation coach. But let's start with a program that does not teach domain-sepcific knowledge at all; just concentrates on teaching good questions to ask.

## The Sounding Board

The Sounding Board implements a theory of creativity enhancement put forth by Roger Schank in his book, *The Creative Attitude* (Schank, 1988). The basic premise behind both the theory and the program is that creative problem solving depends on the ability to have one's mind "systematically wander" over aspects of the problem, and that asking oneself a standard set of questions can enhance one's ability to do this. (This is simpler, less structured model than the one underlying ABE.)

In his book, Schank identifies and categorizes approximately 45 questions along the following lines:

**Questions about objects:**

Where does one usually find items like X?

Who usually uses items like X?

What are items like X usually used for?

What problems do items like X cause?

Why do people want items like X?

**Questions about actions and events:**

What usually causes X to occur?

What does X usually cause?

Who performs actions like X?

**Questions about Agents:**

What motivates X to act as he does?

How does X get the resources to act as he does?

Why didn't X do action Y instead of action Z?

The current version of the Sounding Board program knows approximately 700 questions. We started with questions like those suggested by Schank. The question set then evolved, with many new questions added, and some deleted, as we tested the program on sample problems. Some of the questions are very general, such as those above, whereas others are specific to a particular problem type. The program instantiates a variablized question such as those above in order to ask a question that is specifically tailored to the student's current problem. The goal of the program is simply to ask a question that will get the student thinking in an interesting direction.

An excerpt from a typical Q/A dialog with the program looks as follows. The program is asking the questions, the user providing the responses, and noting any additional ideas that occur to him as a result of thinking about the quesions and responses:

**Question: What problem are you working on?**
**Response: I'm trying to land the Motorola account**

—

**Question: What larger objectives would getting Motorola's business help achieve?**
**Response: it would get our foot in the door of the IC industry**

—

**Question: Should we start by focusing the SPECIFIC problem you've identified or the more GENERAL one?**
**Response: getting Motorola's business**

—

**Question: Is getting Motorola's business a matter of maintaining the status quo, or of achieving something new?**
**Response: Achieving something new**

—

**Question: How would you categorize the goal of getting Motorola's business?**
**(For instance, selling, managing, organizing, designing, fixing)**
**Response: selling professional services**

—

**Question: Who is currently doing the business with Motorola that Alex wants to be doing?**
**Response: An internal shop**
**Response: SciTech**
**Response: Some of it isn't getting done at all**

—

**Question: Why should Motorola do business with Alex rather than SciTech?**

Response: We have more experience in the industry

Response: We have more general business sense

Note: I better make sure that I read up on all the latest innovations in chip manufacture before I make my presentation

—

Question: How can you impress upon Motorola that your advantages are important?

Response: tell them stories of less experienced competitors

—

Question: What success stories could Alex tell Motorola?

Response: none

—

Question: What success stories could you MAKE relevant.

Response: There was the time we reduce Apple's inventory needs by 20%

Note: Be sure to have Joe get me the writeup of the Apple engagement

—

Question: Who is really great at solving problems like getting Motorola's business?

Response: John Smith — the boss where I used to work

—

Question: How could Alex get John Smith's help?

skipping inappropriate question

—

Question: What would John Smith do in this situation that Alex could do?

Response: Tell a lot of good success stories

—

Question: What might keep Motorola from appreciating the advantages Alex offers?

Response: Our staff is young

Response: They might think that we are too expensive

Response: The MIS manager there is kind of threatened by me

Note: Make sure they realize that our solution is less expensive in the long run.

—

Question: Who else within Motorola might be better to deal with?

Response: Maybe the operations manager

Note: Make sure the operations manager is contacted about the presentation.

—

Question: What recreational activities is Alex really good at?

Response: I'm really good at chess

—

Question: How could you apply the skills that make you good at chess to getting Motorola's business?

Response: Maybe I should do more complex strategic planning

—

Question: What sets this client apart from others who Alex has done business with?

Response: They're bigger

Response: They do most of their business with the government

—

Question: How should Alex adapt strategy to take the special qualities of Motorola into account?

Response: We might want to especially emphasize that the firm has experience with large clients

[USER INDICATES A DESIRE TO WORK ON A NEW PROBLEM]

—

Question: What problem are you working on?

Response: I'm having a lot of trouble with Bill

—

Question: Would you describe this issue as a personal problem or an interpersonal problem?

Response: interpersonal problem

—

Question: What about Bill causes problems for Alex?

Response: He never does his status reports on time and they are usually incomplete

—

Question: What reasons does Bill give for not doing status reports adequately?

Response: he says he has higher priorities

—

Question: Why do you think Bill is not doing status reports adequately?

Response: I think he hates paperwork

—

Question: What skills or knowledge does Bill lack which are required to do status reports adequately?

Response: he doesn't write well

—

Jumping to a previous question

—

Question: Why do you think Bill is not doing status reports adequately?

Response: lacks motivation

—

Question: What unintentional rewards are there for not doing status reports?

Response: people meddle in your work less

Response: you can leave for home earlier in the day

—

Question: What can be done to stop associating people not meddling in your work with not doing status reports?

Response: I could scrutinize his work especially carefully when I don't get reports

—

**Question: What unintentional punishment is there for doing status reports?**

**Response: It consumes time**

—

**Question: What can be done to stop associating it consumes time with doing status reports?**

**Response: I could make it clear what I do and don't really need to know**

—

**Question: What does Bill value?**

**Response: autonomy**

**Response: recognition**

—

**Question: What other situation does this problem remind you of?**

**Response: It reminds me of when I kept trying to get my girlfriend to go to a movie she didn't really want to see**

—

**Question: How was that situation resolved?**

**Response: I never wound up seeing that movie and we eventually broke up**

**Note: Make sure not to push Bill too hard on this. Use subtle persuasion and keep things in perspective.**

Many of the questions that the Sounding Board asks are rather obvious. That doesn't mean that the user would necessarily have thought to ask the questions without the program's prompting. Perhaps *because* they are so obvious users often fail to ask themselves such questions *explicitly*. As a result, the ideas that can be sparked by asking the questions don't occur. As the sample dialog above exemplifies, the answers to obvious questions can lead in interesting directions.

## Coming up with the next question

Sounding Board does not use much domain knowledge to perform its task. All of the knowledge is encoded in the questions. Behind-the-scenes inferences are not performed. The basic mode of interaction with the program is not unlike Weizenbaum's Eliza (Weizenbaum, 1966). The program asks a natural-language question, and the user gives a natural-language answer. But the similarity ends there because the objectives of the two programs — and, therefore, the types of questions and the criteria and mechanisms used for question selection — are very different.

Eliza was trying to mimic intelligence, whereas Sounding Board is trying to encourage intelligence. Since imitating a psychotherapist was Eliza's goal, its main task was to find a key-word in the input for which it knew something therapist-like to say. For instance, if the user types *any* sentence containing the word "computer" the program might respond with, "Do computers make you uncomfortable?" On the other hand, Sounding Board's objective is to make the user consider the problem from a useful angle rather than to ask whatever question makes the program appear to know something about a word the user has mentioned. Therefore Sounding Board must respond to user input according to the *type* of problem the user's input refers to. For example, suppose the user indicates that he is trying to make himself more productive. When the program asks, "What is the cause of your productivity problem?", the user answers, "I don't have the computer I need to get my job done." Because the program characterizes this as a missing resource problem, it will go on to ask resource-

related questions, *e.g.,* what the user needs the computer for, how one usually gets a computer, etc. On the other hand, suppose the user instead answered, "My boss keeps making me switch to a new kind of computer." This isn't a resource problem, it is a case of an agent performing an unhelpful action. Therefore the program would proceed in a different direction; it would start asking about the boss's motivations.

To summarize the comparison: Eliza embodied a rather simplistic theory of how to simulate intelligence by indexing the program's questions by key-words that could be expected in the user's input. Sounding Board proposes a slightly more complex mechanism: indexing the program's questions by the categories of answers that the user might give to the program's previous question, to achieve the simpler task of encouraging creative thinking.

The major technical issue not faced when designing AI systems, which must be faced when building educational software, is user interface design. For Sounding Board the main issue was how to allow the user to answer the program's questions. A multiple-choice interface would not allow the user to think in the open-ended way that is required to develop creative solutions. A natural-language interface would allow for open-ended input, but is not realizable since robust natural langauge process requires a great deal of domain knowledge, and the whole premise is that the domain knowledge resides in the user, while the program knows only the questions to ask.

Luckily, the program doesn't really have to understand the implications of an answer. It just has to categorize it. If the user mentions a resource problem, for example, the program doesn't have to infer the implications of missing the resource — that's the user's job. All the program has to do is categorize the problem as a resource problem and extract what the missing resource is.

About one third of the time, categorization task can be accomplished by associating a set of simple response pattern with each question, and seeing if any of the response patterns match the user's answer. The program only relies on this method when the user's response is simple enough that a conclusive interpretation can be made. Most of the user's answers are too unpredictable to be reliably handled by pattern matching.

To handle the majority of cases, in which the program either cannot find a match or has an uncertain match, the system asks the user how to interpret the answer. It presents a categorization/confirmation screen which allows the user to help the Sounding Board interpret the answer. First the program presents the various answer types that it recognizes for the question (e.g., when the program asks what the problem is, it then wants to identify whether the answer indicates a resource problem, an interpersonal problem, or an unachieved goal problem, etc). Then, once the user indicates the category that his answer fits into, the program allows the user to specify the slot-fillers that are relevant to that answer type. For example, if the user specifies that his answer is about an unhelpful agent, then the program will request that the user specify who the agent is, and what problematic action he is performing. In addition to making the answer understandable to the program, this process forces the user to rethink his answer and to internalize the categorization scheme.

## Types of Questions

The main theoretical issue we have faced while building Sounding Board concerns the content of the question theory: What categories of answers should the program recognize for each question that it asks, and what question should the program ask after getting a certain category of response to the previous question? That theory is still very much in flux. Our method for developing it is largely empirical. As we get subjects to use the program, certain questions that the program should have asked become obvious, so we add them. After doing this for a while we began to see patterns develop. At this point we are aware of seven categories of questions that it is useful to have the

Sounding Board ask. Codifying these categories helps us develop the question-base further. For instance, we can now develop new follow-ups by running through the set of categories, and asking ourselves if there are any questions of that type that would make good follow-ups. As the theory of question categories matures we hope to write software tools that will make use of that theory to help question writers develop the question base further.

The categories of questions we have identified thus far a as follows:

**Questions for internal use:**

Some of the questions the program asks are pretty dull for the user to answer, but the answers are important because they allow the program to follow up with more interesting questions. A challenge to the question writer is to strike a balance, limiting the number of dull questions as much as possible, and never stringing too many together in a row. There are two main question-types in this category:

1. **Problem-classification questions.** Example: "What problem are you working on?" These help the program figure out what kinds of questions are appropriate.

2. **Slot-filler identification questions.** Example: "Who are your competitors?" These help the program instantiate the follow-up questions with specifics appropriate to the user's problem.

**Thought-Provoking Questions:**

Most of the questions the program asks are intended primarily to make the user think about the answers, rather than to get answers that the program will use. The following are the categories of such questions that we have identified thus far:

3. **Attention-focussing questions.** Example: "What do you offer that your competitors do not?" These focus the user's attention on parts of the problem he may have been taking for granted.

4. **Barrier-busting questions.** Example: "What computer would you get if price were no object?" The idea of these questions is to eliminate some barriers from consideration long enough for the user to realize what he really wants.

5. **Reminding-facilitating questions.** Examples: "Who would be really good at solving problems like getting Motorola's business?" "How have others solved this problem?" These questions get the student to think about solutions he may already know.

6. **Context-switching questions.** Example: "What recreational activities are you good at?" Sometimes it is helpful to completely forget about the specific problem-at-hand for a moment, and then to follow up by bringing the new context to bear on the old by asking a question about how they relate, such as, "How could you use your chess skills to help you land the Motorola account?"

7. **Domain-level problem-solving questions.** Example: "Is there some way that parallel processing could help here?" If the program knows questions that are specific to the domain of the problem (as identified by problem-classification questions), then these can also sometimes be helpful.

## User Control

The basic premise underlying Sounding Board is that the student will know more about the problem he is working on than his electronic teacher. This implies that the student/user should maintain as much control over the direction that questioning will take as possible. The philosophy we have adopted is that the program should supply *initiative*, while the user maintains *control.* Thus, a passive user will be asked what the program believes are the most appropriate questions at any point, while

the interface to the program allows several different avenues for a more active user to change the course of the questioning at any point. The following facilities help the user exercise this control.

- An **Inappropriate Question** button tells the program to go down track that is slightly different from the one it is currently pursuing. This is important because sometimes a question that makes sense in most contexts doesn't in the user's specific context. For instance, in the transcript presented earlier, the program started one line of questioning by asking who would be good at solving the user's problem. The user responded by naming a competitor, and the program followed up by asking how the user could get the person to help him. Had the user identified a friend or colleague this line of question might have been useful, but in the current context it might prove annoying. By telling the program that the question is inappropriate, the user can get to change tracks. In this example, it instead asks how the user could become more like the person who is good at solving the problem.
- A **History button** allows the user to examine all the questions and answers that have been given so far and to go back to them and reconsider them, in order to add, change, or recategorize answers.
- A **Redirect menu** allows the user to jump to a very different kind of question at any point. For instance, if the user gets bored with specific questions about sales techniques, he could choose to switch to questions about the big picture, or about decision-making, or other situations that this one reminds him of.

# Future Work: A Case-Adaptation Coach

ABE explored the role of question asking in the context of a rather structured, case-based theory of hypothesis formation. In contrast, the question-asking theory explored by the Sounding Board project is much less structured. It is a more free-form, brain-storming kind of theory, rather than a case-based one. However, the educational mode is not, in theory, limited to such free-form theories. For instnace, It can be used to explore the case-adaptation theory as well. We have just begun to design such a program, which call the Case-Adaptation Coach. In this section we describe the outlines of this new project.

The goal of the Case-Adaptation Coach is to teach the student to think creatively about business problems, using Case-Based Reasoning. Three major skills a student must learn in order to reason based on cases are as follows:

1. To change contexts and apply stored experiences in novel ways.
2. To suspend judgement about solutions that don't quite work in their initial form.
3. To realize that sometimes a bit of creative tweaking is better than either abandoning an old solution altogether or ignoring the potential problems with applying an old solution in a new situation.

The basic steps that a session with the program would include are as follows:

1. Present typical textbook problems along with the textbook solutions.
2. Present a difficult problem that is in some way related to at least one of the textbook problems, but for which no textbook solution will work without modification.
3. Coax the student to select one of the textbook cases presented in step 1 to use as a starting point in building a solution.
4. Coax the student to identify concerns with the solution he has chosen.

5. Coax the student to identify ways of building a new variation on the old solution that solves the new problem.

The preparatory steps, 1 and 2, are fairly straightforward, and non-interactive. But steps 3-5 each involve an interesting coaxing module. Each step requires a different kind of skill and calls for different kinds of coaching.

### Coaxing case selection

The idea is to coach the student to remind himself intentionally. This phase begins with the program asking a very broad question, "Does this new case remind you of any of the cases we've discussed?". If the user isn't reminded immediately the program would coach him to remind himself by asking, "What features characterize the current problem?" and then, for each feature that the student mentions, asking if any of the textbook cases shared any of those features. This should generally be enough to help the student notice similarities between the new case and one or more textbook cases. If it isn't, the program can attack the problem from the other direction. It can run down the list of textbook cases, ask the student to verbalize the key features of each, and then ask if the current case shares any of those features.

### Coaxing problem detection

Here again the idea is to start with rather broad questions, and to become more specific only if necessary. The program would begin by asking a question like, "Do you have any concerns about applying the solution that worked for the 123 Company to the case of the XYZ Company?" If the student voices no concern, the program could push a little harder by asking, "Aren't there any differences between the companies that might be relevant?" If a question at this level prompts a concern (for example, the student might indicate that the companies are in different industries), then the program would try to elicit possible implications of the differences with question along the following lines: "How might the fact that the companies are in different industries be relevant to the proposed solution?" The idea is to force the student to explicitly identify a problem he can then fix in the modification phase.

### Coaxing solution modification

Once the student identifies a potential problem with the Case-based solution (for example, "Perhaps the textbook solution would be too expensive in XYZ's industry") the Coach attempts to suggest very general modification strategies that the student might try in order to fix his solution. A number of researchers have identified domain-independent case-adaptation strategies (see, for example, (Carbonell, 1986), (Hammond, 1989), (Collins, 1977), (Kass, 1990)). The strategies are of the following sort:

- Reorder the steps of a plan to eliminate a bad interaction between plan steps.
- Find an alternate method of achieving one of the goals that doesn't cause a negative side effect.
- Combine the current plan with another known plan that achieves whatever goals the current plan doesn't.
- Replace an action that one of the agents couldn't perform with another action that he typically does perform, and which could cause same effects as the original action.
- Provide additional motivation to an agent who might not otherwise do the job required.

Case-adaptation programs have typically tried to select the best strategy and implement it all on their own. This is a difficult, knowledge-intensive task for a program to perform. But it would be neither necessary nor desirable for a case-adaptation coach to do all that. The coach would just be responsible for suggesting some relevant strategies. It is the student's job to make the final selection

and to actually decide what it means to apply a particular strategy to a particular problem. A coach needs only a high-level understanding of its own case library to coax the student toward a creative solution by suggesting a few broad adaptation strategies that might be useful to apply.

# Summary: Using Computers to Study Creativity

This paper has addressed two major issues: The role of question asking in creativity, and the ways that computers can be used to study that role.

The central point is that although researchers in both AI and education have traditionally given primary importance to domain-level facts and rules, there is a kind of knowledge that is even more important. The most important thing that a person or system aspiring to creativity needs to know is how to ask useful questions. Knowing how to ask oneself a question means knowing how to put one's knowledge to use. Sounding Board demonstrates that even a simple program, with very little domain knowledge, can help a user put his knowledge to use, thereby facilitating the creative process by asking useful questions at the right time.

Among other things, ABE and Sounding Board demonstrate that there are really two parts to knowing how to ask oneself questions: One is knowing *what* questions to ask, and the other is knowing *how* to ask oneself those questions. Sounding Board addresses the "what to ask " part but not the "how to ask" part. Sounding Board merely suggests the questions to ask, leaving all the issues related to how to ask those question to the user. This approach makes it easy to explore many different question types quickly and easily. Adding new questions to the Sounding Board system is a relatively simple task compared to adding new questions to ABE's library. For that reason, it has been possible to equip Sounding Board with hundreds of questions in less time than it took to develop a set of twenty-one question-asking strategies for ABE.

Since each question in ABE's library corresponds to a detailed memory-search and inference algorithm, ABE is not quite as facile a vehicle for exploring the space of possible questions. But what ABE lacks in breadth it makes up in depth. ABE specifies a detailed theory of what it actually means to ask oneself a question. It also specifies in precise terms the types of facts and rules (and the structure of that knowledge) needed to answer the questions that it specifies.

ABE and Sounding Board thus have rather complementary strengths as cognitive science research tools. This illustrates a more general methodological point we have tried to make in this paper. AI software and educational software provide two complementary methodologies for using computers to study cognition. Each provides the core advantages inherent in the computational approach, including highlighting process models and enforcing the discipline of producing implementable algorithms. Building an intellectual coach has the additional advantage of allowing a theory to be tested even while parts of the process model remain unspecified (since the student can fill the gap). On the other hand, developing AI programs forces the researcher to be even more precise about the process models being proposed, thus facilitating a deeper, more detailed analysis of the individual components.

In practice, the mode of inquiry chosen to attack a particular problem is generally a function of how a particular researcher has trained, rather than being based on a conscious decision about what sort of tool will best address the questions that the researcher wants to answer. This has usually been particularly true about AI versus educational software; researchers trained in AI build AI systems while those trained in computer education build educational software. This is unfortunate. In the future, researchers interested in using computers to study cognition should be prepared to use whichever tool is best suited to the problem they are currently working on. If the goal is to develop a broad theory and to test it with a very large amount of domain knowledge, then the researcher should

consider the approach of building a piece of educational software to get a human (and his knowledge-base) into the loop. On the other hand, if the research goal is to do a somwhat narrower but deeper study of a particular component of cognition, and the ability to have the program be tested in many different domains is not as crucial, then a stand-alone AI system may be more appropriate.

The two possibilities we have implemented are unlikely to exhaust all the possibities. For instance, the case-adaptation coach we are currently building is a coach of a more knowledge-intensive type, placing it somewhere between the Sounding Board's simplicity, and ABE's complexity. Computers will contribute to cognitive science best if researchers consider the entire array of computational models that can be applied to the problem, so that they can choose the right tool for each job.

# Appendix: ABE's 21 Question-Asking Strategies

There is not enough room here to give the detailed kind of description of the question-asking algorithms presented in (Kass 90). This section contains only a very abbreviated description of each question-asking strategy, concentrating on what the questions are, and when they might be asked rather than on how exactly the system asks them of itself. Each of the twenty-one questions will be presented along with a description of the situations in which it applies, an example of how it works, and a very high-level description of the algorithm involved in asking it.

## Replace an action : Use action hierarchy

**Question:** A very common reason for invoking the tweaker is that an explanation makes a claim of the form, "agent *X* performed action *Y*," that the system does not believe. This first strategy addresses such situations by asking the question:

**What other actions are sort of like action Y?**

This raises a couple of important questions: What does "sort of like" mean in this context, and what's involved in finding a reasonable substitution in memory? Those are among the topics we will address in this tweak description. In addition, since this is the first tweak described, some of the issues relevant to all substitutions will be touched on here, using this specific strategy as an example of how those issues arise.

**Description:** Replace an action found in the original XP with an alternative action.

Search by finding an action that is closely related to the original action in the action hierarchy.

Check proposed replacements to be sure they are compatible with the original agent and are capable of being involved in some of the same causal relations as the original action.

**Applicable to:** This tweak is applicable to any incompatibility problem that involves an action, such as an agent-action mismatch, an object-action mismatch, or an implement-action mismatch.

**Example:** Applying the **JOPLIN XP** to the case of Swale's death generates an explanation which includes the implausible hypothesis that Swale took recreational drugs. Recreational drug-taking is represented in the system's knowledge base as both a kind of drug taking, and a kind of recreational activity. So, when recreational drug-taking shows up somewhere that it isn't appropriate, this strategy brings other kinds of drug-taking actions (such as the taking of medicinal, or performance-enhancing drugs) and other kinds of recreational activities (such as playing at sports, reading books, and going to the movies) into consideration. Some of these related actions can form the basis of plausible explanations in the context of a JOPLIN-based explanation of Swale's death.

## Replace an action : Use agent-theme links

**Question:** This strategy is an alternate method for dealing with a situation in which the system doesn't believe that agent *X* performed action *Y*. It asks a slightly different question:

**What other actions does *X* (or agents like *X*) typically perform?**

**Description:** Replace an action found in the original XP with an alternative action.

Search by finding an agent that is stereotypically associated with one of the categories that the agent of the original XP is in.

Check prospective replacements to be sure they are capable of causing some of the same effects as the original action.

**Applicable to:** The domain of applicability for this strategy is the same as for the previous strategy Any plausibility problem involving an action, such as an agent-action mismatch, implement-action mismatch, or object-action mismatch.

**Example:** Applying the **FIXX XP** to the Bias story yields an explanation which claims that Bias suffered a heart attack as a result of the exertion from recreational jogging. This explanation is not completely implausible, but Bias wasn't known as a recreational jogger. Even less appropriate would be to apply the **FIXX XP** to Swale's death, since racehorses are certainly not typical recreational joggers. But while recreational jogging is not stereotypically associated with Bias or Swale, it is pretty clear that some of the actions which are associated with each of these agents could serve just as well in the explanation. Playing basketball is associated with Bias, and running in races is associated with Swale, and each of those actions could have caused the same type of exertion that jogging caused in Fixx. This strategy brings actions that are stereotypically associated with a category the agent is in --- the way playing basketball is associated with being a basketball player, and running in races is associated with race horses --- into consideration.

## Replace action : Use stereotypical-cause links

**Question:** This strategy is a third method for discovering an action to replace one suggested in the original explanation in order to deal with a situation in which the system doesn't believe that a particular action in the explanation, *Y*, actually occurred. This strategy does so by selecting an event, *Z*, which the original explanation claimed was a result of *Y*, and asking the question:

**What other actions typically cause events like *Z*?**

**Description:** Replace an action found in the original XP with an alternative action.

Search by finding an action that is indexed as a typical cause of one of the events in the XP.

Check that the proposed replacement is compatible with the original agent and the original object.

**Applicable to:** Any plausibility failure involving an action, such as agent-action-mismatch and Implement-action-mismatch.

**Example:** Suppose that the **MAFIA REVENGE XP** is applied to explain a certain death. That XP predicts that *X* was killed by *Y* in retaliation for *X* having killed a mob-brother of *Y*. But suppose that the evaluator does not believe that *X* would have killed someone. The action of killing someone would have to be replaced by the tweaker. Since the relevant consequence of the hypothesis that *X* killed *Y*'s friend is that *Y* becomes very angry at *X*, employing this strategy in this context would mean searching for a replacement action by seeking out actions that stereotypically make people seek revenge. Actions indexed as having such consequences include cheating someone at business, having an affair with someone's spouse, and injuring someone's friend. Each of these would be brought into consideration by this strategy.

## Replace agent : Use goal-satisfaction links

**Question:** This strategy begins a second major sub-category of slot-replacing strategies. The first set all represented methods of replacing an action with another, equivalent action. The members of this second set, are applicable to many of the same explanation failures, but they deal with them differently. Rather than replace the action, they attempt to replace the agent. This strategy does so by asking the question:

**Who would have wanted to bring about one of the consequences of the action?**

**Description:** Replace the original agent of one of the actions in the XP.

Find another agent who is known to have one of the consequences of the action as a goal.

Check that the new agent is capable of performing the action.

**Applicable to:** Any agent-action mismatch or agent-object mismatch.

**Example:** Because the **JOPLIN XP** requires the victim to inject itself with drugs, it is not applicable to a horse, such as Swale. Even when drugs are replaced with performance-enhancing drugs to account for the fact that recreational drugs are not administered to horses, the system is left with an implausible explanation suggesting that Swale gave himself performance-enhancing drugs. The system would modify this variation of the **JOPLIN XP** again, however. Using this strategy, the system would search the knowledge base for another agent who would have been motivated to give Swale drugs. Since trainers are indexed as having the goal of making horses run faster, which is a direct consequence of administering performance-enhancing drugs, the system could develop the very plausible hypothesis that Swale's trainer might have given him drugs to make him run faster.

**Example:** The **SPOUSE INSURANCE XP** fails when applied to Swale because Swale didn't have a spouse. But if the system searches for another agent who could have been motivated to kill Swale in order to collect insurance money, it can develop the hypothesis that perhaps his owner might have acted out of such a motivation.

## Replace agent : Use stereotypical-agent links

**Question:** This strategy involves replacing the agent of one of the action descriptions within an XP. It does so by asking the following question:

**What types of agents typically perform this action?**

**Description:** Replace the agent of one of the actions found in the original XP.

Search for an agent who stereotypically performs that action.

Check that the agent could have performed the action in the current context and that he had some possible motivation to do so.

**Applicable to:** This strategy applies to any XP failure that involves an agent. These include mismatches between the agent and an action, and those between an agent and an object.

**Example:** Applying the **JOPLIN XP** to explain Swale's death generates a hypothesis that includes the claim that Swale administered drugs to himself. Taken literally, this is

implausible because (among other things) Swale, by virtue of being a horse, was physically incapable of administering drugs. Swale is not a suitable agent for the action, **M-ADMINISTER-RECREATIONAL-DRUGS**. However, veterinarians typically do administer drugs to horses, and while a variation of the **JOPLIN XP** in which a veterinarian gave Swale recreational drugs still has some problems, it is closer to an explanation that could be quite interesting. In particular, this strategy can be chained together with the strategy that replaces an action with an alternative that is a neighbor in the action hierarchy. In that case, it could replace the action of administering recreational drugs with the action of administering medicinal, or performance-enhancing drugs. The result would be a plausible explanation that closely matches the hypotheses put forth by human subjects when we asked them what they thought might have happened to Swale.

## Replace agent : Use delta-agency inferences

**Question:** A third question the system can ask in an effort to find an alternative agent it can substitute into an XP is as follows:

**Who could the agent that was originally proposed have caused to perform the action on his behalf?**

**Description:** Replace the original agent of one of the actions in the XP.

Find an agent who the original agent could have caused to perform the action in his stead.

Check that the agent could have performed the action in the current context.

**Applicable to:** This strategy addresses the specific sub-class of agent mismatches in which the agent who was originally proposed by the XP would have been motivated to perform the action, but would have been unable to do so.

This strategy has a slightly more narrow domain of applicability than the other strategies for replacing an agent. It only applies to those agent-related mismatches in which the agent, while unable to perform the action in question, was believed to have the goal of doing so. The other strategies for replacing agents we have discussed eliminate the original agent from the explanation altogether. With this strategy, however, the original agent plays an important role in the output explanation. Although no longer assumed to have performed the action in question, the original agent will be assumed to have persuaded the alternate to convince the new agent to perform the action. (Those other two strategies leave open the possibility that the original agent will play a role somewhere else in the XP, but they don't *require it*, as this strategy does.)

**Example:** After tweaking the **SPOUSE INSURANCE XP** a bit, the system might propose an explanation that suggests that Swale's owner killed him in order to collect the insurance money. This could be a very interesting, plausible explanation, unless the system knew that Swale's owner had an alibi. If Swale's owner was known to be elsewhere at the time that Swale was killed, then he clearly couldn't have performed the killing himself. In that case, this strategy could be invoked to come up with someone, such as one of the owner's employees who Swale's owner could have forced or persuaded to perform the killing.

## Replace implement : Use action-implement links

**Question:** This begins the third set of slot-replacement strategies --- those that replace the filler of the implement slot in an action description. All of the strategies in this

category address plausibility failures in which one of the tools which an XP suggests were used to perform some action is inappropriate in the current context. This strategy addresses such problems by asking the following question:

**What implements are commonly used to perform this action?**

**Description:** Replace an implement used in one of the actions in the original XP.

Search for one that is typically used to perform the action in question.

Check that the original agent is capable of using the implement and that the action, when performed by the agent with the new implement, could have caused some of the effects that the original explanation called for.

**Applicable to:** This strategy is applicable to any plausibility problem involving the implement used to carry out an action. Typically this is either an implement-inadequacy mismatch, where the action, if performed with the suggested implement couldn't have had the proposed effect, or an agent-implement mismatch, in which the proposed agent of some action would have been unable to use the proposed implement.

**Example:** Suppose that the system developed an explanation, based on what it believed about CIA plots against Fidel Castro, which claimed that the CIA attempted to kill Khomeni by planting an exploding cigar in his collection. The evaluator could object to this explanation on the basis that Khomeni, by virtue of being a devout Moslem, would not be expected to smoke cigars. The implement, an exploding cigar, would be inadequate for the task. This strategy would attempt to adapt the explanation by replacing the exploding cigar with a tool typically used to perform political assassinations. Using this method the system could hypothesize that they tried to kill him with a car bomb or with gunfire.

## Replace implement : Use agent-implement links

**Question:** This strategy attempts to find a replacement for an implement mentioned in the original XP by asking the straightforward question:

**What type of implements would this agent typically be expected to have available to him?**

**Description:** Replace an implement used in one of the actions in the original XP.

Search for an implement which is typically used by the agent of that action.

Check that the original agent is capable of using the implement and that the action, when performed by the agent, with the new implement could have caused some of the effects that the original explanation called for.

**Applicable to:** This strategy is applicable to the same set of failures as as the previous one --- any plausibility problem related to an implement mentioned in the XP.

**Example:** Suppose that the system finds out that someone was killed, and invokes the **TERRORIST BOMBING XP** to explain why someone was killed, but evidence turns up that the deceased did not die in an explosion. The implement proposed by the XP for carrying out the killing is a bomb, but that is not a possibility in the current context. If information is known about the suspected killer, then this may provide clues about plausible implements. For instance, if the suspected killer is a prize-winning marksman, then a rifle is a reasonable guess. If the killer is a pharmacist, then the system will consider the use of poison.

**Example:** Applying the **MAFIA REVENGE XP** to the Pan Am disaster yields and explanation that an enemy of the U.S. shot down the airliner with a gun in retaliation for a killing performed by an American. The gun is clearly not the appropriate weapon. The most obvious implement of destruction to replace the

gun with depends on who the hypothesized enemy is. If the enemy was a major military power, such as the Soviet Union, then an air-launched missile would be a likely weapon, since they have been used that weapon to destroy airliners before. On the other hand, if the hypothesis was that a terrorist organization committed the act, then an on-board bomb would be hypothesized since that is the weapon typically associated with such agents.

## Replace implement : Use location-implement links

**Question:** This strategy attempts to find a new implement that could have been used to perform an action by asking:

**What implements are typically available in the location where the action took place?**

**Description:** Replace an implement used in an action in the original XP.

Search for one that is typically found at the location where the action was performed.

Check that the new implement could have been used by the original agent to perform the original action, with at least some of the original causal consequences.

**Example:** Suppose that a stabbing takes place inside a hospital that has a metal-detector at the door, so that the killer could not have brought any weapon in. It is reasonable to guess that the killer used a scalpel to do the stabbing since scalpels are items found in hospitals.

## Elaborate motivation : Downplay negative

**Question:** This strategy remedies a vagueness problem involving an anomalous intentional action by asking the question,

**Why would the negative effects of the action be particularly unimportant to the agent involved?**

**Description:** Elaborate an explanation by explaining one of the decisions made by one of the agents in the original XP.

Augment the explanation with a hypothesis about why a bad effect of the decision would be less important to the agent than one would typically expect.

**Applicable to:** This strategy applies to the sub-class of vagueness failures which result when an explanation posits that an agent performed an action that he wouldn't normally be expected to perform, without providing a sufficient explanation of what might have motivated the agent to perform that action.

**Example:** This strategy searches for rules that indicate reasons why one of the bad side-effects of a problematic action might have been given less weight by the agent than the evaluator would have expected. In the suicide-bombing example, dying might have been less important to the suicide bomber if she were convinced that she were terminally ill and knew she would die soon anyway, or if she were depressed enough to have decided that it wasn't particularly important to go on living.

**Example:** If terrorists perform a killing at an airport the system may replace the standard tool for escaping from the scene of the crime --- via an automobile --- with an airplane. It may hypothesize that since the terrorists were at an airport, where airplanes are available, that they planned to escape via an airplane.

## Elaborate motivation : Amplify positive

**Question:** This strategy remedies problems related to unmotivated intentional actions by asking the question:

**Why would the positive effects of the action be particularly important to the agent involved?**

**Description:** Elaborate an explanation by explaining one of the decisions made by one of the agents in the original XP.

Augment the explanation by constructing a hypothesis about why a positive effect of the decision would be exceptionally important to the agent.

**Applicable to:** The same as for the previous strategy: Unmotivated intentional action problems.

**Example:** This strategy is like the last one in that it also looks for value-system adjustment rules that might be relevant to the story at hand, but this strategy looks for a different kind of rule. While the previous strategy looks for a rule indicating why a bad side could have been less important than normally expected, this one assumes that the negative side effects have their normal value and instead searches for a reason to believe that one of the positive effects might have been *more* important than expected.

For example, the **TERRORIST BOMBING XP** could be made suitable to the suicide bombing story if it were elaborated to include a sub-explanation that showed why the death of the Israeli soldiers was especially important to the bomber; so important that it outweighed the negative effect of the bomber's own death. For instance, since the system knows that the importance of killing an enemy will increase for an agent if the enemy kills a member of the agent's family or one of his close friends, and it knows that many Palestinians have had family and friends killed by Israeli soldiers, it could hypothesize that one of the soldiers might have killed a friend or family member of the bomber.

## Elaborate motivation : Identify knowledge gap

**Question:** The third method for dealing with anomalous actions is to ask the following question:

**Which of the negative effects of his decision might the agent not have known about, and why?**

**Description:** Elaborate an explanation by explaining one of the decisions made by one of the agents in the original XP.

Augment the explanation with a hypothesis about why the agent involved might not have expected one of the bad effects of the decision.

**Applicable to:** The same as for the last two strategies: Vagueness failures involving insufficient motivation for performing a hypothesized action.

**Example:** This strategy can be applied to suicide bombing example to give a different sort of answer than that given by the previous two strategies. Instead of looking for reasons why she would not have cared about dying, or reasons why she would have been especially anxious to kill the Israeli soldiers, this strategy would elaborate the **TERRORIST BOMBING XP** by identifying possible reasons that the bomber would have failed to predict one of the negative effects of the action and therefore wouldn't have brought its negative value into the equation at all. For instance, by using this strategy the system could hypothesize that perhaps the

bomber did not predict that she would die because she didn't realize that she would be inside the car at the moment of the explosion. Perhaps the people who convinced her to perform the bombing had not told her that the car would explode while she was inside.

## Elaborate motivation : Add positive effect

**Question:** The fourth method we present for remedying problems related to anomalous intentional actions does so by asking:

**Are there positive effects of the action that might have motivated the agent but that were not mentioned by the original explanation?**

**Description:** Elaborate an explanation by explaining one of the decisions made by one of the agents in the original XP.

Augment the explanation to include an extended projection of the positive implications of the action which may have motivated the agent.

**Applicable to:** This strategy, like the previous three, is applicable to vagueness failures involving insufficient motivation for a hypothesized decision.

**Example:** As an example we once again consider a way that the **TERRORIST BOMBING XP** can be elaborated to explain the suicide bombing story. The relevant question is: Is there something positive which the agent knew or believed would result from her action which the evaluator may have overlooked in its initial projection? For instance, if the agent held the belief that she would be rewarded after death if she died in this way, this additional benefit could shift the balance in favor of performing the action.

## Elaborate explanation : Explain premise

**Question:** Addresses a situation in which some premise *P* is unconvincing, by asking the following question:

**Do I know an explanation that could provide a sub-explanation relevant to this premise? If not, can I build the sub-explanation on the fly?**

**Description:** Elaborate an explanation by explaining a belief that was a premise in the original XP.

If possible, do so by retrieving another XP that explains the premise. Otherwise, build the explanation by backward chaining.

**Applicable to:** Whereas the previous four strategies were specific to explanations of intentional behavior, this strategy is a less constrained, domain independent strategy relevant to any vagueness problems involving unconvincing premises.

It is common for explanations to be free of contradiction, and produce reasonable causal chains, but to be unconvincing by virtue of the fact that one of the causal chains terminates with a premise that is not believable without further explanation. This type of problem is particularly common in the XPs that are output by other tweaking strategies; changing an XP to address one kind of problem often involves introducing new premises to the XP that may not stand on their own.

**Example:** When the **JOPLIN XP** is applied to Swale's death, the resulting explanation needs to be adapted to account for the fact that Swale couldn't have injected himself with drugs. Several substituters address problems like this by searching for an alternate agent who could have performed the action. In this example, let's

assume that a strategy is invoked that develops the hypothesis that Swale's owner gave him drugs in order to kill him. This explanation is free of contradictions in the way that the original was not, since humans are capable of administering drugs. But this explanation would not be particularly convincing in the absence of an explanation of the premise that Swale's owner wanted to kill Swale. This strategy can address situations such as this by building sub-explanations for arbitrary premises. In this example, the premise could be addressed by invoking other XPs that the system knows. For instance, a tweaked version of the **SPOUSE INSURANCE XP** could be appended to the tweaked version of the **JOPLIN XP** to produce a new explanation that concatenates them in an interesting way: perhaps racehorse owners sometimes kill them by administering an overdose of recreational drugs to them in order to collect one the insurance policies they have taken out on their horses.

Explanation patterns won't be available to address every sub-explanation that this strategy might ever be called upon to build. For instance, suppose that the **FIXX XP** is applied to a racehorse who comes from a family known to be free of hereditary heart defects. In that case, after a strategy is invoked to substitute running in races for recreational jogging, the system will be left with an explanation that is free from contradiction, but that simply assumes that the horse's heart was weak. If the evaluator called upon the adapter to support this premise with a sub-explanation, the adapter would not (given its current XP library) have any relevant XPs so it would have to resort to backward chaining to build an inference chain that would explain the premise.

## Elaborate explanation : Connect events

**Question:** Remedies an explanation that doesn't tie events *A* and *B* together by asking the question:

**Do I know an explanation that could explain how event *A* might have caused event *B*? If not, can I build the sub-explanation linking the two on the fly?**

**Description:** Elaborate an explanation by building a causal chain between two events of the events mentioned in the original XP. If possible, retrieve a secondary XP that can be used to explain the consequent in terms of the antecedent. Otherwise, employ bi-directional search, forward chaining from the antecedent event and backward chaining from the consequent, to build a chain between the two.

**Applicable to:** Vagueness failures related to insufficiently strong causal links.

**Example:** An XP based on the folk wisdom that too much sex leads to death will not convince an evaluator that is looking for a serious explanation because it contains no inference chain between sex and death. However, one could splice in a variation of the **FIXX XP** which relates the physical exertion involved with sex to death. This would result in an explanation that hypothesizes that too much sex leads to stress on the heart, which, combined with a weak heart, leads to a fatal heart attack.

## Elaborate explanation : Question prior belief

**Question:** Remedies a conflict between the knowledge base and the original XP by asking the question:

**Is it possible that one of the beliefs in the knowledge base that led to contradiction with the XP might be inaccurate for some reason?**

**Description:** Elaborate the explanation with a reason that a belief in the knowledge base should be revised.

Consider negating one of the beliefs that causes a contradiction with the claims of the XP.

**Applicable to:** Any plausibility failure.

**Example:** If the system attempted to apply the **FIXX XP** to a particular person, *X*, who died a day after having had a medical exam in which he was given a clean bill-of-health, the evaluator would be likely to object to the resulting explanation because of the following chain of reasoning:

- If someone with a heart problem of even minor severity has a medical exam, the heart problem will be detected during the exam.
- Heart problems take more than a day to develop from a point at which they are not detectable in an exam to the point where they are life threatening.
- *X* had a medical exam the day before he died and no heart problem was detected.
- Therefore, *X* didn't have a detectable heart problem the day before he died.
- Therefore *X* could not have had a critical heart problem on the day he died.
- The **FIXX XP** could not apply to *X* because it only applies to people who have severe heart problems.

If an XP suggests that a victim had a heart problem, this strategy would consider the hypothesis that perhaps the victim actually *did* have a heart problem, even if the knowledge base states otherwise. In order to support such a hypothesis, the system would attempt to augment the explanation to explain away the contradiction. The beliefs involved in the inference chain described above can be called contradiction-creating beliefs since the conclusion that there is a contradiction depends on each of those beliefs. The job of this strategy is to undermine the contradiction by negating one of the contradiction-creating beliefs. Perhaps heart problems can develop in a single day. Perhaps medical exams don't always detect heart problems. Or perhaps the victim didn't actually have a medical exam on the day before he died. Sometimes a creative system must, in service of entertaining an interesting new hypothesis, question the facts it has come to believe.

This strategy considers each of the contradiction-creating beliefs in search of one it can call into question. It augments its input explanation with additional explanatory structure that indicates why the original conclusion (in this case, the belief that *X* lacked a heart problem) was incorrect. The strategy attempts to locate the weakest link in the inference chain that originally led to that contradiction. It builds an explanation suggesting that the inference chain should be negated or at least qualified so that it no longer contradicts the explanation. For instance, if *X* were a competitive athlete, the system might support the hypothesis that *X* didn't really have a medical exam with the explanation that *X* didn't want to have his medical problems discovered for fear that he would be disqualified from competition. Because of this, he lied about having had an exam. Or the system might qualify the belief that medical exams detect heart problems with the proviso that this is true only if the doctor performing the exam is competent. It then adds to the XP the hypothesis that the doctor was not competent in this case.

## Refine agent description : Use goal filter

**Question:** Attempts to make a vague agent description more specific by asking the following:

**Which members of the original agent category would have desired one of the goal states achieved by the action in question?**

**Description:** Refine an agent description.

Search under the category originally specified for an agent who was known to have one of the consequences of the original action mentioned in the XP as a goal.

Check that the new, specific agent is compatible with the action, object, and implement involved in the original action, and that he would have had the goal of performing the action.

**Applicable to:** Vagueness failure that involve an agent description.

**Example:** When a generalized version of the **MAFIA REVENGE XP** is applied to explain the Pan Am 103 disaster, the resulting explanation (that terrorists destroyed the jet in order to extract revenge against the U.S.) is plausible but vague. It is vague with regard to which terrorists might have been involved.

This strategy addresses problems like this by searching for a more specific category that has the following two properties:

1 - The new category must be a sub-category of the category put forth by the original XP.

2 - The members of the sub-category must be known to have a goal that might motivate them to perform the action.

In this example, the original category is **ENEMY OF US**. The motivation proposed by the XP is to take revenge against the U.S. in retaliation for an unspecified wrong (of roughly the same severity as destroying a passenger airliner) that the U.S. perpetrated against this enemy. This strategy searches for sub-categories of **ENEMY-OF-US** (which include the categories **SOVIET-ALLIES** and **EXTREMIST-ARAB-COUNTRIES,** as well as the sub-categories of these categories, and so forth, until the system's representation bottoms out at specific individuals, governments, or other organizations). Each of these categories is searched for members who are known to have the stated motivation. Since one of the members of the extremist Arab nations is Iran, the system checks its knowledge base to see if Iran had a goal that would motivate it to destroy an American plan. As it turns out, Iran did profess to have the goal of destroying an American passenger jet in retaliation for the American shoot-down of one of its planes. Therefore, this strategy puts forward the hypothesis that Iranian terrorists performed this sabotage.

## Refine action description : Use implement filter

**Question:** Attempts to make vague action descriptions more specific by asking the question:

**Which members of the original action category typically involve an implement that is observable in the current situation or that is related to the agent in question?**

**Description:** Refine an action description.

Make the original, vague action description less vague by hypothesizing that the specific method used to perform the action was one that involves an implement that either is mentioned in the current story, or is associated with the agent who performed the action.

Check that the action is plausible in the current context.

**Applicable to:** Vagueness failures that involve an action description.

**Example:** Consider a situation in which a physician's wealthy wife dies unexpectedly. If an explanation-building system applied the **SPOUSE INSURANCE XP** to this situation, it would produce an explanation hypothesizing that the doctor killed his wife in order to collect life insurance money. Such an explanation contains much useful content but it is silent on the issue of *how* the killing was performed. There are many types of killing actions, such as stabbing, hanging, electrocuting, exploding, poisoning, and shooting. If a more specific description of the killing is required, the system would need to invoke a strategy such as this one to make a conjecture about which killing method is used.

What is a reasonable conjecture about the method of killing involved in a murder where the murder suspect is a doctor? The following is one plausible line of reasoning: doctors typically administer drugs; drugs can be used as the implement in a poisoning action; poisoning is a standard method of killing someone; therefore, it is reasonable to conjecture that poisoning was the method of killing used.

## Generalize constraint : Reconcile with slot-filler

**Question:** This tweak addresses a situation in which one of the constraints that the XP places on one of the slots doesn't match the slot-filler provided by the current situation. It does so by asking the question:

**Can the constraint be generalized to make it compatible with the current slot-filler while still maintaining the causal coherence of the explanation?**

**Description:** Generalize a constraint to make it compatible with the current story.

Check that the causal links in the explanation still make sense after the generalization.

**Applicable to:** Plausibility failures in which a slot-filler provided by the current context does not meet one of the constraints posted by the XP.

**Example:** If the system has a **SUICIDE BOMBING XP** that calls for a Moslem religious fanatic, and it tries to use that XP to explain a killing in which the killer is a Christian zealot, the explanation will fail to apply. The system can generalize the explanation by generalizing the constraint that requires a Moslem religious fanatic to one that allows for any religious fanatic.

## Generalize slot-filler : Reconcile with other slots

**Question:** Address a failure in which two slot-fillers, *A* and *B*, are incompatible with each other by asking the following question:

**What generalization of slot-filler *A* would be compatible with *B* while still maintaining the causal coherence of the explanation?**

**Description:** Generalize one of the slot-fillers within one of the XP's beliefs to make it compatible with the other slot-fillers in the belief.

Find the least general node that is compatible with the other slot-fillers in the belief.

**Applicable to:** Any slot-filler incompatibility problem.

**Example:** If the **FIXX XP** is applied to Swale it will fail because Swale does not fit the stereotype for recreational jogging, which is one of the actions that the resulting explanation will conjecture that he performed. However, if this action is generalized from **M-RECREATIONAL-JOGGING** to **M-PHYSICAL-EXERCISE**, the more general version of the explanation is still causally coherent, and makes sense when applied to Swale.

## Delete problematic belief

**Question:** Handles any plausibility within a belief by asking:

**Can the problematic belief be removed while still maintaining the causal coherence of the explanation?**

**Description:** Delete a problematic belief from the XP.

Delete the causal links that the belief participates in.

Delete any beliefs that are no longer connected to the anomaly once the first anomaly is deleted.

**Applicable to:** Any plausibility problem.

**Example:** The **JOPLIN XP** has a number of causal strands that come together to form an explanation of why rock stars die of drug overdoses. One strand involves stress leading to a desire for stress-reducing activities, of which drug-taking is one. Another strand involves a connection between being a rock star and having drug-using friends. This, in turn, has two implications: access to drugs, and peer pressure to take drugs. A final thread involves the wealth that comes from being a rock star, which makes it easy to afford drugs.

Although each strand in the **JOPLIN XP** makes the explanation stronger, that doesn't mean that each strand is strictly necessary. If the strand involving wealth were removed, so that the explanation could apply to rock performers who weren't rich, the explanation would still supply a useful explanation involving the stress of performing and the influence of drug-taking friends. Alternatively, one of the other strands could be removed instead, to produce an explanation that applied, say, to rock performers who were not rich, or star performers who were not rock artists. All that's really needed to build a variation of the XP that applies to these other cases is to prune away the parts of the explanation that don't apply and to check that what is left is still coherent.

# Acknowledgements

Many people have contributed to the design and implementation of the systems described in this paper. ABE is a descendent of the SWALE system, which was designed by David Leake, Chris Owens, and the author, under the direction of Roger Schank. Louise Pryor helped to implement some of ABE's adaptation strategies. Sounding Board has been worked on by a large group of people. Beth Beyer, Rob Campbell, John Regalis and the author designed the initial versions of the system, and developed the initial question bases, again under Schank's direction. Mike Engber and Pete Welter have made many important improvements to the system. Kerim Fidel has greatly extended the question base.

I would also like to thank Robin Burke, Yehiel Hayon, Tom Lauer, and Art Graesser for very helpful editorial comments.

# References


Burton, R.R., and Brown, J.S. (1982). Investigation of Computer Coaching for Informal Learning Activities. In Sleeman, D. and Brown, J.S. (Eds.), *Intelligent Tutoring Systems.* Academic Press: London.

Carbonell, J.G. (1986). Derivational Analogy: A Theory of Reconstructive Problem Solving and Expertise Acquisition. In Michalski, R.S., Carbonell, J.G., and Mitchell, T.M. (Eds.), *Machine Learning: An Artificial Intelligence Approach, Vol. II.* Morgan Kaufmann: Los Altos, CA.

Collins, G. (1977). Plan Creation: Using Strategies as Blueprints. Tech. Rept. 599. Yale University Department of Computer Science.

Cullingford, R. (1978). *Script Application: Computer Understanding of Newspaper Stories.* Ph.D. dissertation, Yale University. Technical Report 116

DeJong, G.F. (1977). Skimming newspaper stories by computer. Tech. Rept. 104. Yale University Department of Computer Science.

Goldstein, I.P. (1977). The Computer as Coach: an Athletic Paradigm for Intellectual Education. Tech. Rept. AI memo 389. MIT.

Hammond, K.J. (1986). *Case-based Planning: An Integrated Theory of Planning, Learning and Memory.* Ph.D. dissertation, Yale University. Technical Report 488

Hammond, K.J. (1989). *Proceedings of the 1989 Workshop on Case-Based Reasoning.* Morgan Kaufmann: San Mateo, CA.

Kass, A.M. (1990). *Developing Creative Hypotheses by Adapting Explanations.* Ph.D. dissertation, Yale University. Published as ILS Tech Report 6, Nothwestern University

Kolodner, J., Simpson, R., and Sycara, K. (1985). A Process Model of Case-Based Reasoning in Problem Solving. In Joshi, A. (Ed.), *Proceedings of the Ninth International Joint Conference on Artificial Intelligence.* IJCAI: Los Angeles, CA.

Koton, P. (1988). Reasoning about Evidence in Causal Explanations. In Kolodner, J. (Ed.), *Proceedings of a Workshop on Case-Based Reasoning.* Defense Advanced Research Projects Agency: Morgan Kaufmann, Inc.: Palo Alto.

Leake, D.B. (1990). *Evaluating Explanations.* Ph.D. dissertation, Yale University.

Minsky, M. (1975). A framework for representing knowledge. In Winston, P. (Ed.), *The Psychology of Computer Vision.* McGraw-Hill: New York.

Schank, R.C. and Abelson, R. (1977). *Scripts, Plans, Goals and Understanding.* Lawrence Erlbaum Associates: Hillsdale, New Jersey.

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