Adaptation-Based Explanation: Extending Script/Frame Theory To Handle Novel Input

Alex Kass*
Yale University Department of Computer Science
P.O. Box 2158 Yale Station, New Haven, CT 06520

Abstract

The ability to develop hypotheses to explain new, unexpected experiences is a hallmark of human intelligence. It is also a crucial concern within artificial intelligence, since intelligent computer systems need to construct explanations in order to guide learning, to recover from planning failures, and to make sense of the stories that they read.

The principal issue in designing a computer program that builds explanations is how to bring the system's causal knowledge to bear on a problem so that it can efficiently infer an unseen cause of observed events. In this paper we discuss some drawbacks of previous approaches to this problem, and present an alternative. The alternative involves extending script/frame theory via a system that adapts its stored explanations to new situations. We discuss the types of explanation failures that occur, and how the system employs adaptation strategies to repair those failures. The execution of one of the strategies is demonstrated.

1 Introduction

The ability to develop hypotheses to explain new, unexpected experiences is a hallmark of human intelligence. It is also a crucial concern within artificial intelligence, since intelligent computer systems need to construct explanations in order to guide learning (as discussed in [DeJong, 1983] and [Mitchell *et al.*, 1986]), to recover from planning failures (as discussed in [Hammond, 1986]), and to make sense of the stories that they read.

The hard part of processing interesting stories is not understanding the language as such, but developing creative hypotheses about what might be going on in the story. For example, consider the following story: "Swale was a star three-year-old racehorse. Swale won the Belmont Stakes. A few days later, he died." This story is an interesting one even though it is short and simple, because it gets a curious reader to start wondering what might have happened to Swale. The goal of the Adaptation-Based Explanation project (henceforth, ABE[1]) is to build a system that can develop creative hypotheses about events, like the one described in the Swale story, by adapting the explanations that it already has in memory to new situations.

2 Inference is the issue

The principal issue in designing a computer program that builds explanations is how to bring the system's causal knowledge to bear on a problem so that it can efficiently infer an unseen cause of observed events.

Approaches to this inference problem have traditionally fallen into two principal categories. The first approach was to build systems that had a large base of inference rules and were able to build explanations by chaining these inference rules together on the fly, in response to each new problem. Examples of this approach include Rieger's [1975] conceptual inferencer, MYCIN [Shortliffe, 1976] and Wilensky's [1978] PAM program. Researchers who take this approach are led to concentrate on methods of controlling the combinatorial explosion inherent in the inference-chaining process. Parallel methods of rule activation, bi-directional search, and application of either domain-dependent or abstract, thematic search-control knowledge are among the important elements in this camp's bag of tricks.

On the other hand, another school of research seeks to view the entire explanation process as an indexing problem. These researchers postulate the existence of ready-made knowledge schemas, such as scripts or frames ([Schank and Abelson, 1977], [Cullingford, 1978], [Minsky, 1975], [Charniak, 1977]) that will provide all of the proper inferences if only the right structure can be found in memory. This is the "retrieve and apply" school of explanation construction. For script appliers, the inference problem is reduced to matching input from the story against a schema in memory This makes script appliers very efficient, provided that the stories they are faced with match their schemas. The concerns of the "retrieve and apply" school naturally revolve around

*This work was supported in part by the Defense Advanced Research Projects Agency, monitored by the Office of Naval Research under contract N0014-85-K-0108 and by the Air Force Office of Scientific Research, under contracts F49620-88-C-0058 and AFOSR-89-0100.

[1] ABE is a descendant of SWALE, which was the joint work of the author, David Leake, and Chris Owens.

how these knowledge structures are represented and especially around how they are organized in memory so that the right one can be found at the right time.

Both sides have strong arguments against the other, and both are right. The script-application approach originally arose out of the realization that the "chaining + control" method was ignoring issues of memory. The systems it produced treated each experience as completely novel, and this limitation doomed those systems to perform expensive analyses every time they recognized a new instance of a plan. On the other hand, the script-application approach can only be employed when memory contains a script that *precisely* matches the episode being processed. In order for the matching process itself to be relatively inference-free, it is important that the expectations represented in a script be at a very specific level. But this means that a "retrieve and apply" system is stymied when confronted with situations that stray even slightly from its specific expectations. In short, "chaining + control" is a weak approach because it knows nothing of stereotypes, and "retrieve and apply" is limited because it knows *only* its stereotypes. What's needed is a system that knows stereotypes, but also knows how to go beyond them when necessary.

3 Adding adaptation to script/frame theory

ABE is an extension of script/frame theory for handling input that is less stereotypical. 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 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 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 re-use 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.) This is essentially an application of the case-based methodology to the problem of constructing explanations. (Other CBR-related adaptation work that has influenced my thinking includes [Hammond, 1986] and [Kolodner *et al.*, 1985]. Some other work on adapting explanations includes [Koton, 1988] and [Simmons, 1988].)

Because ABE draws on schemas as a starting point, but can then make sensible changes to those schemas, it requires both a schema library and a knowledge base of facts and plausible inference rules to represent its domain knowledge. The schema library contains the variablized explanations (called Explanation Patterns, or XPs [Schank, 1986]) that the system will adapt. The XPs are inference networks (built out of the raw material from the knowledge base) that lead from the anomalies they explain back to possible causes. XPs can be thought of as augmented scripts which represent not just the surface expectations to be matched, but also the causal relationships between the various expected events. This causal annotation aids in both the evaluation of instantiated explanations, and, as we shall see, in their adaptation.

The rest of this paper will concentrate on how the tweaker works. We will assume the existence of a retriever, evaluator, and storer, but won't describe how they operate. See [Owens, 1988] and [Leake, 1988] for some discussion of issues related to these tasks. We'll start by presenting some examples of the sorts of explanation failures that the tweaker can handle, next move to a discussion of some of the tweaking strategies used to fix those failures, and then move to more general discussion of the mechanics of tweaking.

4 Explanation failure

The tweaker is initially fed a "failed" explanation (*i.e.* one that the evaluator rejected as inappropriate in the current situation) along with a problem description (abbreviated hereafter as XP-FD, for XP-Failure Description), indicating why the evaluator decided that the XP needed tweaking. There are many different ways that explanations can fail, and there are several ways to fix most failures. A tweaking strategy is an algorithm for searching the knowledge base to find substitutions, generalizations, or specifications that are needed to fix a particular kind of failure. For example, if the problem with an explanation is that it involves some actor performing an action that he is incapable of performing, then the failure may be fixed by substituting in either a new action that the old actor could perform or a new actor who could perform the old action. Furthermore, there are several ways to search for prospective substitutions. What follows are a few examples of the main categories of tweaks that system handles.

4.1 Plausibility failures

Plausibility failures correspond to explanations that do not make sense because they contradict some aspects of the explainer's world-model, as represented in the knowledge base. Consider the following examples of plausibility failures:

- **The 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.

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.

Failure: Len Bias wasn't known as a recreational jogger. (An actor of some action in the explanation is not known to perform that action.)

Fixing the failure: By substituting playing basketball for recreational jogging, the adapter can build an excellent explanation based on this XP.

- **The Swale story:** Swale was a star three-year-old racehorse. The day after winning The Belmont Stakes, Swale died.

  Spouse insurance XP: Someone who is greedy and doesn't really love his/her spouse kills the spouse in order to collect the life insurance money.

  Failure: Swale has no spouse and no life insurance. (A slot referenced in the XP does not actually exist).

  Fixing the failure: By searching for other actors and similar motivations, the adapter can build from this explanation the reasonable hypothesis that Swale's owner got greedy and killed him in order to collect the property insurance.

Plausibility failures can arise from explanation that involve mismatched actor-action pairs, mismatched implement-action pairs, mismatched object-action pairs, reference by the explanation to missing slots, temporal sequence problems, and specific contradictions in between beliefs in XP and beliefs in the system's knowledge base.

4.2 Vagueness failures

Vagueness failures correspond to explanations that are not detailed enough, not sufficiently convincing, or do not contain the kind of information that suits the explainer's needs. What counts as sufficiently convincing or sufficiently detailed is a function of what the explanation will be used for.

For example:

- **The Pan Am Story:** Pan Am flight 103, en route from London to NYC, exploded in mid air, killing all aboard.

  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.

  Failure: 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. (A slot-filler is insufficiently specified.)

  Fixing the failure: 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.

- **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, killing herself and a number of Israeli soldiers.

  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.

  Failure: This doesn't explain an important part of the anomaly — why someone would do something that resulted in her own death. (A decision that the explanation claims occurred is not sufficiently motivated by the XP.)

  Fixing the failure: One way to make an action in an XP seem more motivated is to add an explanation of why the negative side effects of the action would be less important than expected to the actor involved. By employing this strategy the adapter can hypothesize that in addition to 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.

Vagueness failures mainly arise when the XP contains insufficiently specified slot-fillers, or beliefs that, while not contradicted by anything in the knowledge base, require more causal support.

5 Adaptation Strategies

In the previous section the idea was to describe the kinds of problems that the explanation tweaker is faced with, and to give a feel for the kinds of output that it produces. The heart of the adaptation process is a library of tweaking strategies, each of which is a program that is capable of fixing a class of explanation failures. For any given explanation failure that the tweaker can handle there will be a set of tweaking strategies that apply. The tweaker ranks this set according to how efficient the strategy has been in the past, how often it has worked in the past, and how closely tailored it is to the failure at hand. It then runs the strategies in best-first order until one of them produces an acceptable explanation. Now let's examine some of the strategies.

5.1 Substituters

Substituters attempt to fix plausibility problems by replacing a slot filler in one of the beliefs of a failed XP with a component that could have the same causal consequents. For example, one of the strategies available for fixing actor-action mismatches like the one that occurred with the Bias story above is:

T1: Replace old action with an action stereotypically associated with the actor.

This is the rule invoked to change from recreational jogging to playing basketball in the Bias example above. The idea is to climb the actor generalization hierarchy

to find the categories that the actor is in (such as basketball player and college student for Bias), and then to follow actor-theme links to find the actions associated with those categories. (for example, college students attend classes; basketball players play basketball). These are the candidates for substitution.

Next, the actions that are candidates must be examined to see if they link up with the original XP; are there inference rules that indicate that the new action could have caused the same things that the old action caused? Playing basketball involves running, just as recreational jogging does, so it is a good substitution to actually try. Attending classes doesn't involve running, or physical exertion, or anything in the original chain leading to death, and therefore is not a good substitution to try. The causal links within the original XP are crucial to making this distinction.

Briefer descriptions of some other substitution tweaks follow:

T2: Replace old action with an action closely related to it in the action hierarchy.

For example, recreational drugs is a kind of drug taking, and a kind of recreational activity. So, if recreational drugs shows up somewhere that it isn't appropriate, consider other kinds of drug taking (such as medicinal, or performance-enhancing) and other kinds of recreational activities.

T3: Replace old action with an action indexed as causing one of the events in the XP.

For example, if recreational jogging couldn't have been involved in the heart attack, consider other stereotypical causes of heart attacks, such as a sudden scare, or a bad diet.

T4: Replace old actor with an actor known to have motivation mentioned in XP.

For example, Swale didn't have a spouse, but his owner might have stood to collect money upon his death.

T5: Replace old actor with an actor who old actor could have made perform old action.

For example, if the explanation says that Swale's owner killed him, but the owner was known to be somewhere else at the time, maybe it was one of the owner's employees.

5.2 Generalizers

Generalizers fix plausibility failures and XP application failures by producing a version of an explanation that applies to a broader class of situations, at the cost of removing some of the detail from the hypothesis. There are two ways that this can be done:

5.2.1 Component generalizers

Component generalizers work by altering a particular slot filler in one of the beliefs of the XP. In this sense they are similar to substituters, except they try to generalize the invalid component rather than search for an alternative. Examples:

T6: Generalize old action to make it compatible with new actor.

For example, go from recreational jogging to physical exercise.

T7: Generalize constraint on actor to make it compatible with current actor.

For example, change an XP that calls for a Moslem religious fanatic to simply call for any religious fanatic to handle an example where the actor was known to belong to another religion.

5.2.2 XP simplifiers

XP Simplifiers generalize the structure of the XP by make a more global change than those which simply alter one of the components.

T8: Delete problematic belief

Removes conflicting beliefs from the explanation. The resulting explanation thus contains less information, but may still be useful in many contexts.

5.3 Specifiers

Specifiers fix vagueness problems by finding detail to add to an explanation, making it less general, but richer in information content. There are two sub-categories of specifiers, roughly analogous to the sub-categories of generalizers:

5.3.1 Component specifiers

Component specifiers add detail to the explanation by making the description of some slot-filler more specific. Examples:

T9: Specify actor description by finding an actor which matches that description, and that is also mentioned elsewhere in the XP or elsewhere in the story being processed.

For example, in applying the Spouse Insurance XP to Len Bias, the idea that the Celtics killed Bias for the insurance money represents an off-the-wall, but rather creative hypothesis that can be generated by this tweak.

T10: Specify actor description by finding an actor matching description that has motivation mentioned in the XP.

For example, Iran declared an intention to retaliate for the downing of one of its airliners, so it is known to be motivated to destroy American airliners, such as Pan Am 103.

5.3.2 XP elaborators

XP elaborators add detail to an explanation by building more structure into it. These strategies do things like add causal links between beliefs in the explanation, or add additional explanation to support one of the beliefs, or splice two XPs together to form a more complete explanation. Elaborators can involve recursive calls to the explainer. Examples:

T11: Add to the causal connections between two beliefs by splicing in another XP that explains how one would cause the other.

T12: Explain a decision made by one of the actors by splicing in an explanation of why the good effects of the decision would be more important to the actor than to most people.

For example, killing the Israeli soldiers might have become extremely important to the suicide bomber if they had killed someone in her family.

T13: Explain a decision made by one of the actors by splicing in an explanation of why the bad effects of the decision would be less important to the actor than to most people.

For example, dying might be less important to the suicide bomber if she was convinced that she would be rewarded after death, or if she were terminally ill and knew she would die soon anyway.

T14: Explain a decision made by one of the actors by splicing in an explanation of why the actor might not have known about a bad effect of the decision.

For example, perhaps the people who convinced her to do this had not told her that the car would explode while she was inside.

6 Overview of the tweaking process

Tweaking an explanation involves two steps, selecting a tweaking strategy to try, and then executing that strategy. Both of these tasks, **tweak selection** and **tweak execution** have a number of sub-steps. The overall idea is to retrieve a set of tweaking strategies suggested by the XP-Failure Description that was generated by the explanation evaluator, to make an estimate of which of the retrieved tweaking strategies is the most promising in the current situation, and then to figure out how to apply that most promising tweaking strategy to the XP in question, using the knowledge in memory to make the kind of changes to the XP that the strategy calls for.

6.1 Tweak selection

The first job a tweaker has when it is called upon to fix a failed explanation is to decide which **XP Tweaking Strategy** to apply. There are three sub-stages to the tweak selection problem: **tweak retrieval**, **tweak filtering**, and **tweak ranking**.

Tweak retrieval: Tweak retrieval involves pulling a first pass set of tweaking strategies out of the system's library of tweaking strategies. The tweak library of is organized in a straightforward fashion — each strategy is indexed under a variablized XP-FD *pattern* (abbreviated TIP, for Tweak Index Pattern) that corresponds to the type of failures that the strategy addresses. A TIP is a variablized structures that indexes a tweak in the tweak library. TIPs are constructed by the programmer and are static. An XP-FD is a structure that describes a specific explanation failure. XP-FDs are created dynamically by the evaluator. Each TIP matches a class of XP-FDs. The tweak retrieval step simply involves collecting all the strategies in the library whose TIP matches the XP-FD being addressed. This search process is currently implemented using a variablized discrimination net with the patterns serving as indices and the current failure as the key.

Tweak filtering: Not all the strategies that match a given XP-FD will be applicable to the XP at hand. For example, **T4** (described above), which involves substituting another actor mentioned in the XP could not be used to make the FIXX XP applicable to Bias, since the XP mentions no other actors. Each tweak has a set of constraints associated with which get run on the current XP to determine if it makes any sense to try that tweak on that XP. If and of the checks fail, the tweak is filtered out from the list of candidates.

Tweak ranking: Not all the strategies which don't get filtered out are equally likely to produce desirable results. Therefore, tweaks are tried in best-first order, with ranking based on the following three criteria:

- **Match Specificity** is a measure of how closely tailored the tweak is to the XP failure at hand. For example, **T8** can apply to just about any plausibility problem by removing the implausible belief altogether. This very general strategy often leaves the explanation much weaker, however. since it reduces the support for any conclusions that previously depended on the deleted belief. Closely-tailored strategies are preferred over more general ones. In the current implementation this calculation is done by a simple method — counting the number of constants that match between the XP-FD and the tweak's TIP. A strategy with a more detailed TIP will thus match in fewer situations but will be more highly ranked when it does match.

- **Cost Estimate** is a measure of the amount of time that a tweak is likely to consume. For example, a tweak that simply deletes an offending belief is very fast, while a belief that searches memory for an appropriate generalization is medium expensive, and a tweak that builds an entire sub-explanation to support a premise in an XP can be very expensive. In the current implementation, the "cost" of a tweak is estimated based on a very simple-to-calculate number — the average number of inferences that this tweak has performed when run in the past.

- **Hit Ratio** is a measure of how often a given tweak has produced explanations that the evaluator has found usable in the past. When the tweak produces an explanation that the evaluator rejects its hit ratio goes down, when it produces an explanation the evaluator decides to use, the hit ratio goes up.

Explanations are needed for different reasons at different times, and this affects which tweaking strategy is most appropriate in a particular situation. A system in which ABE is embedded can control the tweaker's behavior by determining the weight that each of the above factors should be given relative to one another. For example, in situations where getting *some* hypothesis quickly is more important than necessarily getting the most detailed or most plausible explanation possible, the cost estimate could be weighted more highly than the other two factors.

6.2 Tweak execution

A crucial issue that a theory of adaptation must answer is what level of knowledge belongs in the adaptation strategies. One school of thought is to make the level of knowledge very general, sticking to rules like: *When an explanation provides an inappropriate slot-filler, replace it with another slot-filler.* This approach is elegant in that it produces a small number of domain-independent strategies, but it doesn't promise much in the way of efficiency because the strategies don't specify how to search for the appropriate replacement. On the other hand, one could adopt very domain-specific adaptation rules that say things like: *When looking for someone who killed a racehorse, consider the horse's owner.* Such rules provide excellent guidance to the search process once selected, but there will be very many of them, and one is left without much of a general theory of tweaking. ABE's adaptation rules fall somewhere in between. They are more problem-specific than the general rules described above, but they are still quite domain-independent. The knowledge that is coded into them is not domain knowledge — that is coded into the knowledge base. Instead, what tweaking strategies know is how to search that knowledge base in order to make the necessary modifications.

A tweak is defined by three things: the part of the XP that it modifies, the method of searching for the domain knowledge necessary to do the modification, and a set of checks that have to be run on the modifications to see if they make any sense. For example, **T1** and **T2** are candidates for fixing the same sets of explanation failures, and they both work by replacing the actor in one of the XP's beliefs. However, they differ in the way that they search for the replacement (**T1** uses the actor-generalization hierarchy and actor-theme links, while **T2** uses the action-generalization hierarchy). Since they use different algorithms to search for substitution candidates, the set of checks they must perform is also different. For example, **T2** must check to see that any actions it considers are compatible with the original actor, while **T1** does not since that is guaranteed by its search algorithm.

7 A tweak in action

Several, though by no means all, of the tweaking strategies discussed above are implemented in the ABE computer program. The program output that follows is intended to demonstrate how one of the system's tweaks actually operates. The transcript has been heavily edited to meet space requirements, and concentrates on showing the tweak execution phase. The system is attempting to fix a problem with the application of the FIXX-XP to Swale's death (Swale was not a recreational jogger). It searches for actions stereotypically associated with Swale. It finds three. One (competing in horse races) is the antecedent of an inference rule involving the running action mentioned in the original explanation. This rule forms the basis of a new explanation in which competing in races substitutes for recreational jogging.

```
Running a tweak:
  Substitute an action  that is stereotypically
   associated with the actor.

  Anomaly to be explained: SWALE died.
  XP being tweaked:     FIXX-XP

 Problems is with SWALE as the actor of
                  DID RECREATIONAL JOGGING
    Problem type -
     Actor-action-mismatch : physical-disability

 3 actions associated with SWALE being considered:

  SWALE COMPETED IN HORSE RACES BECAUSE
   SWALE is in category RACEHORSES AND
   RACEHORSES had theme COMPETED IN HORSE RACES.

 Trying to link up to: ("SWALE ran")
 This chain DOES link up with the rest of the xp:
  Inference: ?X? COMPETED IN HORSE RACES -> ?X? ran
 ---
  SWALE TOOK PERFORMANCE DRUGS BECAUSE
   SWALE is in category RACEHORSES AND
   RACEHORSES had theme TOOK PERFORMANCE DRUGS.

  RACEHORSES had theme TOOK PERFORMANCE DRUGS
   BECAUSE
   RACEHORSES is in category  ATHLETES AND
    ATHLETES had theme TOOK PERFORMANCE DRUGS.

 Trying to link up to: ("SWALE ran")
 Does NOT link up with the rest of the XP.
 Cannot create a tweaked explanation based on
  SWALE TOOK PERFORMANCE DRUGS
 ---
  SWALE ATE OATS BECAUSE
   SWALE is in category RACEHORSES AND
   RACEHORSES had theme ATE OATS

  RACEHORSES had theme ATE OATS BECAUSE
   RACEHORSES is in category  HORSE AND
   HORSES has theme ATE OATS

 Trying to link up to: ("SWALE ran")
 Does NOT link up with the rest of the XP.
 Cannot create a tweaked explanation based on
  SWALE ATE OATS
 ---
Creating a new tweaked explanation base on:
  SWALE COMPETED IN HORSE RACES

 Adding an index to FIXX-XP-1:
    ?X? COMPETED IN HORSE RACES

 Adding: ?X? ran BECAUSE
   ?X? COMPETED IN HORSE RACES.
 Adding: ?X? COMPETED IN HORSE RACES BECAUSE
   ?X? had theme COMPETED IN HORSE RACES.
 Adding: ?X? had theme COMPETED IN HORSE RACES
   BECAUSE
   ?X? is in category RACEHORSES AND
   RACEHORSES had theme COMPETED IN HORSE RACES.
 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.

FIXX-XP-1.EXPL-1
  Premise: THE HEART OF SWALE is in category
           HEREDITARY DEFECTIVE HEARTS
  Premise: SWALE is in category RACEHORSES
  SWALE had theme COMPETED IN HORSE RACES BECAUSE
   SWALE is in category RACEHORSES AND
   RACEHORSES had theme COMPETED IN HORSE RACES.
  THE HEART OF SWALE was weak BECAUSE
   THE HEART OF SWALE is in category
    HEREDITARY DEFECTIVE HEARTS.
  SWALE COMPETED IN HORSE RACES BECAUSE
   SWALE had theme COMPETED IN HORSE RACES.
  SWALE ran BECAUSE
   SWALE COMPETED IN HORSE RACES.
  SWALE had a VERY HIGH degree of
   PHYSICAL CONDITIONING BECAUSE
   SWALE had a VERY HIGH degree of EXERTION.
  SWALE had a VERY HIGH degree of EXERTION BECAUSE
   SWALE ran.
  SWALE had a heart-attack BECAUSE
   THE HEART OF SWALE was weak AND
   SWALE had a VERY HIGH degree of EXERTION.
  SWALE died BECAUSE
   SWALE had a heart-attack.
```

8 Conclusion

The goal of this paper has been to describe and to advocate an adaptation-based theory of explanation. I am attempting to extend script/frame theory by using adaptable explanation patterns as my knowledge schemas, and by providing my system with the means of adapting its stored schemas to new situations. I have developed a taxonomy of explanation failures and an arsenal of failure-specific, but domain-independent adaptation strategies.

ABE learns new variations of its schemas as it processes new stories. It attains the efficiency advantages of a script-applier and the flexibility of a system that constructs explanations from scratch.

Acknowledgments

I would like to thank Chris Owens, Louise Pryor, and the IJCAI referees for their very helpful comments on drafts of this paper.

References

[Charniak, 1977] E. Charniak. Ms. malaprop: A language comprehension program. In *Proceedings of the Fifth International Joint Conference on Artificial Intelligence*, Cambridge, Mass., August 1977. IJCAI.

[Cullingford, 1978] R. Cullingford. *Script Application: Computer Understanding of Newspaper Stories.* PhD thesis, Yale University, 1978. Technical Report 116.

[DeJong, 1983] G. DeJong. An approach to learning from observation. In *Proceedings of the International Machine Learning Workshop*, pages 171–176, Monticello, IL, June 1983. University of Illinois.

[Hammond, 1986] K.J. Hammond. *Case-based Planning: An Integrated Theory of Planning, Learning and Memory.* PhD thesis, Yale University, 1986. Technical Report 488.

[Kolodner *et al.*, 1985] J. Kolodner, R. Simpson, and K. Sycara. A process model of case-based reasoning in problem solving. In A. Joshi, editor, *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*, pages 284–290, Los Angeles, CA, August 1985. IJCAI.

[Koton, 1988] P. Koton. Reasoning about evidence in causal explanations. In J. Kolodner, editor, *Proceedings of a Workshop on Case-Based Reasoning*, pages 260–270, Palo Alto, 1988. Defense Advanced Research Projects Agency, Morgan Kaufmann, Inc.

[Leake, 1988] D. B. Leake. Using explainer needs to judge operationality. In *Proceedings of the 1988 AAAI Spring Symposium on Explanation-based Learning.* AAAI, 1988.

[Minsky, 1975] M. Minsky. A framework for representing knowledge. In P. Winston, editor, *The Psychology of Computer Vision*, chapter 6, pages 211–277. McGraw-Hill, New York, 1975.

[Mitchell *et al.*, 1986] T.M. Mitchell, R.M. Keller, and S.T. Kedar-Cabelli. Explanation–based generalization: A unifying view. *Machine Learning*, 1(1):47–80, 1986.

[Owens, 1988] C. Owens. Domain-independent prototype cases for planning. In J. Kolodner, editor, *Proceedings of a Workshop on Case-Based Reasoning*, pages 302–311, Palo Alto, 1988. Defense Advanced Research Projects Agency, Morgan Kaufmann, Inc.

[Rieger, 1975] C. Rieger. Conceptual memory and inference. In *Conceptual Information Processing.* North-Holland, Amsterdam, 1975.

[Schank and Abelson, 1977] R.C. Schank and R. Abelson. *Scripts, Plans, Goals and Understanding.* Lawrence Erlbaum Associates, Hillsdale, New Jersey, 1977.

[Schank, 1986] R.C. Schank. *Explanation Patterns: Understanding Mechanically and Creatively.* Lawrence Erlbaum Associates, Hillsdale, NJ, 1986.

[Shortliffe, 1976] E.H. Shortliffe. *Computer-based medical consultations: MYCIN.* American Elsevier, New York, 1976.

[Simmons, 1988] Reid G. Simmons. A theory of debugging plans and interpretations. In *Proceedings of the Seventh Annual National Conference on Artificial Intelligence*, pages 94–99, Palo Alto, 1988. American Association for Artificial Intelligence, Morgan Kaufmann, Inc.

[Wilensky, 1978] R. Wilensky. *Understanding Goal-Based Stories.* PhD thesis, Yale University, 1978. Technical Report 140.