Creativity and Learning in a Case-Based Explainer

Roger C. Schank and David B. Leake
Artificial Intelligence Project, Yale University, P.O. Box 2158, Yale Station, New Haven, CT 06520, U.S.A.

ABSTRACT

Explanation-based learning (EBL) is a very powerful method for category formation. Since EBL algorithms depend on having good explanations, it is crucial to have effective ways to build explanations, especially in complex real-world situations where complete causal information is not available.

When people encounter new situations, they often explain them by remembering old explanations, and adapting them to fit. We believe that this case-based approach to explanation holds promise for use in AI systems, both for routine explanation and to creatively explain situations quite unlike what the system has encountered before.

Building new explanations from old ones relies on having explanations available in memory. We describe explanation patterns (XPs), knowledge structures that package the reasoning underlying explanations. Using the SWALE system as a base, we discuss the retrieval and modification process, and the criteria used when deciding which explanation to accept. We also discuss issues in learning XPs: what generalization strategies are appropriate for real-world explanations, and which indexing strategies are appropriate for XPs. SWALE's explanations allow it to understand nonstandard stories, and the XPs it learns increase its efficiency in dealing with similar anomalies in the future.

1. Introduction

We cannot hope to anticipate every circumstance that a program might encounter. Even if we could, and we built in all the appropriate responses, the changing world would soon make that frozen knowledge base obsolete. Thus learning is essential in programs that deal with the real world.

Explanation-based learning (EBL) offers considerable advantages over traditional inductive methods, and has been the subject of much research (for a discussion of the explanation-based approach, see Mitchell et al. [21] or DeJong and Mooney [4]). But although EBL systems depend on having good explanations available, the problem of building the explanations to use has received surprisingly little attention. Most EBL programs construct explanations from scratch, chaining together basic causal rules. However, this approach is extremely inefficient for explaining complicated situations where

Artificial Intelligence **40** (1989) 353–385
0004-3702/89/$3.50 © 1989, Elsevier Science Publishers B.V. (North-Holland)

many rules might be relevant. Some responses to this problem have been suggested (for example, using inductive generalizations to focus on the features to consider (Lebowitz [19]), or allowing a program to accept externally provided hints that can help it build explanations otherwise beyond its capabilities [4]). But for situations where these aids are not available, an efficient way to construct explanations is still vital.

When people deal with new situations, they seldom reason from first principles. Instead, they start from their memories of experiences in similar situations. Those memories can then be adapted to suggest strategies for dealing with the situation at hand. The case-based approach to planning and problem solving has been advocated by a number of researchers (for example, Schank [25], Carbonell [2], Hammond [10], Kolodner, Simpson and Sycara [16]). We believe that a case-based approach is also a promising way to address the problem of building explanations. Just as people use experience to do planning and problem solving, they often use remindings of old experiences as the starting point for explanation.

Although remindings of old explanations are sometimes applicable in a straightforward way, they may also be hard to apply. When a person is reminded of a seemingly inapplicable episode, and someone else shows how it suggests a good explanation, we may be impressed by his creativity. For programs to do the learning needed to keep abreast of a complex and changing world, they must have the capability for this kind of creativity: programs will need to revise old explanations to explain events that are not very similar to those they've encountered before, even when the explanations they try to apply don't seem to fit. To profit from these creative explanations in the future, systems will also need to remember the explanations they generate, and to index them so that they will be accessible for straightforward or creative application in the future.

The following sections describe an approach to case-based explanation which was developed by the authors, Chris Riesbeck, Alex Kass and Chris Owens during work on the SWALE project (Schank [27], Leake and Owens [17], Schank and Leake [30], Kass [12]). SWALE is a story understander that detects anomalies in the stories it reads, explains them by retrieving and revising old explanations, and installs the new explanations in memory for future use. While work on SWALE is still in progress, the system has already developed a range of novel explanations. Because revision of old cases to fit new situations is creative, case-based explanation offers not only an efficient way to build explanations, but also a framework for investigating creativity as an algorithmic process.

2. Overview

Case-based explanation considers explanation as a memory process. We are strongly influenced by previous Yale work on memory organization and the

role of failure-driven reminding in learning, and we begin with a sketch of some of that work.

We then consider the representation of explanations in memory. The need for a knowledge structure flexible enough to support the adaptation process prompts us to define *explanation patterns* (XPs) [27], which capture the network of beliefs that underlie explanations. Given this structure, an explainer can directly apply the parts of the old explanation that fit the new case, and can identify and repair the parts of the explanation that do not apply. In this way, many of the results of the original explanation effort can be reused.

We then describe an algorithm for case-based explanation. Our description is in context of understanding stories with novel elements, and learning new XPs by explaining them. We show that our approach generates a range of interesting explanations.

Taking our model as starting point, we argue that creativity is primarily a matter of search and adaptation, and we discuss how our process could be extended to build a more creative system. Two appendices of concrete information follow: a description of the types of beliefs and inference rules represented in the SWALE system, and annotated output from a SWALE run.

3. Previous Work on Learning at the Yale AI Lab

Our work on case-based explanation has arisen from a progression of work on memory organization, the integration of events into memory, and the way that information in memory is indexed for retrieval. This perspective strongly influences our view of explanation as a memory-based process.

3.1. Learning and dynamic memory

Establishing the coherence of a story is vital to understanding it. But blindly chaining together inferences to build connections can result in a combinatorial explosion of possibilities (Rieger [22]). One way that people avoid this is by using their knowledge of standard event sequences to guide understanding. Our initial work on the knowledge structures in memory introduced *scripts* (Schank and Abelson [28]). A script is a stereotyped sequence of events within a particular context, such as the standard events involved in having a meal at a restaurant (wait for a table, sit down, order, etc.). Scripts facilitate routine understanding: in familiar situations, people rely on scripts to direct inference of likely events.

People must be able to refine knowledge structures as the world changes, but our formulation of scripts presented an impediment to some types of learning. Each script is completely independent of the others; distinct scripts share no structure. Consequently, learning done in the context of one script has no effect on the others. This conflicts with what we know about how people learn: if someone tries unsuccessfully to use a credit card to pay for a doctor's visit,

he'll anticipate that a dentist would refuse it also, even though the scripts for doctor visits and dentist visits are different.

Dynamic memory theory (Schank [24]) responds to the need for knowledge structures that allow cross-contextual learning. It postulates knowledge structures called *memory organization packets* (MOPs). Like scripts, MOPs characterize event sequences; unlike scripts, they allow a sequence to be composed of a number of lower-level components which can be shared by a number of MOPs. The basic components, called *scenes*, describe events which take place in a single location, with a single purpose, and in a single time interval. For example, the act of paying for an office visit to a professional (such as a doctor or dentist) is circumscribed in this way. This PAY scene tends to be standard regardless of the profession involved. Although M-doctor-visit and M-dentist-visit are distinct MOPs, they share a single PAY scene. Because the scene is shared, the learning that people do in our credit card example would also occur in a computer program with a MOP-based memory.

Dynamic memory theory led to two computer experiments. CYRUS (Kolodner [15]) models the organization of episodic memory, integrating new episodes into its previous knowledge, and using a range of retrieval strategies to generate indices for retrieving information. IPP (Lebowitz [18]) uses an inductive approach to learn new MOPs on the basis of newspaper stories.

3.2. Learning new MOPs

IPP read newspaper stories about terrorist attacks, using MOPs to provide expectations during the story understanding process. It made inductive generalizations about what it read, and installed them in memory as new MOPs that could guide understanding of later stories. For example, from reading newspaper stories about Italy when the Red Brigades were active, IPP formed the generalization that the usual victims of kidnapping in Italy were businessmen.

3.2.1. *Problems with the inductive approach*

Inductive learning has been used in many AI learning systems (for an overview of some of the methods used, see Dietterich and Michalski [5] or Mitchell [20]). However, because inductive learners have no criterion for importance of features except for their frequency of occurrence, they often make faulty generalizations when they deal with limited sets of data. Some of the conclusions they draw are very unlikely to be considered by people: for example, when IPP read about two bombings in India, each of which resulted in two deaths, IPP generalized that bombings in India always kill two people.

One response to problems dealing with limited data is to restrict a program to considering features that have been preselected as being important. However, this evades the important question of selecting from the many features that are presented by any real-world tasks. In addition, preselecting features

would only work in circumscribed domains, since which features are important varies enormously from situation to situation. If a car breaks down, the time of the breakdown is probably insignificant. But the time of a 2:00 AM car accident might be very important, if 2:00 AM is the time when bars close. What determines relevant features is the causal structure of the situation.

Explanation-based generalization allows important features to be selected dynamically, and allows learning systems, like people, to generalize on the basis of single episodes. A discussion of problems of inductive category formation and the need for explanation-based methods appears in [29], and also in EBL work such as [21].

3.3. Failure-driven learning

While the construction of new generalizations is important, it is also important to be able to recognize and learn from situations where old generalizations fail to apply. IPP built new knowledge structures on the basis of generalizations, and used them to organize episodes in memory. Thus IPP could recognize a story as commonplace, and could also notice novel aspects of a story. For example, consider the following story from the *New York Times*:

> An Arabic speaking gunman shot his way into the Iraqi Embassy here [Paris] yesterday morning, held hostages through most of the day before surrendering to French policemen, and then was shot by Iraqi security officials as he was led away by French officers.

The beginning of the story is a fairly routine incident of terrorism, but it is startling that the Iraqi officials shot the gunman after he'd surrendered. IPP could recognize from experience that the shooting was exceptional. However, it had no way to account for the deviation from expectations.

When people encounter difficulties, they are often reminded of past problems; thinking about them can help point to the explanation. In a dynamic memory, processing failures are stored under the knowledge structures in which they were encountered. When a subsequent failure is encountered within the same MOP, these cases are accessible.

ALFRED (Riesbeck [23]) presents a model of how failure-driven reminding contributes to incremental learning. In order to understand articles in the domain of economics, the system uses rules that reflect everyday knowledge. If one of these rules of thumb fails, an *exception episode* describing the failure and the recovery procedure is indexed under the faulty rule. When the next failure of the rule is encountered, the original failure is remembered. The previous case provides information to aid in revision of the problematic rule: if it is possible to group the failure episodes into a single class, the rule can be modified to deal correctly with that class.

For example, in one run of the program, ALFRED was given two arguments concerning whether the government should control credit cards as an anti-

inflationary measure. The first advocated control, since credit cards account for $55 billion of the credit in the American economy. ALFRED accepted this argument, using the everyday knowledge that $55 billion is probably a significant amount. A subsequent argument said that credit cards accounted for a negligible portion of the $1.23 trillion total credit in the economy. Using this additional information on the relative size of the credit, ALFRED accepted that its old conclusion was wrong.

The program was then given the input that adding a 10¢ per gallon gasoline tax would decrease consumption by 100,000 gallons a day. This conflicted with the system's expectation, again based on everyday knowledge of amounts, that the effect of a small tax should not be so large. This reminded the program of its failure judging the size of credit with respect to the total size of the economy. In fact, the explanation of the problem in this instance is quite similar: since gas consumption is over 6 million gallons a day, the change produced by the tax *is* actually quite small. Comparison of the reminding to the current failure could suggest the explanation that sizes are relative, and suggest the new rule that we shouldn't form conclusions about them until we have points of comparison.

3.4. Using causal structure to select relevant indices

ALFRED is reminded only when an expectation fails, and only of situations where the failure was attributed to the rule that failed in the original episode. In order to profit fully from experiences, we should be able to be reminded of old cases in a wide range of circumstances. To do this a case should be indexed in such a way that it will be retrieved whenever it is likely to be relevant to current processing.

If the explanation for a failure episode is available at the time of storage, the episode can be indexed to make it available in causally similar situations, in order to warn of likely problems before they occur. Likewise, plans can be indexed in terms of the goals that were active in the original situation, so that remindings will suggest solutions to current problems.

This sort of case-based learning is investigated in CHEF (Hammond [10]). CHEF is a case-based planner that devises recipes in the cooking domain. Instead of basing its planning on a rule library, it relies on its experience: new plans are based on memories of past plans. When a plan fails, or it encounters problems it has previously solved, it is reminded of similar past situations. It then uses them as the basis for dealing with the current situation, and installs the new case in memory for future use. (The use of remindings of past failures to resolve difficulties is also investigated in [31, 32].)

CHEF's learning system uses causal knowledge to focus on important features when integrating plans into memory: when a plan is successful, it is indexed in terms of goals it satisfies and the problems it avoids; when it fails, it is indexed under features that predict the type of failure encountered. Such indexing

allows CHEF to anticipate and avoid problems by being reminded of past failures and of plans it has used to deal with them.

Although CHEF uses past cases in its planning, it starts from scratch when it initially explains a plan failure. The following sections show how experience can facilitate explanation.

4. Explanations in Memory

If we ask someone why a team lost a game, he won't have to reason from first principles. There are many standard answers for why a team lost, and his answer will probably be one of them, such as:

- They were overconfident and didn't train hard enough.
- They were tired from a previous tough game.
- Key players were injured.
- They couldn't take the pressure.
- The opponents wanted revenge.

As MOPs are fossilized plans, *explanation patterns* (XPs) are fossilized explanations. The explanations they package may come from personal experience, or they may capture culturally shared knowledge. For example, proverbs like *too many cooks spoil the broth* and *pouring oil on the fire is not the way to prevent it* offer explanations of undesirable outcomes.

4.1. The structure of XPs

XPs need to include information to support direct application to anomalies, and to have a sufficiently rich internal structure to allow them to be adapted to new situations.

In order for an explainer to recognize when an XP applies directly to a given anomaly, each XP must have a component representing the anomaly it explains. In order for an explainer to preserve and apply the useable portions of a near-miss explanation, XPs must include an explicit representation of the beliefs and belief inter-relationships underlying the packaged explanation.

XPs should also package simple criteria for when the XP is likely to hold. This allows an explainer to avoid re-analyzing each step in the belief support chain when deciding if an XP applies. Also useful are criteria for whether an XP is likely to be useful, even if it doesn't apply directly—that information aids when selecting an XP to try from a set of partially applicable XPs.

The knowledge in an XP can be helpful during planning as well as when recovering from failures. For example, the explanation of an unexpected event may be useful as a trace of a means to bring it about; the explanation of an unusual action may suggest a new way to satisfy particular goals. This planning knowledge is often summarized in a proverb or rule of thumb associated with the XP. Finally, XPs organize in memory the episodes they have been used to explain.

Thus the parts of XPs include:

(1) a representation of the anomaly that the pattern explains,
(2) a set of states of the world under which the pattern is likely to be a valid explanation,
(3) a set of states of the world under which the pattern is likely to be useful, even if it isn't immediately applicable,
(4) a pattern of beliefs, with the relationships between them, that show why the event being explained might have been expected,
(5) a proverb or rule of thumb that summarizes the situation for use in planning,
(6) a set of prior episodes that have been explained by the pattern.

For example, a classic XP is: *killed for the insurance money*. This pattern can be used to explain a premature death. When someone who is heavily insured dies, especially under suspicious circumstances, people sometimes wonder about whether he was killed for his life insurance. The associated XP follows:

Killed for the insurance money

(1) An anomaly which the pattern explains: *untimely death*.
(2) A set of states of the world under which the pattern is likely to be a valid explanation: the combination of either *untimely death* or *death heavily insured* with *relatives didn't love him* or *beneficiary is suspicious character*.
(3) A set of states of the world under which the pattern is likely to be useful, even if it isn't immediately applicable: *deceased was rich*; *relatives didn't love him*; *beneficiary is suspicious character*.
(4) A pattern of beliefs, with the relationships between them, that show why the event being explained might have been expected:
- *beneficiary dislikes policy-holder*,
- *dislike makes beneficiary want to harm policy-holder*,
- *beneficiary has goal to get a lot of money*,
- *inheriting is a plan for getting inheritance*,
- *insurance means that inheritance will include a lot of money*,
- *inheriting requires that the policy-holder dies*,
- *beneficiary kills the insured to harm him and to get money*.

(5) A proverb or rule of thumb that summarizes the situation for use in planning: *a good way to get rich and get rid of someone you don't like at the same time*.
(6) A set of prior episodes that have been explained by the pattern: *deaths seen in movies*, *mafia killings*.

Killed for the insurance money is a *culturally shared* pattern. We learn such patterns from those around us, and have them as a common basis for building explanations. Such an XP is like the restaurant script in that we consider it as a

possibility without examining closely the underlying beliefs. Another type of explanation pattern is the *idiosyncratic* pattern. On the basis of their individual experiences and values, people build up a library of explanations. Idiosyncratic patterns may package an incoherent or irrational explanation, yet people use them to guide their behavior. A real-life example follows:

This generation should fight a war because past ones did:

> When I was collecting voters for an anti-war candidate, one of the people I contacted began to argue with me about the war. He said that the reason that I was against the war was that I was a coward and was just trying to avoid fighting in it. I argued my point of view but he just ended up saying: I fought in World War II when I was your age and now it is your turn to fight in your war.

This example reiterates that the summary need not be a rational conclusion from the conditions, nor is there any reason that the XP has to have the correct explanation of the situation. Yet even when patterns are idiosyncratic and ill-formed, they may be useful to the people who devised them: this is clear from people's reliance on them.

5. An Algorithm for Case-Based Explanation

Our model of understanding is the integration of new facts into the knowledge in memory. If an input conflicts with memory, an explanation must be generated to show why previous knowledge or expectations failed to apply, and to generate correct expectations. We propose the following understanding/explanation cycle:

Step 1. *Anomaly detection*. Attempt to fit a new fact into memory.
- If successful, processing of the fact is complete.
- Otherwise, an anomaly has been detected. The anomaly is characterized according to a classification of anomaly types.

Step 2. *XP search*. Using the failure type as an index, try to retrieve an XP that responds to the anomaly type. If none are retrieved, generate other indices to retrieve potential XPs.

Step 3. *XP accepting*. Attempt to apply the XPs.
- If successful, and the XP could be accepted without revision, memory is updated with the beliefs in the XP. Processing of the fact is complete.
- If a revised explanation was accepted, update memory with the beliefs it involves, and skip to Step 5 (XP integration).

Otherwise, characterize the problems of the inapplicable XPs.

Step 4. *XP tweaking*. On the basis of the anomaly characterization generated in Step 3 (XP accepting), select strategies for revising the XP to repair the problem. New XPs that are generated are sent back to Step 3 to be evaluated.

Step 5. *XP integration*. If a new XP is accepted, store it in memory, making appropriate generalizations.

This cycle is implemented in preliminary form in the SWALE system, which has three modules: the main program (by Leake) handling anomaly detection, XP accepting, and XP integration, and modules handling XP search (by Owens) and XP tweaking (by Kass). The following sections will discuss the issues involved in each of these processes, and the ways we address them.

As a continuing example, we will show the role of each phase of the process as SWALE built explanations for a story that surprised many people when it originally happened. In 1984, Swale was the best 3-year-old racehorse, and he was winning all the most important races. A few days after a major victory he was found dead in his stall. The shocked racing community tried to figure out why. Many hypotheses appeared, but the actual cause was never discovered.

The SWALE system detects the anomaly in Swale's premature death, and builds explanations of the death by tweaking remindings of XPs for other episodes of death. It picks the best of these candidate explanations, and adds it to its XP library for future use.

5.1. Anomaly detection

When SWALE reads a story, it tries to connect the input to its existing knowledge; a fact is anomalous if it cannot be integrated into memory. Part of our effort is developing a small number of anomaly classes for categorizing the problems found. The description of an anomaly in terms of this categorization can then be used as an index into explanations relevant to that anomaly class.

On the basis of a list of more than 170 anomalies and explanations collected at Yale (Kass and Leake [13]), and on the basis of the types of tests required to detect and repair different types of problems, we have grouped anomalies into a small number of classes (currently, 14 types). When SWALE detects an anomaly, it categorizes it according to these types. The anomaly classes include:

- *Role-filler of wrong category*.
  Example: A horse jogging, since joggers are usually human.
- *Premature event*.
  Example: A premature death.
- *Planning problems*.
  Example: A plan to drug a horse to improve his performance before a big race, since the horse would be tested and disqualified.
- *Novel causal connection*.
  Example: A suggestion that the market crash was caused by sunspots.

SWALE detects anomalies when new facts cannot be reconciled with its model of the world. To reconcile facts with its model, the system tries to integrate

them into memory in terms of previous expectations, known patterns, and prior knowledge of specific facts; anomalies are failures of the understanding process. Whether a fact is acceptable or problematic is determined by the following questions, which guide SWALE as it attempts to understand an input fact:

Is the input fact already in memory?
If the fact is already known to the system, no more integration needs to be done.

Does the input satisfy an active expectation?
Routine understanding in SWALE is done with a MOP application process modeled on [3]. Input facts are checked to see whether they satisfy the expectations provided by active MOPs. If so, they are stored in memory, organized by the accepting MOPs [15, 18, 24]. Installing a fact in a MOP activates expectations for future actions. For example, the fact that Swale races at Belmont places Swale in the racing phase of the MOP M-racehorse-life, and generates the expectation that he will race for a few years, live at the stud farm for a few years, and then die.

When an input only partially matches an expectation, the conflicts are detected as anomalous. For example, when SWALE installs the event of Swale's death in Swale's M-racehorse-life, the time of the death (during the racing phase of Swale's life) conflicts with the expectations from M-racehorse-life that the death of a racehorse normally happens a few years after the end of its racing career.

By checking the temporal separation of events as it integrates new facts into a MOP, the program detects *premature event*, *delayed event*, and *premature termination of event sequence*.

If the fact was not expected, can an accepting knowledge structure be instantiated, or can the input fact be accepted in terms of other known patterns?
When an input fact is irrelevant to its active expectations, SWALE attempts to instantiate a new MOP to accept it. For example, when the system begins to process the story of Swale, it accepts the new fact that Swale was a racehorse by instantiating the MOP M-racehorse-life for him.

SWALE also tries to account for facts in terms of standard plans, and by using patterns associated with *role themes* [28]. For example, if SWALE is given the input that someone is a jogger, and then is given input about a specific instance of that person jogging, it uses the jogger's role theme to account for the specific instance.

Trying to accept a fact in terms of plans and goals can lead to detection of *planning problems*.

Do other circumstances make the fact more likely?
Even if a fact cannot be accounted for by an existing knowledge structure,

circumstances may make it plausible. SWALE's MOPs include information about factors that make occurrence of the MOPs more likely. For example, its MOP M-heart-attack includes the information that a predisposing circumstance is for the actor to be high-strung.

For all new facts, another question is considered:

Are the fact's role-fillers reasonable?
Each input fact is checked for whether its role-fillers are reasonable. Associated with each MOP in memory is information on the normative role-fillers (e.g., the filler of the diner role in M-restaurant is usually human, though not necessarily: a recent newspaper story described a restaurant for dogs). One of the wild explanations the system generates is that Swale might have eaten something poisonous at a restaurant. The system rejects the explanation both because horses normally don't eat in restaurants, and because restaurant food is normally wholesome. This check detects the anomaly class *role-filler of wrong category*.

5.2. XP search

A number of studies have shown that people often fail to retrieve relevant analogies (e.g. Gick and Holyoak [8, 9] and Gentner and Landers [7]). If relevant experiences are not retrieved efficiently, much of the advantage of case-based reasoning is nullified. Consequently, a key question in case-based or analogical reasoning is how to select indices for storing and retrieving cases from memory. If a case-based system can usually retrieve the appropriate cases from its memory of cases, even as the number of cases grows, increases in the size of its knowledge base will make its processing more efficient: a system with a large library of cases is more likely to have similar experiences to draw on.

5.2.1. *Using the anomaly characterization as an index*

For an explainer, a natural choice of index is a characterization of the anomaly that needs to be resolved. In straightforward cases, an XP will be indexed directly under the observed anomaly. If we learn of the murder of someone heavily insured, we easily retrieve the pattern *killed for the insurance money*.

SWALE's first approach to retrieval of XPs is to try to retrieve an XP that explained a similar problem. XPs are organized both by the anomaly class they involve, and by the role-fillers for slots of the anomaly; for a given anomaly type, the XPs dealing with anomalies whose fillers are closest in the abstraction hierarchy to the new problem are retrieved first. For example, given an instance of premature death of a beagle, the program would first retrieve any XPs for premature deaths of beagles, than of dogs in general, and then of other animals. (SWALE's abstraction information is actually a net rather than a tree;

any node may have multiple abstractions. The search for XPs of events with the closest-possible fillers is done breadth-first up this abstraction net.)

When SWALE tries to explain Swale's death, the system first retrieves its XPs of premature death in animals. For example, animals are sometimes accidentally run over. However, this XP does not apply to Swale, since he would have been carefully supervised.

5.2.2. *Unusual feature search*

If no XP that deals with the anomaly is available, other strategies must be used to retrieve XPs. Although the causal structure of the episode is not yet available, it is possible to select features likely to be relevant.

One heuristic people often use when explaining is *coordination of anomalies* [26]: if a situation has two anomalous features, a useful strategy for explaining them is to try to connect them. For example, if we see an unknown person in a nearby office, and then hear that a computer disappeared from the building, we may connect them by conjecturing that the person we saw was a thief.

The basic principle of coordination of anomalies underlies another search strategy used by SWALE. When an anomaly occurs and the system cannot retrieve an explanation for the anomaly, it examines other unusual features of the situation, and sees if any XPs indexed under them can account for the anomaly. The system determines features to check by comparing features of objects in the anomaly with normative expectations for their abstractions. For example, Swale is compared against normative expectations for star racehorses (compared to that class he has no unusual features that index XPs), and then to racehorses in general (compared to which class he is in exceptionally good physical condition). Using the indices of death and outstanding physical condition, the system retrieves a possible explanation: the explanation of the death of Jim Fixx, who died because exertion over-taxed a hereditary heart defect.

5.2.3. *Search for folkloric explanations*

When people have no experience to fall back on, they sometimes try to explain by using proverbs, even though they must work to select an appropriate one and to figure out how it applies to the specific case at hand. One index for retrieving proverbs is the class of event they explain. For example, for an anomalous death, we might retrieve *dead men tell no tales*. While this does not apply immediately to the death of an animal, it might be possible to revise it to apply. *Dead men tell no tales* involves killing to keep from being incriminated. This might explain the death of a racehorse who had been stolen, and was killed to destroy the evidence when the people who stole him thought police were closing in. When SWALE exhausts its library of experiences, it attempts to retrieve proverbs that account for events of the type that was anomalous.

5.3. XP accepting

Once an XP has been retrieved, the anomaly explained by the XP is matched with the anomaly to be explained, binding variables of the XP. In some cases this instantiated XP will apply immediately. In the above example of the team's loss, any of the explanations might apply without modification. The XP accepting phase must determine whether the explanation is acceptable and identify any problems that need repair. Since the system may generate a number of explanations, none of which is completely certain, it also must weigh the relative merits of competing explanations.

Evaluation cannot be done in the abstract: it must be influenced by what the explainer knows and needs to learn. Thus an explanation should be judged on the basis of three criteria: whether it answers the explainer's underlying question, whether it substantiates its answer, and whether its answer gives the explainer the supplementary information needed to respond in accordance with the explainer's active goals.

5.3.1. *Relevance of an explanation*

An explanation is relevant if it addresses the underlying question that prompts the explanation attempt. Many EBL systems explain situations that cannot be accounted for by existing schemas. For them, the underlying question is *what caused this event to occur*? Another class of situations needing explanation, which has received less attention, is those where an active expectation is contradicted. To explain the mistaken expectation, an explainer must answer *why did my expectation go wrong*?

As an example of the difference between these questions, and of the reason that addressing expectation failures is important, suppose we expect a restaurant to be good because a restaurant guide recommended it. If we have a bad meal there, an explanation of the *event* of the bad meal might be that the chef is incompetent. EBL systems could generalize this to learn that restaurants with incompetent chefs serve bad meals. However, we could also learn from explaining why we predicted incorrectly. Since our prediction was based on the review, a relevant explanation might show why we shouldn't have trusted the guide. If we found that the guide always gives favorable reviews to places that advertise in it, we would know to avoid similar problems by discounting its praise of advertisers. In general, a system confronting an expectation failure must try to explain *both* why the surprising event happened, and why it shouldn't have formed the expectation.

What constitutes a relevant explanation of an event is well understood: it is simply a causal chain leading to the event. Since SWALE explains expectation failures, it needs explanations that address them as well. An explanation of an expectation failure must show a flaw in the reasoning that led to the expectation. For example, an explanation that is relevant to the expectation failure

underlying a *premature event* is one that shows why normal temporal delays were superceded: either by catalyst (e.g., bread might have risen quickly because a room was unusually warm), or by an independent cause that brought the event about earlier than expected (e.g., someone died prematurely because he was hit by a car). SWALE has procedures, indexed by anomaly types, to check whether an XP's belief support chain gives a reason that the original expectation does not apply.

5.3.2. *Judging believability*

SWALE judges believability of an XP's hypotheses by integrating them into its model of the world, and seeing if they generate anomalies. Thus this phase of evaluation is simply trying to understand the facts of the explanation in light of its other knowledge.

To check the believability of an XP's causal links, SWALE first tries to retrieve the rule used from the system's rule library. If the rule is known to the system, templates for its antecedent and consequent are matched with the belief nodes linked in the XP, and any restrictions on role-fillers of the rule are checked. If the rule is unknown, the XP is returned to the tweaker for elaboration of the connection.

5.3.3. *Evaluating detail*

Even if an explanation is believable, it is unsatisfying unless it is adequately detailed. The level of detail needed depends on the explainer's goals. For example, when a car owner has car trouble, all he needs to know is whether the problem is transient (such as getting a bad tank of gas) or needs to be repaired. A mechanic requires a more detailed explanation, since he must trace the problem to a faulty part or a needed adjustment.

As a first step towards evaluating detail in terms of explainer needs, SWALE has a library that associates role themes [28] with tests for the detail that theme-driven actors need in different situations. For each theme, the information has two parts:

(1) *A list of types of anomalies whose resolution is important to the theme.* For example, two types of anomalies important to a veterinarian are animals' physical changes (e.g., weight loss) and behavioral changes (such as loss of appetite), since they might be signs of a health problem. A detective would investigate anomalies such as premature deaths and violent acts.

(2) *For each relevant anomaly, a test to judge whether the explanation provides enough information for a theme-driven actor to respond.* For example, a detective would need to trace the cause of a premature death until he found whether it was due to natural causes or foul play.

The program applies this knowledge to judge the elaboration of explanations. Its criteria need to be expanded so that in addition to standard

theme-based requirements, requirements for detail could be dynamically determined on the basis of active goals and plans of the system.

SWALE now has only simple criteria for comparing explanations. Relevant explanations are divided into three classes of believability: *confirmed* (all assumptions verified or accounted for by expectations), *assumed* (some hypotheses couldn't be verified, but there are no problems reconciling memory with the hypotheses), and *unacceptable* (problems have been found). Out of the most believable explanations, the system tries to elaborate those with insufficient detail, and the evaluator finally accepts any resultant explanation that is relevant, believable and adequately detailed. If no explanation is acceptable, the explanation and a description of its problems are passed to the tweaking phase of the process, which attempts to repair the problems.

5.4. XP tweaking

The goal of the tweaking process is to make it possible to use an XP in a wide range of circumstances. Even when a new situation shares few features with the one explained by an XP, the XP can give a starting point for generating hypotheses.

5.4.1. *Applying explanations to new situations*

For any analogical reasoning, a key issue is how to map old knowledge to a situation. Work on analogy has presented a number of approaches. Gentner's *structure-mapping theory* [6] assumes that specific objects from a source domain have been previously identified with objects in a target domain, and gives criteria for exporting some relationships from the source domain to the target domain. The relationships chosen to map are those that have high systematicity (i.e., that belong to coherent sets of mutually consistent relationships). For example, in the analogy "the hydrogen atom is like the solar system," the relationship more-massive-than is preserved, because it is related to other relations such as distance and attractive-force, while the relationship hotter-than is not.

However, as Holyoak [11] points out, it is possible to imagine many relationships that hotter-than *is* related to, so syntactic criteria alone are not sufficient to determine which relationships to preserve. Also, Gentner's approach implies that for a given source domain, the same relationship with a target should always be favored, regardless of the circumstances. This conflicts with our experience: the aspects of an analog that people notice depend on context.

Holyoak [11] suggests that pragmatic context, such as goals of a problem solver, focuses analogical mapping. When a previously solved problem is applied to a new situation, the first step is to generate an abstraction of the old problem's statement that is sufficiently general to apply to the new case. This

mapping of goals, constraints and operators is extended to include an abstraction of the old solution, which is then mapped to the new problem.

SWALE's mapping between cases is also determined by the problem the system needs to resolve, but is formed directly. SWALE retrieves an XP that explains a similar anomaly, and matches the old anomaly with the one to be explained. This match establishes the correspondence between the two cases; on the basis of the variable bindings it determines, SWALE instantiates the old XP.

Whenever transfer of a previous case is attempted, the mapping may break down: the mapped case may need to be adapted to deal with aspects that have no direct analog in the source case. For example, someone who knows how to drive a car with automatic transmission will not be able to use a manual transmission without adding shifting gears to the driving procedure. Because SWALE maps XPs without any further abstraction of their structure, the ability to repair such problems is especially important. To do such repairs, the system maintains a set of revision rules. The flavor of this approach is similar to that of Carbonell [1], who transforms a solution applicable to an old problem into one for a new problem; SWALE's *tweaking strategies* are similar to Carbonell's analogy transformation operators.

5.4.2. *SWALE's tweaking algorithm*

Tweaking strategies are procedures for revising or repairing parts of XPs that are inapplicable to the situation being explained. The types of revisions suggested by SWALE's strategies range from quite abstract to highly specific. Some strategies call for operations such as splicing in new steps to provide support for unsupported links; these may require considerable search to be applied. Others give a highly constrained way of patching a specific problem.

Which tweaking strategies are applied to a problem depends on the problem characterization generated by the XP accepting process. SWALE's method of selecting tweaks is similar to that used in CHEF [10] to repair plan failures: CHEF's goal failure configurations are used as indices for retrieving plan repair strategies; SWALE's anomaly types are used as indices into its library of repair strategies. The basic tweaking cycle is:

Step 1. Index from a failure description to a tweaking strategy.

Step 2. If strategy is found, attempt to apply it.

Step 3. If no tweak applies to the problem, abandon the belief that caused the failure.

5.4.3. *Examples of SWALE's tweaking strategies*

SWALE uses tweaking strategies at varying levels of specificity. If a failure description gives a very general characterization of a problem, the tweaks indexed under that description can only give general repair suggestions. For

examp.., to repair the problem *novel causal connection*, which occurs when an XP linking two facts does not substantiate the connection between them, SWALE uses the strategy:

Find connecting XP
If the causal connection between beliefs in an XP cannot be substantiated, splice in the reasoning of another XP that substantiates the connection.

However, for tweaking to be efficient, it is important whenever possible to constrain the alternatives considered. Our approach to repairing *role-filler of wrong category*, when the role being filled is actor of an action, shows the types of restrictions that are useful. For an action with a problematic actor, the system attempts two types of tweaks. The first tries to retain the actor, but substitute another action for the problematic one. If possible, it uses information about the actor to guide selection of a substitute action that is likely for him. The second type of strategy retains the basic action, and tries to use characteristics of that action to suggest likely substitute actors.

5.4.4. *Strategies for substituting an alternative action*

Substitute alternate theme
If an XP depends on an actor having a particular role theme [28], but the actor in the current situation doesn't have that theme, try to substitute a role theme of the current actor that has causally equivalent effects.

For example, the XP for Jim Fixx's death depends on Jim Fixx's jogger role theme to support the belief that Fixx was running, but that theme does not apply to Swale since Swale is not human. Using *substitute alternate theme*, SWALE examines the themes of Swale to see if any support the belief that he was running. Since Swale's horse-racing theme is a support, the tweaker generates the explanation that Swale's racing caused him to run, which caused exertion, which over-taxed a heart defect and caused his death.

Substitute related action
When an action doesn't apply, it may be possible to obtain the desired effects by substituting a similar action that is applicable.

For example, when someone visiting New York is late for a meeting, we might remember that he was late once before because he took the wrong subway train. If we remember that there is a subway strike, we might use *substitute related action* to generate the hypothesis that this time he got on the wrong bus.

To hypothesize related actions, the system uses its abstraction hierarchy: it selects abstractions of the problem action that fit the causal structure, and looks at their specifications to find other possibilities. Thus although one abstraction of subway could be "dangerous place," the system would not attempt to splice dangerous places into the explanation, but would consider other modes of transportation.

5.4.5. *A strategy for substituting an alternative actor*

Tweaks to substitute an alternative actor have not been implemented, but an example of such a tweak is:

Substitute an actor who could benefit
When the hypothesized actor is unlikely, consider actors with goals satisfied by the action. For example, if we investigated the explanation that Swale's owners killed him to collect his insurance, we would rule it out because Swale was under-insured. But this might make us look at who might have had reasons to kill him, suggesting possibilities like *killed by a competitor's owner.*

SWALE currently has a small set of tweaking strategies. When a problem is encountered, all applicable ones are tried, and the resultant set of new XPs is evaluated. However, we anticipate that there will be many more strategies; heuristics will be needed to guide more selective retrieval.

5.5. XP integration

Once SWALE has accepted a new XP, the XP is integrated into its library of explanations. This allows it to efficiently explain similar anomalies in the future. In order to integrate the XP into memory, SWALE performs two tasks: generalization of the new XP, and indexing it in memory.

5.5.1. *Generalization*

The traditional assumption in systems using explanation-based methods is that they have access to perfect causal knowledge of their domains. Given this assumption, it is reasonable for them to generalize their explanations to the greatest extent licensed by their rules. However, knowledge of situations in the real world is usually approximate, and too many factors are involved in events to explain them completely. Consequently, people rarely make explanations that are deductively valid.

In response to the incompleteness of domain theories for real-world events, the only generalization SWALE always does is variabilizing of the belief support chain of an XP. Further generalization is only done to the extent necessary to accommodate examples the system has seen. When the system has retrieved an explanation that doesn't fit, but successfully tweaked it to apply, the system compares the two. For segments of the belief support chains that do not match, the system tries to produce the most conservative generalization that subsumes the original explanations. (Generalization based on the comparison of two explanations' causal structure is also suggested by Kedar-Cabelli [14].)

For example, when the system accepts that Swale's death was caused by exertion caused by running in races, combined with a heart defect, it compares the new XP to the explanation of Jim Fixx's death that it tweaked to generate the Swale explanation. Both explanations share the same structure, except that

Swale's death was supported by the role theme of running in horse races, and Jim Fixx's death by the role theme of jogging. The system uses its abstraction net to look for a shared abstraction of horse racing and jogging that could substitute for an initial segment of nodes in the belief support chain. It finds that having a role theme of physical exertion supports the basic chain, and it generalizes the two explanations to form the generalized explanation *performing physical exertion raised the actor's exertion level, which over-taxed a heart defect, causing a heart attack, which led to death.*

Once the generalized explanation of Swale's death has been installed in memory, a new class of anomalies can be explained efficiently by retrieving and directly applying the XP. For example, even though the form of a plow horse's exertion is different from that of Swale and Jim Fixx, the generalized XP could be retrieved to explain his premature death. The new XP is also available for tweaking to fit more distantly related situations.

5.5.2. *Indexing*

Initial work on explanation-based methods focused on using explanations for generalization [21]. A wider view is suggested by DeJong and Mooney [4], who propose that explanation is important not just for generalization, but for specialization of concepts. We believe that EBL includes another important task: the selection of indices for storing cases or generalizations in memory.

For real-world situations, the problem of applying old cases to new events is best approached by concentrating on indexing an episode so that it will be retrieved in relevant cases, rather than doing all generalization when the episode is stored. (Our perspective is similar to that of Hammond [10], who argues that a case-based planner does not generalize the plan itself, but instead generalizes the indices under which the plan is stored.) When a new case cannot be accounted for by an existing XP, the retrieved cases can be modified to fit the situation, and generalization can then be done to form a new XP that explains both cases.

In addition to avoiding overgeneralization when causal knowledge is uncertain, modifying cases when necessary gives more flexibility: old cases can contribute to dealing with situations that may not be subsumed by any obvious generalization of the explanation.

For this approach to be effective, we need to index cases so that similar cases will be retrieved first, but apparently dissimilar cases will be available when there are no direct solutions. While surface features have a large effect on people's retrieval of analogies [7], we believe that indexing with deeper features, such as causal factors, is very important [24]. SWALE uses two classes of features to index XPs:

– *The anomaly categorization.* As was discussed in the section on XP search, the XPs that explain a particular type of anomaly are organized first by the anomaly type, and then hierarchically on the basis of their primary role-fillers.

For example, XPs considered for premature death of a racehorse would first be premature deaths of racehorses, then of horses, and then of animals.

– *Causal preconditions of the situation.* The system uses the explanation to identify features that are causally relevant, and indexes new XPs under the event type and the causally relevant features. For example, the generalized XP for Swale's death is indexed under *death + exertion.*

We believe that the following types of features are also important, even though they are not directly implicated in the explanation. While we have not implemented indexing of new XPs under these types of features, they are used to index XPs in SWALE's initial XP library:

– *Features participating in anomalous aspects of the situation described by the XP.* One reason that Jim Fixx's early death was anomalous was that he was in excellent physical condition. The Jim Fixx XP is indexed under the combination *death + good physical condition.*

– *Membership in stereotyped groups.* Death of a rock star, for example, might remind us of the deaths of Jimi Hendrix or Janis Joplin. In SWALE, the explanation of Janis Joplin's death from a drug overdose is indexed under *death + star performer.*

6. Creative Explanation

Creativity is sometimes thought of as an almost mystical process, where a creative act produces something out of nothing. However, there is a fundamental flaw in any attempt to consider creativity as something that transcends our knowledge. No human discovery occurs in a vacuum: If we really accepted that creative acts must build something out of nothing, we would be hard-pressed to argue that human creativity exists.

It seems obvious that to come up with new ideas, we must start with old ideas. Creativity comes from retrieving knowledge that is not routinely applied to a situation, and using it in a new way. Thus the key issues of explanation by XPs, search and adaptation, are also the key issues for creativity.

Creativity is not an all-or-nothing thing: there is a spectrum of creativity, going from minor revisions of old knowledge all the way to a totally new world view. Application of XPs can take place at any point along the spectrum of creativity, from the completely noncreative application of a perfectly appropriate XP, to the very novel use of an inappropriate XP that must be totally revised to be useable. This section presents some of the explanations that SWALE generates along that spectrum, and considers what is necessary to extend such a system's creativity.

6.1. The range of explanations generated by SWALE

In order to give an idea of the range of explanations that a system using our approach can generate, below are some remindings that the SWALE system has,

and a few of the explanations it generates from them. Some are reasonable explanations; others are quite fanciful or can be ruled out on the basis of other knowledge. However, they show that a memory-based explanation system, even if it has a limited range of XPs and of retrieval and tweaking strategies, can come up with a variety of interesting explanations.

Reminding. Thinking of other deaths of those in peak physical condition causes the system to be reminded of the death of the runner Jim Fixx, who died when his running over-taxed a hereditary heart defect.

Explanation. Swale might have had a heart defect that caused his racing to prompt a heart attack.

Reminding. Thinking about other deaths of young stars, the system is reminded of Janis Joplin's death from a drug overdose.

Explanation 1. The pressure of being a superstar was too much for Swale, and he turned to drugs to escape. He died of an overdose.

Explanation 2. Swale might have been given performance-enhancing drugs by a trainer, and died of an accidental overdose.

Reminding. Thinking of folkloric causes of death causes the system to recall the old wives' tale *too much sex will kill you*.

Explanation 1. Although racehorses are prohibited from sex during their racing careers, Swale might have died of a heart attack from the excitement of just *thinking* about life on the stud farm.

Explanation 2. Swale might have committed suicide because he became depressed when thinking about sex.

Explanation 3. Swale might have died in an accident when he was distracted by thinking about sex.

6.2. Requirements for creative explanation

The above examples show that interesting explanations arise when we try to use an XP that doesn't quite apply. In order to obtain creative explanations, an explainer might try to *intentionally misapply* XPs. Interesting ideas can arise from using old explanations to deal with situations where those explanations were never intended to be used.

While using an XP that doesn't apply gives a fresh perspective on a situation, the idea of building a system to intentionally misapply XPs raises many issues: Which XPs do you retrieve? Which tweaks should be applied? How long should the tweaking process be continued? As our research progresses, we hope to be able to answer these questions. However, we can now suggest some of the things that are needed to build creative case-based explainers:

We need heuristics for the intentional reminding of explanation patterns

XP retrieval is the process of formulating questions to memory: we characterize an anomalous situation in terms of a set of indices, and ask what XPs in memory explain similar situations. When no answer is available, we must

reformulate the question into one that we can answer [15, 27]. When no solution is directly available, people often fall back on asking standard questions that give background information. Answers to *explanation questions* like *what physical causes underlie this event*?, *what special circumstances made the event happen now*?, *what motivates the actor of this surprising action*?, *how did the victim enable this bad event*?, or *what groups might the actor be trying to serve*?, may suggest relevant factors that can be used as indices for XP retrieval. Though the XPs accessed in this way might not be directly applicable, it may be possible to adapt them. A creative system needs a set of explanation questions for gathering information, rules for selecting which questions to apply in a given situation, and rules for transforming them to fit.

We need tweaking strategies that can do significant revisions
Tweaking must also be able to make significant revisions. Rather than requiring tweaks to always maintain causal structure, we should allow them to make broad changes. Their revisions will not always be successful, but failures produce new possibilities for still more revision.

We need heuristics for knowing when to keep alive seemingly useless hypotheses
In addition to choosing between explanations generated by the system, the evaluation process also has a more direct part in the creative process. We cannot tweak a candidate explanation indefinitely; the evaluator must decide whether a hypothesis is worth pursuing. This estimation will always be imperfect, but the better it is, the more resources the system will be able to devote to fruitful revision of XPs. One heuristic would be to continue tweaking an explanation as long as each tweak generates a better explanation. But the decision whether to continue tweaking should also depend on the availability of competing candidate XPs, and on an estimate of how important the final explanation is to goals of the system (since that affects how many resources should be expended explaining).

We need a system with a rich memory of explanations
Finally, a creative case-based explainer must have access to a wide range of explanation patterns. There are two ways that people or machines might learn new XPs: by being taught them directly (as children are given explanation patterns by parents, teachers, or friends), or by learning new ones through creative misapplication. One step towards making a computer creative would be to collect an extensive list of XPs that it could use as the starting point for adaptation. Many interesting explanations might be constructed starting with a collection of culturally shared XPs, such as proverbs.

7. Conclusion

Machine learning will be successful when computers are able to learn interesting things. Because of the importance of explanation-based learning, constructing interesting explanations is important.

We have argued for a case-based approach to building explanations: that just as people use experience to explain, so should computers. For this approach to be successful, systems will need an extensive library of explained episodes, in a form that is flexible enough to permit revision. We have defined the *explanation pattern* knowledge structure to package the information needed to support revision.

Case-based explanation requires heuristics for retrieving and revising XPs to deal with new situations, and we have sketched the algorithms SWALE uses to generate new XPs. In order for a system to improve future performance on the basis of experience, learning the XPs it generates is also important. While we believe that XPs must be indexed under a wide range of features, our current implementation of SWALE indexes them under two main types of indices. One is a categorization of the anomaly being explained: XPs can be used most efficiently when they are retrieved for anomalies similar to those they were built to explain. The other major class of indices is causally relevant features; we suggest that *explanation-based indexing* is an important type of explanation-based learning.

SWALE deals with incomplete and uncertain knowledge, which is unavoidable in real-world domains. Consequently, our generalization strategies for XPs are very conservative. We believe that the flexibility necessary to deal with a range of events should be maintained by indexing cases under a wide range of indices and modifying the case to fit once it is retrieved, rather than by trying to generalize as much as possible when it is generated.

The SWALE project shows that a memory-based explanation system, even if it has a limited range of XPs and of retrieval and tweaking strategies, can come up with interesting explanations. We have sketched issues that will have to be confronted to attack creative explanation more thoroughly: the key problems of creative explanation are the problems of retrieving and tweaking existing explanations. Each is a major problem in its own right, but they are problems on which we can make progress.

Appendix A. SWALE's Representation of Beliefs and Belief Supports

Input to SWALE, and the beliefs and links in XPs, are represented in terms of a limited number of belief types. In the current implementation of the system, the following classes of facts are represented:

– *Action description.* The fact that an action occurred. For example, that Swale raced at Belmont.

– *Theme description.* The fact that an actor frequently performs a certain class of action. (This is part of the information in *role themes* [28].) For example, a jogger tends to go jogging frequently.

– *Value description.* That fact that a given attribute of an object has a given value. For example, that Swale's color was brown.

– *Packaging description.* The fact that an object or event fills a role in a MOP. For example, that Swale filled the role of actor in the MOP representing racehorse-life.

– *Goal description.* The fact that an actor holds a certain goal. For example, the fact that Swale's owner had the goal for Swale to win races.

In the belief support structure of an XP, assertions of the above types are linked by nodes representing the inferences by which acceptance of one fact would support acceptance of another.

As described above, the rules used by the system are not deductive rules; a reasonable belief support chain cannot be considered a proof that an inference is true. Because the correctness of a rule's conclusion depends on many factors, people sometimes require a number of independent supports before they believe an event occurred. In *killed for the insurance money*, either the beneficiary wanting to hurt the policy-holder or wanting to inherit money is support for the belief that he killed the policy-holder. Even though we wouldn't normally believe that either one alone resulted in a murder, the two together are fairly convincing evidence.

The types of rules used by the program include:

– *Physical causation.* Effects accounted for by physical or biological laws. For example, that running raises exertion level.

– *Social causation.* Effects we expect because of custom or social patterns. For example, that a Frenchman will like wine.

– *Economic causation.* Effects that we expect because of economic principles. For example, that a decrease in supply will cause prices to increase.

– *MOP sequence.* That in a given context, certain events tend to follow others. For example, that in a fast-food restaurant, customers pay immediately after ordering.

– *Specification.* That an object or event belonging to a certain category can be inferred to belong to a more specific subcategory if it has certain features. For example, that a successful rock performer belongs to the category of rock star (which carries with it default assumptions about his life style, etc.).

– *Preservation goal activation.* That a threat to a desired state prompts a goal to neutralize the threat. For example, that working in a high pressure environment causes people to want to decrease stress.

Appendix B. Output from SWALE

The annotated output below demonstrates our case-based explanation process for a few of the explanations the system generates. (The trace generated by the program is more complete; some output has been deleted.)

B.1. Detecting that Swale's death is premature

The following output is generated by the program as it instantiates the MOP

racehorse-life to accept input about Swale's Belmont victory. On the basis of its representation of racehorse-life, the program generates the expectation that he will go to the stud farm in a few years, and die a few years later. When the program is given the input that he died a week after the race, it detects that his death is premature.

Input. The fact *Swale won the Belmont Stakes*, represented as:

```
[HORSE-RACE
  ACTOR: SWALE
  LOCATION: BELMONT
  TIME: TIME-TOK-1
  OUTCOME: WIN]
```

Output from processing:

```
Integrating SWALE's HORSE-RACE into memory.

Trying to link SWALE's HORSE-RACE to SWALE's active MOPs.

The fact SWALE's HORSE-RACE will be stored as the HORSE-RACE scene
  in SWALE's RACEHORSE-LIFE.
```

Input. The fact *Swale died a week later*, represented as:

```
[DEATH
  ACTOR: SWALE
  TIME: TIME-TOK-2]
```

(TIME-TOK-2 represents a time a few weeks after TIME-TOK-1 above.)
Output from processing:

```
Integrating SWALE's DEATH into memory.

Trying to link SWALE's DEATH to SWALE's active MOPs.

The fact SWALE's DEATH will be stored as the SWALE's DEATH scene in
SWALE's RACEHORSE-LIFE.

Temporal anomaly detected:
  SWALE'S DEATH occurs abnormally early in RACEHORSE-LIFE.
```

B.2. Explaining an anomaly by tweaking existing XPs

To resolve the anomaly, the system first tries to retrieve XPs indexed under similar anomalies. None of the XPs it retrieves are applicable, so the program

tries to retrieve XPs indexed under other features. To retrieve XPs indexed by unusual features of Swale, the search module climbs the abstraction hierarchy, starting with the node representing Swale, and attempts to retrieve XPs indexed under any features of his that are unusual with respect to the given category. The following output traces the search process:

```
Asking explorer to find possible explanation.

Considering SWALE's DEATH as an instance of a generalized DEATH.

SWALE's NAME is SWALE, while
  a generalized SUCCESSFUL-RACEHORSE's NAME is UNKNOWN.
  Nothing found connecting (NAME SWALE)
  with DEATH.

A generalized SUCCESSFUL-RACEHORSE's HEALTH is HIGH, while
  a generalized RACEHORSE's HEALTH is UNKNOWN.
  Nothing found connecting (HEALTH HIGH)
  with DEATH.

A generalized SUCCESSFUL-RACEHORSE's PHYSICAL-CONDITION is
HIGH, while
  a generalized RACEHORSE's PHYSICAL-CONDITION is UNKNOWN.
  Found JIM-FIXX-XP-263 indexed under DEATH
  with index: (PHYSICAL-CONDITION HIGH).
```

The Jim Fixx XP (that his jogging role theme caused him to run, which caused exertion that combined with a heart defect to cause a heart attack) is retrieved, and the XP is evaluated. In the following output, the program detects that Swale is not a suitable role-filler for the ACTOR role in jogging, since the actor is normally a human. The program then uses the tweaking strategy *substitute alternate theme* to find another support for the belief that Swale was running.

```
Checking #{EXPLANATION 183 FIXX-XP}.

Found problem:
  [XP-FAILURE
    TYPE: INAPPLICABLE-THEME
    ROLE: ACTOR
    PROBLEM-DESCRIPTION:
      [TYPE-MISMATCH
        DESIRED-ATTRIBUTE: #{TYPE-NODE 140 HUMAN}]
    BELIEF-LABEL: JOG]

Attempting to tweak XP titled FIXX-XP.

Searching for appropriate modification strategies . . .
One strategy retrieved.
```

```
Attempting to apply a modification strategy:
  "Substitute a theme that is associated with the actor"
  to XP: FIXX-XP.

The belief labeled JOG is a support in the following way(s):
  It supports the belief RUN via support type MOP-SCENE.
  Asking memory for other life-themes for SWALE.
  Associated themes:
    SWALE often has the ACTOR role in the HORSE-RACE theme.
    SWALE often has the ACTOR role in the EAT-OATS theme.

Seeing whether any of these themes can substitute...
```

Swale's horse-racing theme is successfully substituted for the jogging theme in the XP. The system then checks whether Swale's other known theme, eating oats, could provide support for RUN. That tweak fails.

The explanation that Swale had a heart defect, and died when exertion of horse racing over-taxed his heart, is evaluated. The explanation does not conflict with the system's beliefs, but the premise that Swale had a heart defect cannot be substantiated.

```
The explanation #{EXPLANATION 187 FIXX-XP-1} may be OK,
but requires some assumptions.
```

Since the explanation cannot be confirmed, the program seeks alternative explanations. No more XPs are indexed under unusual features of Swale, so the program looks for folkloric XPs.

```
Looking for folkloric explanations of DEATH...
The explorer has returned #{EXPLANATION 189 TOO-MUCH-SEX}.
```

The explanation retrieved is the old wives' tale *too much sex will kill you.*

The program's representation of the MOP for a racehorse's life includes that he is prevented from breeding during his racing days, and then is sent to the stud farm to be bred. Thus the old wives' tale might have applied later in his career, but doesn't apply during his racing days. When an XP hypothesizes that an event took place, but that event could not have happened early enough to contribute to the explanation, one of SWALE's tweaking strategies is *substitute anticipation*: see if effects of *anticipation* of the event could have been similar to those of the hypothesized event itself. (This rule is reasonable in a number of domains. For example, if the rise in a company's stock is explained by the XP *new product introduction makes stock rise*, but the introduction actually didn't take place before the rise, people's anticipation of the new product is a possible explanation.)

The problem, and the attempt to repair it by tweaking, are reflected in the following output:

```
Checking #{EXPLANATION 189 TOO-MUCH-SEX}.

Due to being a SUCCESSFUL-RACEHORSE, SWALE might be prevented
from SEX.

Misorder found: SWALE hasn't yet reached SWALE's STUD-FARM.

Found problem:
  [XP-FAILURE
    TYPE: INAPPLICABLE-THEME
    ROLE: ACTOR
    PROBLEM-DESCRIPTION:
    [MOP-SCHEDULING-FAULT
      POSITIONING-PROBLEM: [ACTION-TOO-EARLY]
      MOP-NAME: RACEHORSE-LIFE
      LAST-SUCCESSFULLY-PROCESSED-SCENE: 4
      MATCHING-SCENE: 3]
    BELIEF-LABEL: SEX]

Attempting to tweak XP titled TOO-MUCH-SEX.

Searching for appropriate modification strategies . . .
2 strategies retrieved. Will try each in turn.

Attempting to apply a modification strategy:
  "Try substituting anticipating action for action itself"
  to XP: TOO-MUCH-SEX.
```

This tweak generates the explanation that Swale's death was caused by thinking of future sex. The other tweaking strategy tried, *substitute a theme that is associated with the actor* for the XP's theme of sex, fails to yield possible causes of death.

The XP *death caused by thinking of sex* is then evaluated. The connection between thinking of sex and death is not substantiated:

```
Checking #{EXPLANATION 190 TOO-MUCH-SEX-1}.

Explanation has unconvincing support.

Attempting to tweak XP titled TOO-MUCH-SEX-1.

Searching for appropriate modification strategies . . .
One strategy retrieved.

Attempting to apply a modification strategy:
```

"Splice in another explanation pattern to explain a connection"
to XP: TOO-MUCH-SEX-1.

To construct a link, the program chains forward one step from sex to its effects, and tries to retrieve XPs linking them to death. One effect of thinking of sex is depression (if you're thinking about problems in your love life):

Looking for an XP to connect:
?X?'s VERY-LOW EMOTIONAL-STATE ... with ... ?X?'s DEATH

Found an XP to connect ?X?'s VERY-LOW EMOTIONAL-STATE to ?X?'s DEATH:
XP is #{XP 194 DISTRAUGHT-SUICIDE}

The resultant XP is that Swale was depressed because of thinking about sex, and he committed suicide.

From distraction and excitement, two other possible effects of thinking of sex, the program generates the explanations that Swale might have died from an accident caused by his being distracted, or that he might have died from a heart attack caused by excitement. The three new XPs are then evaluated. Although none of the new XPs can be confirmed, the attempt to retrieve more potential XPs is unsuccessful, so the program selects the best of the candidate XPs.

All of the XPs are judged relevant to Swale's death, since each suggests a new factor intervening to cause death before old age. When the system judges believability, death from suicide is ruled out (suicide is motivated by anticipating good effects from death, while the program assumes that animals are only aware of the effects of events they have already experienced). Accidental death is ruled out because racehorses are carefully supervised.

Death from being excited and having a heart attack is possible. But since Swale's becoming excited is an unsubstantiated hypothesis, while his running can be confirmed on the basis of his racing role theme, the explanation *heart attack caused by running and a heart defect* is preferred.

Finally, detail must be evaluated. In the output below, the program evaluates detail with respect to the detective's role theme:

Checking sufficiency of explanation for detective's role theme.

Anomaly is a negative health change, so adequate detail is important to detective.

Checking whether the explanation traces SWALE's DEAD HEALTH to natural causes, to an accident, or to a crime and suspect.

Explanation hypothesizes ORGAN-FAILURE as a natural cause:

HEART-1078's M-HEART-ATTACK. It also shows a physical-result chain between this cause and SWALE's DEAD HEALTH. Consequently, level of detail is acceptable.

Conclusion: the explanation is ACCEPTABLE.

After the explanation is accepted, the old and new XPs are generalized:

Looking at possible generalization of original and tweaked XPs.

Looking for a generalization of JOG and HORSE-RACE which will support the same causal chains.

The theme of actor in PHYSICAL-EXERTION accounts for all necessary facts.

Installing generalized XP.

Indexing the XP under the anomaly class it explains.

Installing a class of specializations of DEATH explainable by FIXX-XP-1.

Adding to memory the facts implicit in #{EXPLANATION 187 FIXX-XP-1}.

B.3. Using the new XP

Since the new XP has been installed in memory, SWALE can retrieve it and apply it without modification to another story in which it detects a similar anomaly. The following output was generated in processing the story of a racehorse named Last Chance Louie, who died a few weeks after the Kentucky Derby.

When the anomaly of Louie's premature death is detected, three XPs of similar anomalies are retrieved from the program's XP library. *Death from exertion combined with heart defect* is indexed under premature death of an actor with a role theme of exertion, while *death by being run over* explains the premature death of an animal, and *death from illness* explains the death of any living thing. Since the exertion theme gives a more specific characterization of a racehorse than animal or living thing, the first XP SWALE tries is *death from exertion combined with heart defect.* This XP applies without revision.

Temporal anomaly detected:
LAST-CHANCE-LOUIE'S DEATH occurs abnormally early in RACEHORSE-LIFE

Trying to pull up XPs indexed by the anomaly.

Seeing if one of these XPs is relevant:
FIXX-XP-1

XP-EARLY-DEATH-FROM-RUN-OVER
XP-EARLY-DEATH-FROM-ILLNESS

Checking #{EXPLANATION 209 FIXX-XP-1}.

The XP is accepted.

ACKNOWLEDGMENT

We would like to thank Chris Riesbeck and Alex Kass for helpful comments on a draft of this paper.

This work is supported in part by the Air Force Office of Scientific Research, under grant 85-0343, and by the Defense Advanced Research Projects Agency, monitored by the Office of Naval Research under contract N00014-82-K-0149.

REFERENCES

1. Carbonell, J.G., Learning by analogy: Formulating and generalizing plans from past experience, in: R.S. Michalski, J.G. Carbonell and T.M. Mitchell (Eds.), *Machine Learning: An Artificial Intelligence Approach* (Tioga, Palo Alto, CA, 1983).
2. Carbonell, J.G., Derivational analogy: A theory of reconstructive problem solving and expertise acquisition, in: R.S. Michalski, J.G. Carbonell and T.M. Mitchell (Eds.), *Machine Learning: An Artificial Intelligence Approach* **2** (Morgan Kaufmann, Los Altos, CA, 1986) 371–392.
3. Cullingford, R., Script application: Computer understanding of newspaper stories, Ph.D. Thesis, Tech. Rept. 116, Yale University, New Haven, CT (1978).
4. DeJong, J. and Mooney, R., Explanation-based learning: An alternative view, *Mach. Learning* **1** (1986) 145–176.
5. Dietterich, T. and Michalski, R., A comparative review of selected methods for learning from examples, in: R.S. Michalski, J.G. Carbonell and T.M. Mitchell (Eds.), *Machine Learning: An Artificial Intelligence Approach* (Tioga, Palo Alto, CA, 1983) 41–81.
6. Gentner, D., Structure-mapping: A theoretical framework for analogy, *Cognitive Psychol.* **7** (1983) 155–170.
7. Gentner, D. and Landers, R., Analogical reminding: A good match is hard to find, in: *Proceedings IEEE 1985 International Conference on Systems, Man and Cybernetics* (1985) 607ff.
8. Gick, M.L. and Holyoak, K.J., Analogical problem solving, *Cognitive Psychol.* **12** (1980) 306–355.
9. Gick, M.L. and Holyoak, K.J., Schema induction and analogical transfer, *Cognitive Psychol.* **15** (1983) 1–38.
10. Hammond, K.J., Case-based planning: An integrated theory of planning, learning and memory, Ph.D. Thesis, Tech. Rept. 488, Yale University, New Haven, CT (1986).
11. Holyoak, K.J., The pragmatics of analogical transfer, in: G.H. Bower (Ed.), *The Psychology of Learning and Motivation* (Academic Press, Orlando, FL, 1985).
12. Kass, A., Modifying explanations to understand stories, in: *Proceedings Eighth Annual Conference of the Cognitive Science Society*, Amherst, MA (1986).
13. Kass, A. and Leake, D.B., Types of explanations, Tech. Rept. 523, Yale University, Department of Computer Science, New Haven, CT (1987).
14. Kedar-Cabelli, S.T., Purpose-directed analogy, in: *Proceedings Seventh Annual Conference of the Cognitive Science Society*, Irvine, CA (1985).
15. Kolodner, J.L., Retrieval and organizational strategies in conceptual memory: A computer model, Ph.D. Thesis, Tech. Rept. 187, Yale University, New Haven, CT (1980).

16. Kolodner, J., Simpson, R. and Sycara, K., A process model of case-based reasoning in problem solving, in: A. Joshi (Ed.), *Proceedings IJCAI-85*, Los Angeles, CA (1985) 284–290.
17. Leake, D.B. and Owens, C., Organizing memory for explanation, in: *Proceedings Eighth Annual Conference of the Cognitive Science Society*, Amherst, MA (1986).
18. Lebowitz, M., Generalization and memory in an integrated understanding system, Ph.D. Thesis, Tech. Rept. 186, Yale University, New Haven, CT (1980).
19. Lebowitz, M., Integrated learning: Controlling explanation, *Cognitive Sci.* **10** (1986) 219–240.
20. Mitchell, T.M., Generalization as search, *Artificial Intelligence* **18** (1982) 203–226.
21. Mitchell, T.M., Keller, R.M. and Kedar-Cabelli, S.T., Explanation-based generalization: A unifying view, *Mach. Learning* **1** (1986) 47–80.
22. Rieger, C., Conceptual memory and inference, in: R.C. Schank (Ed.), *Conceptual Information Processing* (North-Holland, Amsterdam, 1975).
23. Riesbeck, C.K., Failure-driven reminding for incremental learning, in: *Proceedings IJCAI-81*, Vancouver, BC (1981) 115–120.
24. Schank, R.C., *Dynamic Memory: A Theory of Learning in Computers and People* (Cambridge University Press, Cambridge, 1982).
25. Schank, R.C., The current state of AI: One man's opinion, *AI Mag.* **4** (1) (1983) 3–8.
26. Schank, R.C., Explanation: A first pass, Tech. Rept. 330, Yale University, Department of Computer Science, New Haven, CT (1984).
27. Schank, R.C., *Explanation Patterns: Understanding Mechanically and Creatively* (Erlbaum, Hillsdale, NJ, 1986).
28. Schank, R.C. and Abelson, R., *Scripts, Plans, Goals and Understanding* (Erlbaum, Hillsdale, NJ, 1977).
29. Schank, R.C., Collins, G. and Hunter, L., Transcending inductive category formation in learning, *Behav. Brain Sci.* **9** (4) (1986).
30. Schank, R.C. and Leake, D.B., Computer understanding and creativity, in: H.-J. Kugler (Ed.), *Information Processing 86* (Elsevier Science Publishers B.V. (North-Holland), Amsterdam, 1986) 335–341.
31. Simpson, R.L., A computer model of case-based reasoning in problem-solving: An investigation in the domain of dispute mediation, Ph.D. Thesis, School of Information and Computer Science, Georgia Institute of Technology, Atlanta, GA (1985).
32. Sycara, E.P., Resolving adversarial conflicts: An approach integrating case-based and analytic methods, Ph.D. Thesis, School of Information and Computer Science, Georgia Institute of Technology, Atlanta, GA (1987).