Representing abstract plan failures

Christopher Owens
The University of Chicago

An intelligent agent must be able to recover from and learn from its failures. This involves building a causal explanation of why the failure occurred and using that causal explanation as the basis of further reasoning about how to deal with the failure. This paper argues that the tasks of building the explanation and reasoning from the explanation should be tightly coupled, to avoid the problem of factually correct but pragmatically useless explanations. This integration can be accomplished by using a model of reasoning about plan failures that is based upon knowledge structures that link descriptions of stereotypical plan failures with descriptions of repair and recovery strategies appropriate to those failures.

1 Dealing with failures

An intelligent agent operating in an uncertain environment must constantly face the fact that its plans may fail. Knowledge about the world is incomplete, knowledge about the necessary preconditions for a particular course of action is likely to be underspecified, and the state of the world might change between the time a particular action is planned and the time it is executed.

While some failures are the result of circumstances that are genuinely beyond the control and predictive abilities of the agent, others result from planning errors which can be corrected if the agent understands the connection between the planning error and the failure. A widely-accepted paradigm for dealing with failed plans (see, for example, [Sussman, 75], [Hayes-Roth, 83], [Hammond, 86], [Birnbaum and Collins, 88]), is that an agent must:

- **Explain** the failure. Assign blame for the failure to some condition over which the agent could have had control. (Or, if no such condition can be found, identify the failure as an unforseeable, unavoidable one.)
- **Recover** from the failure. Plan some activity that will lead toward the original goals, taking into account the changed world resulting from the failure. Or, if achieving the original goal now looks too expensive, work on some other goal.

- **Repair** the plan that resulted in the failure. If the explanation assigns blame to some condition internal to the planner, for example failure to look for a certain contraindicating condition before commencing a particular activity, modify the plan so that future uses of the same plan will not result in the same failure.

It is clear that under this model, the steps of recovering from a failure and repairing the plan depend upon a good explanation – one that assigns blame to some state of the world or state of the planner's knowledge that is both responsible for the failure and within the power of the agent to change. It is also true, though less immediately clear, that whatever process builds the explanation of the plan failure must be informed by the system's knowledge of recovery and repair.

To understand this latter point, consider the example of a robot that, given the instruction to carry two chairs from one room to the next, picks up both chairs together and, in the process of trying to maneuver through a narrow doorway, damages the chairs. Some potential explanations of the failure are:

1. The doorway was too narrow.
2. The chairs were too large.
3. The robot is poor at judging the widths of passageways.
4. The robot did not know to re-estimate its size when carrying loads.
5. The robot knew to re-estimate its size, but is poor at it.
6. The robot did not know about the vulnerability of chairs to impact damage.
7. The robot chose the plan of carrying two chairs at once even though the plan was risky, because it considered the decrease in time worth the increased risk of damage (This might not be an error, but might in fact be a reasonable course of action if, for example, the room were on fire.)

Although each of these explanations refers to the same manifestation of a failure (i.e. dented chairs), the repairs and recoveries derived from each will be different. The first two explanations assign blame to states of the world that aren't reasonably within the robot's control. The recoveries or repairs they suggest (make the doorway larger, for example) are not feasible. The next four explanations deal with errors or omissions in the robot's knowledge about carrying objects, and the repairs they suggest involve modifying or adding to that knowledge. The final explanation deals with the mechanism the robot uses to mediate between competing goals, and it suggests modifying that mechanism.

Examining the difference between the first two explanations and the others points out that the goodness of an explanation depends not only upon the degree with which it faithfully captures some aspect of the causality leading up to the failure, but also upon the degree to which it suggests an operational repair. This dependency has implications for the role of memory in reasoning about failures.

2 Building explanations

There are basically two mechanisms whereby a system can build a useful explanation of a failed plan: "Build from scratch", which involves chaining together causal rules into an explanation that connects the unexpected failure with known conditions, and "Retrieve and apply", whereby the system maintains a library of abstract explanations that can be matched against the circumstances of a failure and instantiated with the particulars of the situation.

The "retrieve and apply" method of explanation construction is discussed in [Schank, 86] and [Kass *et al.*, 86]. The fundamental idea is that certain fairly complex patterns of causality tend to recur in the course of an agent's interaction with the world. These are not in principle different than the patterns that the system could recognize via the "build from scratch" approach, but the number of them that the system actually encounters is small relative to the total number of syntactically valid patterns that the system could build by chaining together its primitive causal rules. Caching these large and frequently-encountered causal patterns is not only useful from the point of view of efficiency, but also, as has been argued in [Kass and Owens, 88], from the point of view of likely utility in the face of incomplete knowledge.

In the context of reasoning about plan failures, there is a third advantage to the "retrieve-and-apply" approach to explanation, and this advantage pertains to the above-discussed question of how the system's notion of repair and recovery should guide its choices when it builds an explanation of a failed plan. The remainder of this paper deals with the representational form and content of a class of knowledge structure that addresses the task of understanding, recovering from and repairing failed plans.

3 What to represent

As has been argued in more detail in [Schank, 86] and [Owens, 88], there is an interesting correspondence between the type of stereotypical plan failure that an intelligent agent needs to represent, and common advice-giving proverbs and aphorisms. *The lazy man's load*, for example, refers to the failure resulting from trying to minimize the number of trips required to move some number of objects by moving all of them at one time. A more generally applicable pattern describing the same class of error might be the *too many irons in the fire* pattern, which deals with the general problem resulting from overdoing the optimization of combining tasks and executing them simultaneously.

These proverbs categorize situations in some useful way. All *too m y irons* situations share certain causal properties, for example that the failure is related to the fact that multiple tasks are being combined. It is generally easy for people to decide based on a *post facto* description, whether or not a particular situation is of the *too many irons in the fire* type. There is a common set of failure recovery strategies that should be successful in all *too many irons* situations, for example interleaving fewer tasks, putting some tasks on hold or finding someone to help. People are able to enumerate these strategies when asked about *too many irons* situations in the abstract. The goal for knowledge structures to be used as

part of an intelligent planning system, is to similarly categorize experiences into meaningful abstract classes or clusters.[1]

It is the observation that proverbs seem to link abstract failure descriptions with abstract recovery strategies that addresses the question of how the tasks of explaining, recovering from and repairing failures can be better integrated. They are done so via a knowledge structure that combines an abstract description of a plan failure with recovery and repair strategies. It is called a **plan failure explanation pattern**, or PFXP, and it is an extension and specialization of the **XP**s described in [Schank, 86] and [Kass *et al.*, 86].[2]

3.1 Understanding the failure

The central content of a PFXP is a template that can be used to build a causal model of the class of situation represented by that knowledge structure. The template characterizes, in abstract terms, the common causality that all situations covered by this PFXP share. In the case of the *Too many irons in the fire* situation, the causal pattern that characterizes the failure, when written out in English, looks something like:

> Simultaneously executing multiple tasks can fail if several of the tasks have time-critical steps, in that the agent will be busy attending to one time-critical step at the time that another time-critical step requires the agent's attention, or if the total workload of the tasks exceeds the agent's capabilities, in that the agent will be overloaded.

By deciding that this particular causal pattern applies to the current situation, the system makes several commitments about which features of the current situation will participate in the explanation. The fact that the object being moved is a chair, for example, is not involved in matching the situation against this pattern, and is consequently ignored, while the fact that the situation involved scaling the size of a task is highlighted. When it comes time to repair the problem, the system can ignore the former and focus on the latter.

3.2 Recovering from the failure

A recovery strategy, such as the decision to restart the plan using less interleaving of tasks, could be calculated dynamically from the causal model. Since the causal model shows that the interleaving is implicated in the plan failure, reformulating the plan to use less interleaving is not a particularly complicated inference. Other recovery strategies, like deciding to look for help, are not quite so easily and cheaply inferred from the initial causal explanation of the plan failure.

[1] See also [Lehnert, 81], [Schank, 82], [Dyer, 82] for a more general discussion of the relationship of proverbs to abstract thematic structures in the context of story understanding.

[2] See also the planning **TOP**s discussed in [Hammond, 86] for another abstract characterization of planning failures

In a system based on PFXPs, though, there is very little advantage to dynamically calculating repair strategies by inferring them from the causal model. Since the causal model is a static object stored as part of the knowledge structure, a set of potential repair strategies can also be packaged with a PFXP in memory. The system's task can therefore be to choose from among several possible recovery strategies rather than to try to build one from scratch. By comparison, in a system that builds explanations from scratch by chaining primitives together there is no obvious place to put recovery strategies in memory.

While the abstract strategies stored with a PFXP are static, the instantiation must of course be handled dynamically, since the abstract knowledge structure cannot know in advance the details of the situations to which it will be applied. The variables that are instantiated in the process of matching the causal model of the PFXP to the current situation, are carried over and used to instantiate the recovery strategy into a specific course of action.

The two recovery strategies mentioned above are representative of two distinct general classes of strategies. The first, re-executing with less interleaving, is basically an application of a general strategy that could be applied to any failure:

> *If some fact* **x** *is causally implicated in the failure, then cause* **x** *not to hold and try the plan again.*

While this strategy is general and simple, it is fraught with problems. A large number of conditions are causally implicated in the failure; yet many of them cannot or should not be changed by the planner, either because they are present in service of some goal or because they are beyond the control of the planner. This recovery strategy, given some of the explanations from the beginning of this paper, could easily generate the "solution" of making the doorway larger.

The second recovery strategy, looking for help, is much more specific to the *too many irons* failure. As a design principle, PFXPs should be set up with this more specific type of recovery strategy whenever possible. There is less inferential complexity involved in instantiating the strategy in the context of the failure, and because the inference chain is shorter, we can have more confidence that the strategy will be appropriate.

3.3 Identifying the bad planning decision

The role of a PFXP in allowing a system to correct its inappropriate planning behavior is that it provides a pointer to a planning decision believed likely to be the one responsible for the failure characterized by this particular PFXP. Often the decision pointed to will be a plan transformation. In the case of the *Too many irons* PFXP, the bad decision suggested looks something like:

> *Enhance this plan by simultaneously performing similar steps*

The type of bad decision that is pointed to by a PFXP can take several forms, among them:

Spurious action Some action was taken that caused the plan to fail; had it not been taken the plan would have succeeded. This is the type of failure that the *too many irons* PFXP embodies.

Omitted action Some action could have been taken to prevent this particular failure, but was not taken. This is the type of failure implicated in, for example, the PFXP corresponding to the proverb *A stitch in time saves nine*, which typifies failures in which some low-cost preventive measure would have avoided a high-cost outcome.

Inappropriate plan choice The choice of plan to accomplish some particular goal or subgoal was inappropriate. This type of bad decision can be referred to by PFXPs such as the one corresponding to the proverb *You can catch more flies with honey than you can with vinegar.*

Inappropriate resource choice The choice of resource with which to implement some plan was inappropriate. An example of this can be found in a PFXP corresponding to some interpretations of the proverb *You can't make a silk purse out of a sow's ear* or *A handsaw is a good thing, but not to shave with* – each of which warns against applying a resource to some task for which it is manifestly unsuited.

Ignored factor Some factor was not taken into account that, if it had been taken into account, would have avoided the failure. An extremely general instance of this failure is found in the *Look before you leap* PFXP.

Spurious factor Some factor was considered that, if it had been ignored, would have avoided the failure. The system paid too much attention to an insignificant or irrelevant factor. A number of decisions related to risk-taking fall into this category, such as those implicated in the proverb *He who waits for a fair wind misses many a voyage*

Inappropriate goal prioritization Some goal was given either excess or insufficient consideration, relative to the other goals on which the system was working at the time. This corresponds to the final explanation for the damaged chairs suggested at the beginning of this paper.

Incorrect assumption From the point of view of credit and blame attribution, this type of bad decision is different from the others in that it isn't really a decision that is pointed to here – the system didn't do anything wrong that led it into the failure, but rather it relied on information that is not correct. The **bad decision** field of this type of PFXP points to the offending assumption.

Some of these descriptions are oversimplified here. For example, **spurious factor**, **ignored factor**, **spurious action** and **omitted action** need not be boolean descriptors as suggested above. In fact, this type of error is much more often a matter of degree than it is of absolutes. The description of a bad decision is less likely to be something like

You paid attention to something you should have ignored

than it is to be something like

You paid more attention to this factor than you should have.

The basic idea behind encoding a bad decision as part of the representation of a PFXP is analogous to the credit assignment function of the basic causal model of the PFXP. Just as the basic causal model attempts to focus credit assignment on aspects of the world that are potentially under the planner's control, the **bad decision** field provides a means to focus internal credit assignment on aspects of the system's planning behavior that are modifiable.

A problem with this kind of credit assignment is that there is no guarantee of accuracy. The particular plan transformation or other decision pointed to is neither necessary nor sufficient as an explanation of the failure. Clearly the decision to interleave tasks does not always result in a bad outcome. Generally, in fact, it is a desirable thing to do. Furthermore, and more importantly from the point of view of constructing explanations, not all instances of the *Too many irons* situation result from an explicit decision to interleave tasks. Sometimes the situation might result from, for example, failing to notice a problem developing as more and more situations requiring immediate attention arise. Or, the situation might result from the conjunction of several seemingly unrelated planning decisions. The contents of the **bad decision** field are of heuristic value rather than of provable correctness.

3.4 Modifying the bad planning decision

Just as each description of a failure in the world has associated with it a recovery strategy, likewise each description of a bad planning decision has associated with it a repair strategy. The role of a PFXP in repair is, however, less clear-cut than it is in the area of developing a causal explanation or recovering from specific failures. This is partially due to the fact that internal credit attribution is not so clear-cut as external credit assignment. In the case of the robot moving the chairs, what, exactly, is the bad decision that resulted in the damage? Was it the decision to carry two chairs at once? To move quickly through the door? Or was it the result of placing too much importance on getting the job done quickly, or not enough importance on preserving the chairs.

If, a system constantly encounters errors of the same type, say *too many irons*, it should be less willing to perform the plan transformation that says to enhance a plan by allocating multiple agents to work on it. If, on the other hand, this type of error never occurs, that is a sign that the system is probably missing opportunities to enhance plans by being too conservative – it should be less fussy about applying that particular plan transformation. Ideally, the way to make the system more fussy about a particular plan transformation is to add preconditions for that transformation, and the way to make it less fussy is to drop preconditions. A very crude approximation to this behavior that requires much less knowledge on the part of the system, is to adjust some numerical parameter corresponding to the likelihood of a particular plan transformation being applied based upon the number of errors encountered that point to that transformation. While the latter approach is easier to implement, it does not allow the kind of learning enabled by the former.

4 Conclusions

Reasoning about failed plans by using an approach based upon matching large causal structures is more than just an efficiency issue related to the compilation of rules into larger rules with more detailed applicability conditions. The approach addresses the question of how to more closely link the task of understanding a plan failure with the task of acting based upon that understanding. Because the only knowledge structures available for matching are those that are linked to operational repair and recovery strategies, the system avoids the problem of generating factually correct but pragmatically useless explanations.

References

[Birnbaum and Collins, 88] L. Birnbaum and G. Collins. The transfer of experience across planning domains through the acquisition of abstract strategies. In J. Kolodner, editor, *Proceedings of a workshop on Case-based Reasoning*, pages 61–79, Palo Alto, 1988. Defense Advanced Research Projects Agency, Morgan Kauffmann.

[Dyer, 82] M. Dyer. In-depth understanding: A computer model of integrated processing for narrative comprehension. Technical Report 219, Yale University Department of Computer Science, May 1982.

[Hammond, 86] K. Hammond. *Case-based Planning: An Integrated Theory of Planning, Learning and Memory*. PhD thesis, Yale University, 1986. Technical Report 488.

[Hayes-Roth, 83] F. Hayes-Roth. Using proofs and refutations to learn from experience. In R. Michalski, J. Carbonell, and T. Mitchell, editors, *Machine Learning: An Artificial Intelligence Approach*, pages 221–240. Tioga, Palo Alto, 1983.

[Kass and Owens, 88] A. Kass and C. Owens. Learning new explanations by incremental adaptation. In *Proceedings of the 1988 AAAI Spring Symposium on Explanation-Based Learning*. AAAI, 1988.

[Kass *et al.*, 86] A. M. Kass, D. B. Leake, and C. C. Owens. SWALE: A program that explains. In *Explanation Patterns: Understanding Mechanically and Creatively*, pages 232–254. Lawrence Erlbaum Associates, Hillsdale, NJ, 1986.

[Lehnert, 81] W. Lehnert. Plot units and narrative summarization. *Cognitive Science*, 5:293–331, 1981.

[Owens, 88] 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.

[Schank, 82] R. Schank. *Dynamic Memory: A Theory of Learning in Computers and People*. Cambridge University Press, 1982.

[Schank, 86] R. Schank. *Explanation Patterns: Understanding Mechanically and Creatively*. Lawrence Erlbaum Associates, Hillsdale, NJ, 1986.

[Sussman, 75] G. Sussman. *A computer model of skill acquisition*, volume 1 of *Artificial Intelligence Series*. American Elsevier, New York, 1975.