A Functional Taxonomy of Abstract Plan Failures

Christopher Owens
The University of Chicago, Department of Computer Science
1100 East 58th Street, Chicago, IL 60637
E-mail: owens@gargoyle.uchicago.edu

Abstract

To reason about a plan failure requires an appropriate explanation of the failure. Such explanations, as has been argued elsewhere, can be generated by instantiating and adapting abstract knowledge structures. But the two fundamental goals in reasoning about failure, *recovering* from the failure and *repairing* the knowledge that led to the failure, require qualitatively different types of explanation. This paper presents an abstract taxonomization of failures, oriented toward the latter problem. Each abstract failure type in the taxonomy is tied to heuristics for recognizing occurrences of failures of that type and for identifying the knowledge that can be repaired to avoid future occurrences of the same failure.

Failures and explanations

Autonomous agents operating in complex environments will inevitably experience failures. A robot vehicle, for example, may find an expected fuel cache missing, run low on fuel, and therefore be unable to continue operation.

A standard approach to handling failure (see, for example, [Sussman, 75], [Hayes-Roth, 83], [Hammond, 89], [Birnbaum and Collins, 88]) dictates that an agent experiencing a failure must both **recover** from the immediate failure, and **repair** its knowledge so as to avoid similar situations in the future. Recovery and repair are predicated upon building a good causal **explanation** of the failure.

For the purposes of reasoning about failure, an explanation is a pattern of inference that causally connects the anomalous experience to previously known facts. A good explanation is one that correctly assigns blame for the failure, and that assigns blame to some condition under the planner's control, therby leading to an appropriate recovery strategy. If the autonomous vehicle, for example, explains its predicament as being due to its own poor navigation and inability to find the cache, the appropriate recovery may involve gathering more position information. If the explanation posits that the fuel cache is actually missing, for example having been stolen, then the appropriate recovery involves obtaining fuel from some other source (turning back, calling for assistance).

Similarly, an appropriate explanation can lead to an appropriate repair. The robot, for example, could explain its current predicament in terms of a prior decision not to take on extra fuel, which was made with the expectation that the fuel cache would be present and easy to find. The explanation implicates the agent's reasoning about the persistence of fuel caches ("I should have known someone might have stolen the fuel."), its predictions of its own navigation abilities ("I should have realized it would be hard to find the fuel."), and its reasoning about resource reserves ("I should have taken on a full tank when it was available.").

Memory-based explanation

An obvious way to build an explanation is to chain through a space of explanatory inference rules so as to connect the anomalous experience to known facts. Work on memory-based explanation (e.g. [Schank, 86],[Kass, 90], [Leake, 90], [Owens, 90a]) has argued that a better way to build explanations is to retrieve, instantiate, and adapt knowledge structures that characterize familiar, recurring patterns of causality. These knowledge structures are referred to as **explanation patterns**, or XPs. One advantage of explanation by adaptation and instantiation is that, given an incomplete and imperfect domain theory, explanations derived by incremental modification of known valid causal patterns are more likely to be useful than explanations built by chaining through the space of primitive rules. Another advantage is that storing successful explanations for re-use provides a natural learning mechanism.[1]

Memory, recovery, and repair

In [Owens, 90a], and [Owens, 90b] I present an additional argument in favor of memory-based explanation. I argue that abstract explanatory knowledge structures known as **plan failure explanation patterns**, or

[1]These arguments are made in more detail in [Kass and Owens, 88].

PFXPs, can play a direct role in both recovery and repair. For each known category of failures, the corresponding PFXP provides a place in memory to store repair and recovery strategies appropriate to that class of failure. Instead of dynamically calculating a repair or recovery strategy from the explanation, a pre-existing strategy, stored in memory with the causal pattern, can be instantiated and used. For example, if a PFXP stores the causal pattern describing situations in which some needed object is not found where expected, it can package the recovery strategies:

- Gather more information about your current position.
- Gather more information about the location of the needed object.
- Find an alternate way to obtain the object.

A key question associated with such explanatory patterns is *indexing*: how should structures be labeled, and what features should the system use to describe the world. A step towards the development of labeling features is to taxonomize the types of explanations the system will be required to make. The appropriate basis for such a taxonomy is functional: the system should discriminate between different abstract patterns of causality occurring in the world, to the degree that different patterns of causality suggest different recovery and repair strategies.

Taxonomizing for repair

Recovery depends upon the concrete particulars of the situation. Failures involving a resource shortfall suggest one class of recovery strategies; failures involving being in an inappropriate location dictate another; failures involving lacking some piece of information, a third. The appropriate explanation for selecting a recovery strategy is one that attributes blame to some condition in the world that is under the planner's control.

Repair, on the other hand, requires an explanation that ties the current failure to some inappropriate decision made by the planner, and which in turn ties that inappropriate decision to either a missing piece of knowledge or an inappropriate control strategy. Once the appropriate missing knowledge or inappropriate control strategy has been identified, the planner can learn by acquiring the knowledge or modifying the strategy.

The remainder of this paper outlines a taxonomy of abstract descriptions of plan failures. These abstract descriptions are presented at a domain-independent, but task-specific level of abstraction, and are oriented toward repair rather than recovery. This particular taxonomization breaks failures into **goal-related**, **task-related**, **resource-related**, and **time-related** failures. Each failure type is briefly described, along with strategies for recognizing and repairing failures of that type.

Goal-related failures

Running out of fuel is on the surface a resource management problem, but not necessarily. It can also be the presenting symptom for goal-related problems. The robot, for example, may have failed to generate the goal of preserving its fuel supply, or it may have inappropriately let another goal take precedence. Agents mismanage goals by neglecting goals, by pursuing irrelevant goals, or by inappropriately mediating among conflicting goals. Many of the goal relationships described in [Schank and Abelson, 77] and [Wilensky, 83] can be useful in building goal-related explanations of failures, and it is in the context of that work that the the goal-related failure explanations below are described.

Neglected goals

The simplest goal-related explanation for the out-of-fuel situation is that the agent did not have the goal of preserving its fuel supply. This could reflect such underlying flaws in the agent's knowledge as:

- Having no knowledge about the consequences of having no fuel.
- Failing to infer a threat to the goal of preserving fuel.

Recognizing neglected goal failures Recognizing a failure resulting from a totally absent goal is difficult. Since the vehicle does not have the goal of preserving its fuel supply, it will never notice nor reason about threats to that goal. A person who doesn't know that automobiles require gasoline, will be at a loss to explain why his car suddenly stops working. Only once an explanation is obtained — either by consulting an external knowledge source or, less likely, by reasoning from known principles, can the system build the causal connection between having an adequate fuel supply and being able to perform operations. At that point it can recognize that preserving a fuel supply is goal that should have been present.

If the agent has the goal of preserving its fuel supply but does not correctly infer a threat to that goal, the failure is easier to detect simply because the agent already has a goal structure in place against which to explain the failure: "My goal of preserving my fuel supply has been violated, and I have no record of having reasoned about any threats to that goal." Detecting this class of neglected goal failures requires maintaining a trace of reasoning about threats to goals, indexed by the goal that was threatened.

Repairing plans that neglect goals If a failure results from a system totally lacking a certain goal, then the obvious repair is to add that goal. But goals, in their detailed, specific instantiations, are not static objects in a system's memory. If one wanted to add, as a static entity, the goal to preserve one's fuel supply when operating in uncertain territory, it is not clear where in memory this static entity should reside or how it should be activated at the appropriate time.

If, on the other hand, a system explains its failure in terms of failing to notice a threat to a goal, the repair involves building a detailed causal explanation of the failure, and using that explanation to learn features that are predictive of the threat.

Inappropriate goal priorities

More common goal-related failures involve assigning inappropriate priorities to conflicting goals. Many goal prioritization failures result from inappropriately resolving **goal conflict** or other goal interrelationships.

For example, the robot vehicle might have placed more emphasis on completing its mission quickly than on preserving its fuel supply, and therefore decided not to take on additional fuel from a known supply and chance the availability of the fuel cache.

Detecting goal prioritization errors

Just as the system must keep a trace, indexed by the involved goals, of all reasoning about inferred threats to goals, it must also keep a trace of all goal conflict resolutions if it is to detect goal prioritization errors. If a plan fails and the initial explanation relates the failure to a violated goal, and the system has a record of having mediated a conflict between that goal and some other goal, then the goal mediation is suspect.

Repairing goal prioritization errors Explicit reasoning about conflicting goals is often expressed in terms of **costs** — paying too high a price to obtain some goal, or wrongly failing to undertake a plan when because the cost was (erroneously) considered too high.

But characterizing an error in terms of, for example, *being too conservative about costs*, does not repair the incorrect knowledge that caused the error. If the priorities of two goals are inappropriate (e.g. if the vehicle inappropriately rates the time necessary to take on extra fuel as more costly than the consequences of running out of fuel far from home, then the repair is to change the priorities of those goals. But one cannot simply find the "preserve fuel" goal and increase its weight, because there is no preserve fuel goal except when the goal is dynamically spawned by the interaction of knowledge about fuel supply and the knowledge about the likely fuel utilization of the current mission. If a failure results from goal conflicts being resolved inappropriately, than the repair must look at and modify the source of the goals.

Task-related failures

The second major category of planning failures captures the errors that arise from choosing an inappropriate action in service of some goal, from pursuing an otherwise appropriate course of action either too vigorously or not vigorously enough, from wrongly combining two actions into a single plan, and from splitting a task into subtasks inappropriately.

Task selection

Task selection errors involve choosing an action that is inappropriate either because it fails to achieve its goal or because, despite achieving the goal, it has an unacceptable side effect. A task-selection explanation of the out-of-fuel situation would be that searching for a fuel cache was an inappropriate task in service of the goal of obtaining fuel.

Although the action turned out to be inappropriate, the agent had good reason to expect the approach to work: searching for a cache is a known plan for obtaining fuel. It just didn't work here — either because another agent took the fuel or because the location of the fuel cache was not as well known as had been believed.

Recognizing task selection failures Whenever an action fails to achieve its intended goal, by definition the action was an inappropriate choice given the state of the world prior to undertaking that action.[2] The question, from the point of view of repair, is whether the choice was inappropriate given the **known** or **inferrable** state of the world. Certain conditions can strengthen the planner's belief that the choice of actions was faulty, the most significant being the planner's knowledge that it made a difficult choice. Just as with the traces of goal mediation described above, the agent can keep a trace of all decisions involving substitution of one action for another, indexed by the action that was the final result of the choice. For example, if the standard plans for obtaining fuel are to take on extra at a known supply point or to rely on a cache, and the system made a close call in choosing between them, that choice becomes suspect in building a repair-oriented explanation of the failure. Keeping track of selections that were close calls can be integrated into the framework for saving the justifications for decisions described in [Carbonell, 83] and [Carbonell and Veloso, 88].

Repairing task selection errors Merely recognizing that the planner chose an inappropriate action is not a functionally useful explanation because it does not suggest a repair to the planner's knowledge. Any of several causes might underlie the inappropriate choice of actions:

1. It was the only action known to satisfy that goal.
2. It was one of several actions known to satisfy that goal, but it was the only one without known unacceptable side-effects.
3. It was one of several apparently equally appropriate actions known to satisfy this goal, and the planner chose arbitrarily.
4. There was no action known to satisfy this goal, but the planner selected this action because it is known to satisfy a similar goal.

The repair consists of augmenting or refining those specifications. The planner can, for example, repair its

[2]Barring, of course, the explanation that the failure resulted from some unforseeable fluke beyond the control of the agent.

knowledge of the domain by adding new side-effects or constraints to the description of a particular action, such as "Searching for a cache is unreliable if the territory is known to be used heavily by other agents, or if the description of the location of the cache is known to be approximate."

Task scaling

A different class of task-related failures results from pursuing a course of action too vigorously or from not pursuing it vigorously enough. For a robot vehicle, a clear example of performing an action too intensely is traveling too quickly. The actual failure can take the form either of a bad side effect of the action that would not be a problem at a reduced degree of intensity (running off the road) or from the waste of resources that is entailed.

Failing from insufficient intensity is a more obvious type of failure. The critical question in this latter class of situation is **why** the agent failed to perform the action with sufficient intensity — whether the required level of intensity was beyond the agent's capabilities, whether the agent did not assess the required level correctly, or whether the agent was overly conservative about a potential side effect.

Recognizing task scaling failures There are two different kinds of situations in which task scaling failures occur. In the first, the system made an explicit decision about the degree of intensity with which to pursue the task (such as deciding to travel faster than the "normal"[3] speed in order to accomplish some mission more quickly). In the second, the original "normal" level of intensity with which the action was performed was inappropriate — for example the normal rate of travel for a given road is inappropriate if it is snowing.

In the former case, with an explicit decision to modify the level of intensity, that decision can be remembered and used as a starting point for explanations of failures involving that task. In the latter case, failures are difficult to notice. "Overload" failures resulting from increasing the intensity of a particular action (e.g. running off the road from excessive speed) tend to have distinctive presenting symptoms, but failures that result from not performing an action intensely enough do not have their own set of presenting symptoms.

Repairing task scaling errors Obviously the repair to a plan causing a task scaling failure is to change the intensity with which the task is pursued. But, as was the case with goal prioritization errors, that dodges the issue of knowing **when** the change should be applied. If the vehicle runs off the road, does that mean that the vehicle should travel more slowly in general? Probably not.

A noteworthy aspect of the failures resulting from performing an action too intensely is that they result from taking a good plan (a travel plan), and applying a good plan transformation to it (get the job done faster by traveling faster.) Despite the desirable plan transformation strategy, the resulting, transformed plan, is inappropriate. The problem is analogous to the problem with task selection: either that the applicability conditions for that plan transformation were not well enough defined, or that the description of the current situation was not rich enough. The repair is to learn features that make sharper discriminations between situations in which the transformation is applicable and those in which it is not.

Task combination and splitting

The final class of task-related planning failures involves splitting and combining tasks. Several transformations that are normally desirable can have bad results if applied indiscriminately. These are:

- Make maximum use of your time by working on more than one task at a time. When this fails, symptoms include missing deadlines and failing to complete subtasks because one lacks the capacity. The failure comes from the system's inability to predict the interaction of the two tasks; the repair is to enrich the descriptions of each of the tasks involved so that bad interactions can be more easily predicted.
- Subdivide a task and work on part of it at a time such that the entire task is done more slowly. Not all tasks can be subdivided because some tasks require continuity. One cannot, for example, strip the old paint off a house one season and paint it the next, because the intervening weather will destroy the exposed surfaces. Again, this failure comes from an inadequate representation of the interaction of the two tasks.

Resource-related failures

The third major category of planning failures reflects to errors in resource management. Once a planner has identified a worthwhile goal and chosen a plan to satisfy this goal, errors arise in choosing the resources and the scale of the resources that will be used in service of this plan.

Resource selection

There are two fundamental resource selection failures; they are analogous to the task selection failures described above. The first, **inappropriate resource failure**, occurs when an agent uses an inappropriate resource in service of an otherwise acceptable plan. The failure results from inadequate representations of the resources involved and inadequate specifications of the relationship of the resources to the task. If its knowledge about fuel and vehicles were underspecified, the robot vehicle, unable to obtain its usual fuel (diesel fuel), might inappropriately apply the transformation **substitute similar resource** and generate a plan to fill its tanks with some other kind of fuel (gasoline), or,

[3] A great deal has been swept under the representational rug here.

if its knowledge were even less rich, with some other liquid that is put in tanks in other situations (water). The problem is that the system's representations of its fuel needs, of gasoline, and of diesel fuel were not sufficiently rich.

The second, **resource substitution conservatism** occurs when an agent fails to recognize that one resource is an acceptable substitute for another. For example, the vehicle might have avoided filling its tanks with an acceptable substitute (winterized diesel fuel) if its normal fuel (straight diesel fuel) were unavailable.

Recognizing resource selection failures Resource selection failures are somewhat difficult to separate from task selection failures. If an agent chooses a particular task and a resource to implement that task and the task fails, it is difficult to say whether the task or the resource was inappropriate.

One heuristic solution is to look at which choice (action or resource) was more questionable at the time it was made (e.g. which involved a substitution or second choice, or which was decided arbitrarily). If the choice of tasks was less problematic than the choice of resources, then any subsequent plan failure can try to implicate the resource choice as a possible explanation. As with task selection, resource selection failures involving inappropriate conservatism are difficult to detect.

Repairing resource selection errors In a manner analogous to task selection errors, the repair of a resource selection error depends upon an explanation of why the system made that selection:

- If the resource was the only possible choice (if no substitute was available) there is no repair.
- If the resource was chosen arbitrarily from among several apparently equal possibilities, then the refine the description of the resource. For example, if gasoline and diesel fuel are both described as fuel for running engines, and if the system failed because it chose gasoline to fill the tanks of a diesel vehicle, then the selection criteria for fueling need to be modified to reflect vehicle engine type as a discriminator.
- If the resource was chosen as a substitute for another unavailable resource, then the substitution procedure needs to be modified.

Resource scaling

Resource shortfalls provide a a good example of how explaining toward repair differs from explaining toward recovery. If a vehicle is low on fuel, the recovery strategies involve either finding more fuel or altering the operation so as to reduce the rate of fuel consumption. The repair strategies, however, require a different kind of explanation:

- The agent might have mis-estimated the amount of the resource necessary to complete the task, either by mis-estimating the scale of the task or by mis-estimating the rate at which the task consumes the resource.
- The agent might have mis-estimated the available stock of resource before commencing the task.
- The agent might have started the task without enough of the resource on hand, under the mistaken assumption that more would be available.

Recognizing resource scaling failures A resource shortfall is easy to detect. When a system goes to use a particular resource and finds it unavailable, the current plan will block. This condition will be the symptom by which the failure is made known to the planner. Unfortunately, as seen above, a resource shortfall can often be the symptom for other failures as well.

A resource surplus is more difficult to detect because there is no reason for a system to check its supply of a resource that it is not currently using. A resource surplus can be detected if it is explicitly searched for (when finishing a trip, check to see if there is excessive reserve fuel) but it does not present itself as a failure as does a resource shortfall.

Repairing resource scaling errors Repairing resource scaling errors is very much like repairing task scaling errors: the obvious repair is to apply more or less of this resource to the task next time. Analogously to task scaling repairs, there is a problem with situating the repair at the appropriate level of generality. If, for example, the vehicle finds that it did not take on enough fuel to complete the full range of planned activities, it should not always take on more fuel in the future. The system must explain why it predicted the required amount of the resource incorrectly, and it must modify the erroneous knowledge that led to that prediction.

Time-related failures

The fourth major category of plan failures relates broadly to to time. Along with scaling errors involving decisions about how much time to spend on a particular task (which look very much like resource scaling errors), time-related errors include errors resulting from decisions about task scheduling and task ordering.

Task ordering

"Goal Clobbers Brother Goal" and related characterizations in [Sussman, 75] and [Sacerdoti, 77] are the prototypical **task ordering** failures. Two specializations of task ordering failures are *failure to establish precondition* and *prematurely establishing precondition*. They describe the potential results of decisions about when to schedule the task that will establish a precondition for some larger goal. The system can establish this precondition as soon as possible, which guarantees that the precondition will be enabled in time to satisfy the larger goal but which risks wasting resources — establishing that precondition for nothing if the larger goal is deferred. On the other hand, the system can defer enabling a precondition until it becomes necessary, which avoids the risk of wasting the effort, but which risks the precondition not being enabled in time to meet the need..

Recognizing task ordering failures Failure to establish a precondition is its own presenting symptom. Consider, for example, arriving at a door and being unable to open it because one lacks a key. This failure can be a symptom of an improper task ordering, e.g. "I should have stopped by the hardware store to pick up the keys that were being copied *before* going home for lunch rather than on the way back to work." When the system is blocked because some precondition has not been met, it can check to see if it ever explicitly planned to achieve that precondition. If so, then the failure resulted from having scheduled that plan inappropriately. If, on the other hand, the system never planned to achieve the precondition, the problem is not one of task ordering but rather of a neglected goal described above.

Repairing task ordering errors As with other failure types, the correct repair to a task ordering error depends upon how the tasks came to be misordered. For example, if the tasks are mis-scheduled because some transformation put them out of order (see, e.g. [Collins, 87] for a taxonomy of task reordering transformations), then the transformation was insensitive to the dependence of one of the tasks on the other. As with task selection and resource selection errors, the problem is either that the descriptions of the tasks being reordered is underspecified, or that the applicability criteria for the re-ordering transformation is underspecified. The repair is to add discriminating features to either of these descriptions.

Other scheduling errors Other types of scheduling errors include making plans contingent upon an event whose occurrence has not been ascertained (counting on the availability of the fuel cache), for which the repair is to patch the knowledge used to predict the likelihood of the event.

Conclusions

The purpose behind taxonomizing plan failures is that certain abstractly defined failures tend to recur. If a system possesses a canonical set of abstract descriptions of plan failures, it can use those descriptions to recognize occurrences of familiar failures, and can use the recognition as a basis for selecting appropriate recovery and repair strategies. Accordingly, the appropriate basis for taxonomizing plan failures is functional: it must be informed by a model of recovery and repair.

Examples of planning failures are important data that can serve two purposes. Using recovery as a functional basis for taxonomization can drive the development of domain-level knowledge about the relationship between an agent, its actions, and the environment. Using repair as a functional basis, on the other hand, can drive the development of a domain-independent model of the agent's interaction with its own knowledge.

References

[Birnbaum and Collins, 88] L. Birnbaum and G. Collins. The transfer of experience across planning domains through the acquisition of abstract strategies. *Proceedings of a workshop on Case-based Reasoning*, Palo Alto, 1988. Morgan Kauffmann.

[Carbonell and Veloso, 88] J. Carbonell and M. Veloso. Integrating derivational analogy into a general problem solving architecture. *Proceedings of a Workshop on Case-Based Reasoning*, Palo Alto, 1988. Morgan Kaufmann, Inc.

[Carbonell, 83] J. Carbonell. Derivational analogy and its role in problem solving. In *aaai83*, Washington, DC, August 1983. AAAI.

[Collins, 87] G. Collins. *Plan creation: using strategies as blueprints*. PhD thesis, Yale University, 1987. Tech Report 599.

[Hammond, 89] K. Hammond. *Case-Based Planning: Viewing Planning as a Memory Task*, volume 1 of *Perspectives in Artificial Intelligence*. Academic Press, San Diego, CA, 1989.

[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, 90] A. Kass. Developing creative hypothesis by adapting explanations. Technical Report 6, Institute for the Learning Sciences, Northwestern University, 1990.

[Leake, 90] D. Leake. *Evaluating Explanations*. PhD thesis, Yale University, 1990. Technical Report 769.

[Owens, 90a] C. Owens. *Indexing and Retrieving Abstract Planning Knowledge*. PhD thesis, Yale University, Department of Computer Science, 1990. In preparation.

[Owens, 90b] C. Owens. Representing abstract plan failures. In *Proceedings of the Twelfth Annual Conference of the Cognitive Science Society*. CSS [?], 1990.

[Sacerdoti, 77] E. Sacerdoti. Language access to distributed data with error recovery. In *Proceedings of the Fifth International Joint Conference on Artificial Intelligence*, Cambridge, Mass., August 1977. IJCAI.

[Schank and Abelson, 77] R. C. Schank and R. Abelson. *Scripts, Plans, Goals and Understanding*. Lawrence Erlbaum Associates, Hillsdale, New Jersey, 1977.

[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.

[Wilensky, 83] R. Wilensky. *Planning and Understanding: A Computational Approach to Human Reasoning*. Addison-Wesley Publishing Company, Reading, Massachusetts, Reading, Mass, 1983.