Learning to Anticipate and Avoid Planning Problems through the Explanation of Failures*

Kristian J. Hammond

Department of Computer Science
Yale University

ABSTRACT

This paper presents an approach to learning during planning that focuses on learning to predict planning problems through an analysis of the planner's own failures. The need to predict failures in order to avoid them is argued and a method for learning the features that predict problems from a causal analysis of planning failures is discussed. A further argument is also given concerning the natural integration of this approach to learning with an overall theory of case-based planning.

An implementation of these learning ideas is presented in the case-based planner CHEF, which creates new plans from old in the domain of Szechwan cooking. The CHEF planner uses an *anticipate and avoid* approach to planning problems that is sharply contrasted with the *create and debug* approach taken by existing planners.

I LEARNING FROM PLANNING

In recent years the study of machine learning has moved away from simple concept learning towards the idea of learning in the context of other tasks. Many researchers have proposed learning systems that work with planners to store the results of the planner's own efforts [1,3,4] or the refined versions of a tutor's examples [2]. The stress in all of these systems has been on the learning of new plans for later use. The results reported by researchers developing such systems has centered around planning-time gains that result from reusing the stored plans.

There is, however, another aspect of learning that these systems do not address: learning to avoid failures that the planner has encountered before. Because existing systems focus on the final result of the planning process and not the errors that were made along the path to that result, they do nothing to ensure that those errors will not be made again. While these systems store plans in terms of the goals that they may satisfy, any problems that the planner may have encountered and repaired are ignored. Goals or goal combinations that have proved to be problematic are not noted, meaning that they will be handled in the same fashion time after time, even if that handling always leads to a failure.

A different approach to learning from planning is to learn to recognize the situations in which failures occur and use that recognition to avoid them. One way to do this is to design a planner that uses its own failures to learn problematic features in a domain. These problematic features can then be used to predict problems in later planning situations so that the planner can construct its new plan knowing the problems it must avoid. By also storing plans in terms of the problems that were encountered while building them, a planner could use the prediction of a problem to find a plan in memory that avoids it.

This idea of using failures to learn the features that predict them is implemented in the case-based planner CHEF, that creates new plans in the form of recipes in the domain of Szechwan cooking. The CHEF planner uses an ***anticipate and avoid*** approach to planning problems that is sharply contrasted with the *create and debug* approach taken by existing planners. CHEF attempts to predict and plan for possible failures before they actually occur rather than waiting for them to happen and repairing them once they have. The ability to learn from its own failures allows the CHEF planner to anticipate and avoid those problems that it has seen before.

II THE NEED FOR ANTICIPATION

One of the persistent problems that planners have to deal with is the changes that take place in the world as a result of their own actions. When a planner has to plan for a set of goals from a given state of the world it is possible for it to construct a plan for one goal that will change the conditions that are required for planning for another. Goal and plan interaction has been the subject of much work in Artificial Intelligence [5,7,8] but has always been handled from the approach of dealing with the planning failures they arise. This has resulted in a class of planners that are able to debug faulty plans but only *after* the faults have arisen.

An alternative approach to planning failures is to anticipate them before they arise. Once anticipated, a failure can be avoided by finding a plan that deals with the problem indicated by the prediction while also satisfying the planner's current goals.

The CHEF planner anticipates problems that it has previously encountered, using links that it builds at the time of a failure from the features that caused it to the memory of the failure itself. When the features reoccur in later circumstances, the planner is reminded of the past experience and this reminding serves as a warning to the planner that it has to plan for the fact that this failure is going to occur again. Because CHEF

*This report describes work done in the Department of Computer Science at Yale University. It was supported in part by ONR Grant #N00014-85-K-0108.

also stores the repaired plans that were built in response to past failures, indexed by the fact that they solve the problem corresponding to the failure, CHEF is able to use the prediction of the problem to find a plan that avoids it.

For example, in trying to create a plan for a strawberry soufflé, CHEF encounters problems with the liquid added by the addition of chopped strawberries to the soufflé batter. This added liquid causes an imbalance in the relationship between liquid and leavening in the recipe. This alters the condition which was required for the soufflé to rise, which results in a fallen soufflé.

CHEF is able to recover from this failure by adding more egg white to the recipe. But because it does not want to repeat the failure later on, it has to change more than the current plan. It also has to change the way in which it will plan for similar circumstances at a later date. So CHEF does two things. First it stores the new plan in memory, indexed by the fact that it is a plan to deal with the problem of added liquid in a soufflé recipe. But indexing a plan by the fact that is solves a problem is of no use unless the problem is anticipated. So CHEF also builds links from the features in the situation that caused the problem (added fruit or extra liquid) to a memory of the failure itself. With these links in place, it is later able to infer the occurrence of the failure from the reoccurrence of the features that participated in causing it earlier.

In dealing with a later situation, in which CHEF is planning for a soufflé with the liqueur kirsch, it is reminded of the past failure. This tells CHEF that it is in a problem situation and it adds the goal to avoid the problem to the list of goals used to search for an initial plan. It then finds the strawberry soufflé recipe, with the added egg white, and modifies it to include kirsch rather than strawberries. Without the anticipation of the failure, CHEF would have used a more basic vanilla soufflé recipe and would have built a plan with the same flaw as in the failed strawberry soufflé plan. By anticipating the failure, CHEF is able to find a plan that avoids it.

III AN OVERVIEW OF CHEF

CHEF's input is a set of goals for different tastes, textures, ingredients and types of dishes and its output is a single recipe that satisfies all of its goals. Its basic algorithm is to find a past plan that satisfies as many of the most important goals as possible and then modify that plan to satisfy the other goals as well.

Before searching for a plan to modify, CHEF examines the goals in its input and predicts any failures that might rise out the interactions between the plans for satisfying them. If a failure is predicted, CHEF adds a goal to avoid the failure to its list of goals to satisfy and this new goal is also used to search for a plan. For example, if it predicts that stir frying chicken with snow peas will lead to soggy snow peas because the chicken will sweat liquid the pan, it searches for a stir fry plan that avoids the problem of vegetables getting soggy when cooked with meats. In doing so, it finds a past plan for beef and broccoli that solves this problem by stir frying the vegetable and meat separately. The important similarity between the current situation and the one for which the past plan was built is that the same problem rises out of the interaction between the planner's goals, although the goals themselves are different.

CHEF indexes its plans in memory by both the goals that they satisfy and the the problems that they avoid. It also tries to predict problems *before* any other planning is done. This means that the anticipation of a problem can be used to find a plan in memory that avoids it while also satisfying the goals that the planner has been given, allowing CHEF to anticipate and then avoid problems before they actually arise.

IV LEARNING FROM FAILURE

Once CHEF builds a plan, it runs a simulation of it that is CHEF's version of the real world. The results of this simulation are checked against the goals that CHEF expects the plan to satisfy. These goals include the goals CHEF is given by the user as well as those it understands should be satisfied by an instance of the type of plan it has built. CHEF understands that its plan for strawberry soufflé should include the strawberries requested by the user and also understands that it should, like all soufflés, be baked and fluffy.

If the goals that the plan is designed to satisfy are not met in the results of the simulation, CHEF considers the plan a failure and begins the task of fixing the faulty plan and altering the faulty understanding of the world that was used to create the plan. It is important to note here that "failure" means the failure of a plan to achieve its goals, not a failure of the planner to create a plan. CHEF builds up a causal explanation of why the failure occurred and uses that description to access a set of repair strategies. This explanation is built by back chaining from the failure to the initial steps or states that caused it, using a set of causal rules the describe the results of actions in different circumstances. Once the explanation is built and the strategies are accessed, CHEF tries to implement the different strategies as actual changes to the plan. It makes the one change which seems most likely to succeed without introducing any new problems.

In the example of the failed strawberry soufflé, CHEF explains the fallen soufflé as a result of an imbalance between the liquid and leavening in the recipe. This imbalance is traced back to the strawberries that were added in order to meet the user's goals. In terms of CHEF's vocabulary, this problem is a case of SIDE-EFFECT:DISABLED-CONDITION:BALANCE because a side-effect of adding the strawberries has disabled a balance condition that is required for the success of the BAKE step in the plan. This explanation is used to find a planning TOP, one of a set of structures that correspond to different planning problems, and this TOP suggests the actual strategies that are used to repair the plan. These TOPs are planning versions of the Thematic Organization Packets suggested by Roger Schank for use in understanding [6]. They are designed to organize memories around complex goal and plan interactions.

```
Searching for TOP using following explanation:

Failure = It is not the case that: The batter is risen.
Initial plan = Bake the batter for twenty five minutes.
Condition enabling failure = There is an imbalance
 between the whipped stuff and the thin liquid.
Cause of condition = Pulp the strawberry.
The goals enabled by the condition = NIL
The goals that the step causing the condition enables =
 The dish now tastes like berries.
```

```
Found TOP TOP3 -> SIDE-EFFECT:DISABLED-CONDITION:BALANCE
TOP -> SIDE-EFFECT:DISABLED-CONDITION:BALANCE has 5
       strategies associated with it:
          ALTER-PLAN:SIDE-EFFECT        RECOVER
          ALTER-PLAN:PRECONDITION       ADJUNCT-PLAN
          ADJUST-BALANCE:UP
```

Each of the different strategies under the TOP suggests a change in the causal chain that leads to the current failure.

- ALTER-PLAN:SIDE-EFFECT suggests using an action to achieve the initial goal that does not have the offending side-effect. In this situation this would mean finding a way to add the taste of strawberries that does not add extra liquid. The alteration CHEF finds is to use strawberry preserves rather than crushed strawberries.
- ALTER-PLAN:PRECONDITION suggests finding a alternative to the blocked step that does not require the conditions that the first part of the plan has violated. This would mean finding a step to make the batter rise that does not require the balance between the liquid and leavening. CHEF can find no action that will do this.
- RECOVER suggests putting a step between the action that caused the side-effect and the step it interferes with that will remove the offending state. This means finding a step that will remove the liquid that results from chopping the strawberries before the batter is baked. CHEF finds that draining the strawberries will do this.
- ADJUNCT-PLAN suggests adding a new step to run concurrent with the step that has the violated condition that will allow it to satisfy the goal even in the presence of the violation. In this example this means finding a step that will allow baking the existing batter to have the desired effect. CHEF finds that adding flour to the batter will do this.
- ADJUST-BALANCE:UP suggests adjusting the down-side of the imbalanced relationship by adding more of what there is less of. In this example, this means adjusting the down-side of the imbalanced liquid and leavening relationship. This means adding more of the egg-white used as leavening.

CHEF ends up using the suggestion made by ADJUST-BALANCE:UP to add more egg white because this is the change that has the least possibility of creating any unwanted side-effects. This is determined using a set of heuristics that evaluate different changes at the level of the domain.

Once this change is made CHEF is in a powerful position. It has a working plan for a set of goals and it knows that this plan avoids a particular problem. It also has an explanation of why the problem occurred in the first place and can use this explanation to figure out which features will predict it at a later date. This means it can perform both of the tasks it needs to do in order to avoid this failure in the future: it can index the new plan in memory by the fact that it is a special plan that deals with this problem and it can build the links between features in the situation and its memory of the failure that will allow it to anticipate the problem in similar circumstances and thus find the plan that handles it.

CHEF indexes the new plan under all of the goals that it satisfies as well as the problems that it solves. The fact that it solves a certain problem is one of the important features of a plan but is not the only one. Other features include the initial input goals and the goals inferred by CHEF from the nature of the dish requested and the ingredients used.

```
Indexing STRAWBERRY-SOUFFLE under the features:

 Goals requested and inferred:
  Include strawberry in the dish.
  Make a souffle.
  The batter is now risen.
  The dish now tastes like berries.
  The dish now tastes sweet.

 Problems avoided:
  The plan avoids the failure
      'It is not the case that: The batter is now risen.'
  caused by conditions:
        "Chopping fruits produces liquid."
        "Without a balance between liquids and leavening
         the batter will fall."
```

The repaired plan is only part of what the planner learns. It also learns to recognize the situations in which the plan is useful. It does this by stepping through the causal explanation it has built and using the constraints on the rules that were used in connecting actions to effects and states to the results that they enable.

CHEF uses the explanation to point out which features in a situation are responsible for a failure and uses it again to find the features that will be predictive of the failure. Here CHEF wants to know not only the exact features that caused the problem but also the more general versions of them that might cause it again. It gets these more general features by *generalizing to the level of the rules.* This means generalizing an object in an explanation up to the highest level of description possible, while staying within the confines of the rules that explain the failure.

In the STRAWBERRY-SOUFFLE situation, one rule explains that the liquid was a product of the chopping of the strawberries. A simple way to predict this failure in the future would be for the planner to mark STRAWBERRY as predictive of it and be reminded of the failure whenever it is asked to make a strawberry soufflé. But the rule that explains the added liquid as a side-effect of chopping the strawberries does not require that the the object of the step be strawberries. It explains that chopping *any* fruit will produce this side-effect. So, instead of marking STRAWBERRY as predictive of the problem, CHEF can mark FRUIT as predictive.

```
Building demons to anticipate failure.

Building demon: DEMON2 to anticipate interaction
 between rules: "Chopping fruits produces liquid."
                "Without a balance between liquids
                 and leavening the batter will fall."

Indexing demon: DEMON2 under item: FRUIT
by test: Is the item a FRUIT.
```

```
Indexing demon: DEMON2 under style: SOUFFLE

Goal to be activated = Avoid failure of type
SIDE-EFFECT:DISABLED-CONDITION:BALANCE
exemplified by the failure 'The batter is now flat'
in recipe STRAWBERRY-SOUFFLE.
```

A causal explanation of why a failure occurs is a *chain* of events and states in which each link is a potential predictor of the failure occurring again. The goal to include strawberries is the outermost link in this chain while the liquid they produce when chopped is a more direct cause of the failure. Because the liquid from the strawberries is just as much a cause of the problem as the goals to include the strawberries, it can also be used to predict the failure at a later date.

States that are intermediate links in causing a failure are marked as predictive of the problem along with the initial goals that started the chain of events leading to it. CHEF generalizes these states up to the level of the rules that explain the failure and links them to a token representing the failure. Because the presence of liquid is implicated in causing the failure with the STRAWBERRY-SOUFFLE, the goal to include any liquid spice is linked to the memory of the failure. This is implemented by placing a test on SPICE that checks the texture and partially activates the memory of the failure when it is liquid.

```
Building demon: DEMON3 to anticipate interaction
 between rules: "Liquids make things wet."
                "Without a balance between liquids
                 and leavening the batter will fall."

Indexing demon: DEMON3 under item: SPICE
by test: Is the TEXTURE of item LIQUID.

Indexing demon: DEMON3 under style: SOUFFLE

Goal to be activated = Avoid failure of type
SIDE-EFFECT:DISABLED-CONDITION:BALANCE
exemplified by the failure 'The batter is now flat'
 in recipe STRAWBERRY-SOUFFLE.
```

By examining this failure, CHEF is able to learn the features that will predict similar problems at a later date. This knowledge is in the form of the links going from the surface features of CHEF's own goals to memories of the failure itself. When these surface features arise in later situations, the memory of the failure is activated and CHEF infers that the problem is going to arise as well.

These links are arranged so that all features responsible for a failure have to be present for it to be predicted. But different combinations of features may all predict the same failure. Figure 1 shows a simplified version of the activation links leading to a failure. When all links leading into the memory of a failure are activated the memory is also activated. The test for the texture of the goal to include any spice controls the flow of activation through that link.

By using the explanation of the failure to identify the important features in the situation CHEF gains in three ways. First it is able to learn from a single instance and avoid the problems inherent to the repetition of examples required by inductive learning systems. Second, it is able to identify a range of situations as predictive of a problem by following the causal chain defined by the explanation from the first causes of the problem to the ones more immediate to the actual failure. Third, it is able to use the rules that were used to explain the situation to control the level of generalization of the features marked as predictive of the problem. The explanation gives CHEF the information it needs to learn from a single instance and anticipate this problem in the most general and widest range of situations possible.

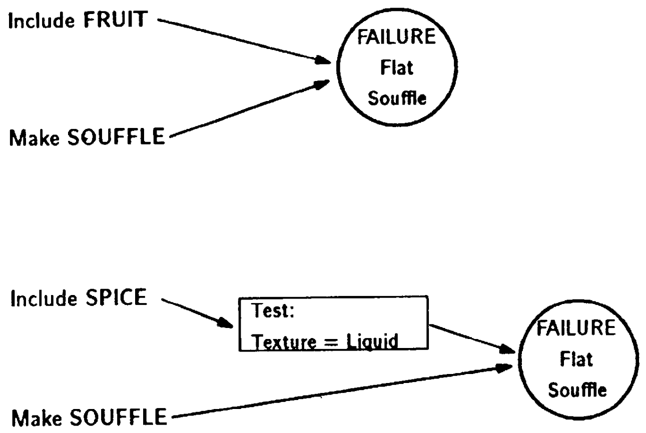

Figure 1: Links leading to the memory of the fallen soufflé

V PREDICTING A NEW PROBLEM

Once CHEF has learned the features that predict a problem, it is able to anticipate the problem from those features. It does this by sending out activations along all of the links from the goals in a new input and attending to any past failures that it is reminded of.

After solving the problem of the strawberry soufflé, CHEF is asked to make a soufflé with the liqueur kirsch. Before planning for these new goals, CHEF activates them and this activation is spread to any failures that they predict. In this case, the goal to include kirsch activates the goal to include any spice, which in turn sends an activation towards the memory of the fallen soufflé. Because kirsch is a liquid and thus passes the test along this line of activation, the signal reaches the memory of the failure. At the same time, the goal to make a soufflé sends off an activation signal to the same memory. When both links leading to the memory are activated it is also activated and CHEF responds by adding a goal to avoid this problem into its current goal list.

```
Searching for plan that satisfies -
   Include kirsch in the dish.
   Make a souffle.

Collecting and activating tests.

Fired: Is the dish STYLE-SOUFFLE.

Fired: Is the item a SPICE.
       Is the TEXTURE of item LIQUID.

   Kirsch + Souffle = Failure
  "Liquids make things wet."
```

```
    "Without a balance between liquids and leavening the
     batter will fall."
     Reminded of STRAWBERRY-SOUFFLE.
     Fired demon: DEMON3

Adding goal: Avoid failure of type
SIDE-EFFECT:DISABLED-CONDITION:BALANCE exemplified by
the failure 'The batter is now flat' in recipe
STRAWBERRY-SOUFFLE.
```

CHEF is able to predict this problem, even though the surface features of its current situation do not match those of the past situation, because the links formed in response to the past failure were made on the basis of a causal understanding of why the failure actually occurred. By using this causal explanation, the planner was able to learn the true extent of the problem and anticipate it in markedly different circumstances than those in which it originally occurred.

VI USING THE PREDICTION

Once the prediction of a failure is made, the planner searches for an existing plan that satisfies as many of its current goals as possible while avoiding the predicted problem. Plans that are modified in response to failures are indexed by the fact that they deal with those failures, so the prediction of a particular problem can be used to index to a plan that solves it.

In planning for the kirsch soufflé, the prediction of the failure allows CHEF to access the existing strawberry soufflé plan. Without the prediction, another plan, a recipe for a vanilla soufflé, would have been used because it has more surface features in common with the goals that CHEF has in hand. But this plan was used in the past to construct the failed strawberry soufflé and the standard modifications that CHEF uses would have also led to a failure in this instance. The fact that CHEF recognizes that its present situation is analogous to a past one in which a problem has occurred allows it to find a past plan that avoids that problem even though it has fewer surface features in common with the present situation than another plan in memory.

```
Searching for plan that satisfies -
   Make a souffle.
   Avoid failure of type
     SIDE-EFFECT:DISABLED-CONDITION:BALANCE.
   Include kirsch in the dish.

Driving down on: Make a souffle.
 Succeeded -
Driving down on: Avoid failure of type
      SIDE-EFFECT:DISABLED-CONDITION:BALANCE.
 Succeeded -
Driving down on: Include kirsch in the dish.
 Failed - Using recipe from current level.
Found recipe -> REC12 STRAWBERRY-SOUFFLE

Recipe exactly satisfies goals ->
    Make a souffle.
    Avoid failure of type
      SIDE-EFFECT:DISABLED-CONDITION:BALANCE.

Recipe must be altered to match ->
    Include kirsch in the dish.
```

Because this recipe has already been adapted to the problems of added liquid, it can easily be modified to include kirsch rather than strawberries and runs without failure.

VII CONCLUSIONS

Planning failures can tell a planner where its own reasoning has gone wrong. They can provide information about what features will tend to lead to a failure and when to anticipate them in later planning. A planner that learns from one failure to anticipate later ones and uses that anticipation to find the plans that deal with it is able to avoid those failures that it has already encountered.

CHEF learns from its own errors and thus avoids them in the later planning. Learning from a dozen examples at the same level of complexity as the one discussed here, it identifies the problematic features of its domain and creates the plans to deal with them.

By using a causal explanation of why a failure has occurred to identify the features will predict it in the future CHEF is able to learn from a single instance and anticipate the problem in the most general and widest range of situations possible. And once the problem is anticipated, it can be avoided by making use of a plan designed to deal with it. Unlike planners that only store their successes, CHEF is able to improve itself by learning to avoid the mistakes that other planners are unable to anticipate.

REFERENCES

[1] Carbonell, J., Derivational Analogy and its Role in Problem Solving, *Proceedings of the National Conference on Artificial Intelligence,* AAAI, Washington, DC, August 1983.

[2] DeJong, G., Acquiring Schemata Through Understanding and Generalizing Plans., *Proceedings of the Eight International Joint Conference on Artificial Intelligence,* IJCAI, Karlsrhue, Germany, August 1983a.

[3] Minton, S, Selectively Generalizing Plans for Problem-Solving, *Proceedings of the Ninth National Conference on Artificial Intelligence,* AAAI, Los Angeles, CA, August 1985, pp. 313-315.

[4] Mitchell, T., Learning and Problem Solving, *Proceedings of the Eighth International Joint Conference on Artificial Intelligence,* Karlsrhue, Germany, August 1983. Computers and Thought Award Lecture.

[5] Sacerdoti, E., *A structure for plans and behavior,* Technical Report 109, SRI Artificial Intelligence Center, 1975.

[6] Schank, R., *Dynamic memory: A theory of learning in computers and people,* Cambridge University Press, 1982.

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

[8] Wilensky, R., *Planning and Understanding,* Addison-Wesley, Reading, Mass, 1983.