and new solutions. It is a theory of learning within the context of active planning.

A case-based planner learns new plans, the features that predict failures and past repairs to faulty plans that it can reuse. This learning is accomplished by saving the different results of the planner's own experience.

6.5 CHEF: A Case-based Planner

CHEF is a case-based planner whose primary domain is Szechwan cooking. CHEF creates new recipes, i.e., cooking plans, from old ones, in response to requests for dishes with particular ingredients and tastes. CHEF has to build one plan that satisfies a number of goals simultaneously. In this, CHEF's job is much harder than many other planners, which can create a simple sequence of plans to satisfy several goals. The need to satisfy many goals at once occurs in many design domains, such as industrial design or circuit construction.

Here is an example of CHEF planning for the problem of building a stir-fry dish with beef and broccoli. CHEF has three goals: to have a stir-fried dish, to include beef, and to include broccoli. Instead of searching for individual plans that satisfy each of the goals separately and then merging them, as most hierarchical planner's would do, CHEF searches its memory for a single plan that matches, or partially matches, as many of the goals as possible. If there are several such plans that match equally well, CHEF chooses one at random.

Searching for plan that satisfies -
   Include beef in the dish.
   Include broccoli in the dish.
   Make a stir-fry dish.

Found recipe -> REC2 BEEF-WITH-GREEN-BEANS

Recipe exactly satisfies goals ->
   Make a stir-fry dish.
   Include beef in the dish.

Recipe partially matches ->
   Include broccoli in the dish.
   in that the recipe satisfies:
      Include vegetables in the dish.
CHEF begins planning by finding a plan that satisfies as many of its active goals as possible.

Once the base-line plan is found, it is modified to match whatever goals it doesn’t already satisfy. The modification rules are specific to the kind of dish (STIR-FRY), and by the goal of the modification (to include broccoli). This kind of knowledge in cooking is similar to the knowledge in other design domains such as architecture. In architecture, instead of knowing how to modify STIR-FRY plans, an expert knows how to alter plans for houses and apartment buildings. Likewise, instead of knowing how to add broccoli or beef to an existing recipe, an expert would know how to add windows or doors to an existing plan.

In this case, the partial match between the target goal, including broccoli, and an object in the existing recipe, green beans, tells the planner that it can replace the broccoli for the green beans directly. Other structures, called *object critics* then change the cooking time of the vegetables to account for the difference between the requirements of green beans and broccoli, and add a step to chop the broccoli before stir frying it.

```
Building new name for copy of BEEF-WITH-GREEN-BEANS
Calling recipe BEEF-AND-BROCCOLI

Modifying recipe: BEEF-AND-BROCCOLI
  to satisfy: Include broccoli in the dish.

Placing some broccoli in recipe BEEF-AND-BROCCOLI

- Considering ingredient-critic:
  Before doing step: Stir fry the -Variable-
    do: Chop the broccoli into pieces the size of chunks.
- ingredient-critic applied.
```

CHEF alters old plans to satisfy new goals using a set of modification rules and a set of *object critics*.

Once these modifications are made, the planner has a plan that should satisfy all of the initial goals it was given.

BEEF-AND-BROCCOLI
A half pound of beef
Two tablespoons of soy sauce
One teaspoon of rice wine
A half tablespoon of corn starch
One teaspoon of sugar
A half pound of broccoli
One teaspoon of salt
One chunk of garlic

Chop the garlic into pieces the size of matchheads.
Shred the beef.
Marinate the beef in the garlic, sugar, corn starch,
rice wine and soy sauce.
Chop the broccoli into pieces the size of chunks.
Stir fry the spices, rice wine and beef for one minute.
Add the broccoli to the spices, rice wine and beef.
Stir fry the spices, rice wine, broccoli and beef
for three minutes.
Add the salt to the spices, rice wine, broccoli and beef.

CHEF knows more than just the steps of this plan. It also knows what
goals should be satisfied when it is run. These goals are derived from its
knowledge of stir-frying and its understanding of what items are important in
this recipe. In particular, executing this plan should result in the following:

The beef is now tender. The dish now tastes salty.
The dish now tastes savory. The dish now tastes sweet.
The broccoli is now crisp. The dish now tastes like garlic.

CHEF derives the goals that a plan should satisfy from general
plan specifications and the particulars of the current plan.

But not all plans work the first time. In this plan, the goal to have the
broccoli crisp will fail, because it is being stir fried with the beef and is cooked
in the liquid that the beef produces.

6.5.1 When Plans Fail

CHEF discovers the failure by simulating the execution of the plan. This
simulation is the program's equivalent of real world execution. The simulator
uses a set of inference rules to determine the outcome of each step of the
plan. CHEF looks for any failures to achieve its positive goals as well as any occurrences of negative states that it would like to avoid.

Checking goals of recipe -> BEEF-AND-BROCCOLI

Recipe -> BEEF-AND-BROCCOLI has failed goals.

The goal: The broccoli is now crisp.
is not satisfied.
It is instead the case that: The broccoli is now soggy.

Unfortunately: The broccoli is now a bad texture
In that: The broccoli is now soggy.

Changing name of recipe BEEF-AND-BROCCOLI
to BAD-BEEF-AND-BROCCOLI

CHEF checks the goals of a plan against the results of a simulation of the plan that is its equivalent of the real world.

A plan failure means two things to CHEF. First, it must repair the plan. Second, it has to repair its understanding of the world. For both of these tasks CHEF needs to understand exactly why the failure has come about. It is not enough that it can point to the fact that the broccoli isn’t crisp and call this a failure. It must also be able to explain why this particular failure has happened. What series of steps and states that led to the failure? CHEF uses the explanation to find the appropriate repair strategies. Because a causal explanation is used to describe the problem, a set of fixes that are aimed at altering the causality behind the failure can be used, instead of weaker methods such as back-tracking or total replanning.

6.5.2 Explaining Plan Failures

An explanation describes the failure, the step that caused it, and the conditions that had to be true for it to come about. CHEF also knows what goals were being served by the steps and states that caused the failure. With this knowledge, CHEF identifies what has to be changed in the plan to solve the present problem and also identifies what has to be changed in the planner’s knowledge of the world so that the failure will not occur again.

CHEF diagnoses the failure by “asking” itself a sequence of questions. The answers to these questions organize the information about the cause of
the failure. There is a significant simplification in the failure explanation process in CHEF. The simulation rules that test the recipe leave a record of all the causal connections. The simulation does not just report "the broccoli is soggy." It also reports that "water in the pan made the broccoli soggy," "cooking the meat left water in the pan," and so on. In other words, the real-world simulator and CHEF's causal model of the world are one and the same. This is clearly not the way things work in real life, but a model of explanation is way beyond the scope of this research. (See Collins in this volume for an example of how complicated causal explanation can become.)

Explaining the following failures:

It is not the case that: The broccoli is now crisp.

in that: The broccoli is now soggy.

The broccoli is now a bad texture.

in that: The broccoli is now soggy.

In: BAD-BEEF-AND-BROCCOLI

ASKING THE QUESTION: 'What is the failure?'

ANSWER-> The failure is: It is not the case that: The broccoli is now crisp.

ASKING THE QUESTION: 'What is the preferred state?'

ANSWER-> The preferred state is: The broccoli is now crisp.

ASKING THE QUESTION:

'What was the plan to achieve the preferred state?'

ANSWER-> The plan was: Stir fry the sugar, soy sauce, rice wine, garlic, corn starch, broccoli and beef for three minutes.

ASKING THE QUESTION:

'What were the conditions that led to the failure?'

ANSWER-> The condition was: There is thin liquid in the pan from the beef equaling 4.8 teaspoons.

ASKING THE QUESTION:

'What caused the conditions that led to the failure?'
6.5. CHEF: A CASE-BASED PLANNER

ANSWER→ There is thin liquid in the pan from the beef equaling 4.8 teaspoons was caused by: Stir fry the sugar, soy sauce, rice wine, garlic, corn starch, broccoli and beef for three minutes.

ASKING THE QUESTION:
'Do the conditions that caused the failure satisfy any goals?'

ANSWER→ The condition: There is thin liquid in the pan from the beef equaling 4.8 teaspoons is a side effect only and meets no goals.

ASKING THE QUESTION:
'What goals does the step which caused the condition enabling the failure satisfy?'

ANSWER→ The step: Stir fry the sugar, soy sauce, rice wine, garlic, corn starch, broccoli and beef for three minutes. Enables the satisfaction of the following goals: The dish now tastes savory. The beef is now tender.

CHEF explains the failures that it encounters through a causal description of why they have occurred.

6.5.3 Plan Repair Strategies and TOPs

At this point, then CHEF knows what steps and states combined to cause the current failure. It also knows which goals were being pursued in taking the actions and creating the states that led to its failure.

This explanation serves two functions. First, it describes the planning problem in a general causal vocabulary that can be used to access some equally general repair strategies. Second, the explanation points out which features in the domain interact to cause this sort of failure to occur, and hence which features should be watched out for in the future.

Repair strategies are indexed under goal interaction structures called Thematic Organization Packets or TOPs [Schank, 1982]. TOPs are similar in
function to the critics found in HACKER [Sussman, 1975] and NOAH [Sacerdotti, 1977]. Each TOP is indexed by a particular type of planning problem and organizes a set of strategies for dealing with that type of problem. These strategies take the form of general repair rules such as REORDER steps and RECOVER from side effects. Each general strategy is filled in with the specifics of the particular problem to build a description of a change in the plan that would solve the current problem. This description is used as an index into a library of plan modifiers in the cooking domain. The modifications found are then tested against one another using rules concerning the efficacy of the different changes, and the one that is most likely to succeed is chosen.

The idea behind TOPs is simple. There is a great deal of planning information that is related to the interactions between plans and goals. This information cannot be tied to any individual goal or plan but is instead tied to problems that arise out of their combination. Each TOP corresponds to a planning problem due to the causal interaction between the steps and the states of a plan. When a problem arises, a causal analysis of it identifies the TOP and hence locates the repair strategies needed to fix the problem.

This explanation of the failure in this example indexes to the TOP SIDE-EFFECT:DISABLED-CONDITION:CONCURRENT (SE:DC:C), a memory structure related to the interaction between concurrent plans in which a side effect of one violates a precondition of the other. This is because the side effect of liquid coming from the stir-frying of the beef is disabling a precondition attached to the broccoli stir-fry plan that the pan being used is dry.

CHEF knows about 20 TOPs. Besides SE:DC:C, there is another one, DESIRED-EFFECT:DISABLED-CONDITION:SERIAL (DE:DC:S), that describes a situation in which the desired effect of a step interferes with the satisfaction conditions of a later step. CHEF recognizes that the current situation is a case of SE:DC:C rather than DE:DC:S, because it has determined that no goal is satisfied by the interfering condition (the liquid in the pan). Had the liquid in the pan satisfied a goal, the situation would have been recognized as a case of DE:DC:S, and different repair strategies would be needed.

Found TOP TOP1 -> SIDE-EFFECT:DISABLED-CONDITION:CONCURRENT
TOP -> SIDE-EFFECT:DISABLED-CONDITION:CONCURRENT has 3 strategies associated with it:
SPLIT-AND-REFORM
ALTER-PLAN:SIDE-EFFECT
ADJUNCT-PLAN

CHEF uses its causal description of a problem to find a TOP that has strategies which will solve it.
6.5. CHEF: A CASE-BASED PLANNER

This TOP has three strategies associated with it: SPLIT-AND-REFORM suggests breaking a problematic step into several separate steps, ALTER-PLAN:SIDE-EFFECT suggests replacing the problematic step with a different plan, and ADJUNCT-PLAN suggests adding a new step to counteract the effects of the problematic step. CHEF tries to apply each strategy to the current example. It can only instantiate SPLIT-AND-REFORM. The other strategies turn out not to fit the current situation, given CHEF's limited knowledge base.

Applying TOP \(\rightarrow\) SIDE-EFFECT:DISABLED-CONDITION:CONCURRENT to failure it is not the case that: The broccoli is now crisp.

in recipe BAD-BEEF-AND-BROCCOLI

Asking questions needed for evaluating strategy:
SPLIT-AND-REFORM

ASKING \(\rightarrow\) Can plan
Stir fry the sugar, soy sauce, rice wine, garlic, corn starch, broccoli and beef for three minutes.

be split and rejoined

Found plan: Instead of doing step: Stir fry the sugar, soy sauce, rice wine, garlic, corn starch, broccoli and beef for three minutes
do:
S1 = Stir fry the broccoli for three minutes.
S2 = Remove the broccoli from the result of action S1.
S3 = Stir fry the sugar, soy sauce, rice wine, garlic, corn starch and beef for three minutes.
S4 = Add the result of action S2 to the result of action S3.
S5 = Stir fry the result of action S4 for a half minute.

Asking questions needed for evaluating strategy:
ALTER-PLAN:SIDE-EFFECT

ASKING \(\rightarrow\) Is there an alternative to
Stir fry the sugar, soy sauce, rice wine, garlic, corn starch, broccoli and beef for
three minutes.
that will enable
The dish now tastes savory.
which does not cause
There is thin liquid in the pan from the
beef equaling 4.8 teaspoons.

No alternate plan found

Asking questions needed for evaluating strategy:

*ADJUNCT-PLAN*

ASKING → Is there an adjunct plan that will disable
- There is thin liquid in the pan from the beef
  equaling 4.8 teaspoons.
- that can be run with
  - Stir fry the sugar, soy sauce, rice wine, garlic, corn starch, broccoli and beef for three minutes.

No adjunct plan found

Deciding between modification plans suggested by
strategies:
- Only one modification can be implemented
  → *SPLIT-AND-REFORM*

Implementing plan → Instead of doing step: Stir fry the
- sugar, soy sauce, rice wine, garlic, corn starch, broccoli and beef for three minutes.
do: S1 = Stir fry the broccoli for three minutes.
  S2 = Remove the broccoli from the result of action S1.
  S3 = Stir fry the sugar, soy sauce, rice wine, garlic, corn starch and beef for three minutes.
  S4 = Add the result of action S2 to the result of action S3.
  S5 = Stir fry the result of action S4 for a half minute.

Suggested by strategy SPLIT-AND-REFORM.
CHEF tries to implement the strategies suggested by the TOP by searching for the actual plan modifications that the strategies define.

6.5.4 Anticipating Failures

After changing the plan to solve immediate problems, CHEF also changes its understanding of the world so that it will be able to anticipate and avoid similar problems in the future. To do this, CHEF must figure out what it was that caused the failure in the first place. It must decide which features in the initial situation contributed to the failure so that later requests for similar goals can be planned for. Here CHEF again makes use of the causal explanation that it has for the failure to back-chain to the features that can be used to predict this kind of problem. These features are then generalized to the highest level of description allowed by the rules that explain the situation. BEEF is generalized to MEAT in the example, because the rule that explains the liquid in the pan states that stir frying any meat will have this result. The resulting descriptions are marked as predictive of this problem. Once this is done, CHEF is able to predict and plan for a problem of this kind whenever it encounters a similar situation.

Building demons to anticipate failure.

Building demon: DEMON0 to anticipate interaction between rules:
"Meat sweats when it is stir-fried."
"Stir-frying in too much liquid makes vegetables soggy."

Indexing demon: DEMON0 under item: MEAT by test:
Is the item a MEAT.

Indexing demon: DEMON0 under item: VEGETABLE by test:
Is the item a VEGETABLE.
and Is the TEXTURE of item CRISP.

Goal to be activated = Avoid failure of type SIDE-EFFECT:DISABLED-CONDITION:CONCURRENT exemplified by the failure 'The broccoli is now soggy'
BUILDING DEMON: DEMON1 to anticipate interaction between rules:
"Liquids make things wet."
"Stir-frying in too much liquid makes vegetables soggy."

Indexing demon: DEMON1 under item: SPACE by test:
Is the TEXTURE of item LIQUID.

Indexing demon: DEMON1 under item: VEGETABLE by test:
Is the item a VEGETABLE.
and Is the TEXTURE of item CRISP.

Goal to be activated = Avoid failure of type SIDE-EFFECT:DISABLED-CONDITION:CONCURRENT
e exemplified by the failure 'The broccoli is now soggy' in
recipe BEEF-AND-BROCCOLI.

CHEF uses its explanation of why a failure has occurred to
build links that will allow it to predict the failure later, in
similar circumstances.

Once a plan is debugged and the sources of any problem are marked as
predictive of that problem, CHEF stores the plan using two types of features.
The first is the standard vocabulary of goals met by the plan, such as what
foods and tastes are included in it, as well as what type of dish it is. These
features are generalized so that plans which only partially satisfy a set of goal
requests can be accessed in the absence of exact matches. The generalizations
allowed depend on the modification rules available. Since CHEF knows how to
replace one vegetable with another, a recipe with green beans can be indexed
as a recipe with a vegetable.

The second vocabulary indexes repaired plans by the failures they avoid.
By storing plans this way, it is possible for the planner to use the prediction
of a previously encountered problem to find a plan that solves that problem.
CHEF can avoid past errors by predicting them and then finding a plan that
deals with them. In the case of the beef and broccoli, the plan is indexed by
the fact that it deals with the interaction between the meat sweating and the
need to have a dry pan while cooking crisp vegetables.
CHEF uses its new knowledge of what went wrong in the making the BEEF-AND-BROCCOLI recipe to anticipate and avoid the same problem in making a later one. We test this by next asking CHEF for a stir-fried dish with chicken and snow peas. The output shows CHEF predicting the possibility of the snow peas getting soggy if cooked with the chicken.

Searching for plan that satisfies -
Include chicken in the dish.
Include snow pea in the dish.
Make a stir-fry dish.

Collecting and activating tests
Fired: Is the dish STYLE-STIR-FRY.
Fired Is the item a MEAT.
Fired: Is the item a VEGETABLE.
Is the TEXTURE of item CRISP.

Chicken + Snow Pea + Stir frying = Failure
"Meat sweats when it is stir-fried."
"Stir-frying in too much liquid makes vegetables soggy."
Reminded of BEEF-AND-BROCCOLI.
Fired demon: DEMO

Based on features found in items snow pea, chicken and stir fry
Adding goal: Avoid failure of type
SIDE-EFFECT:DISABLED-CONDITION:CONCURRENT exemplified by
the failure 'The broccoli is now soggy' in recipe
BEEF-AND-BROCCOLI.

CHEF uses its understanding of past failures to anticipate
problems that will arise in new situations.

Once a problem has been anticipated, CHEF can use the prediction of the problem to find a plan that avoids it. It does this by adding a goal to avoid the problem to the list of goals it is using to search for a plan. The BEEF-AND-BROCCOLI plan is indexed in memory by the fact that it satisfies the goal of avoiding the problem of the interaction between meat and vegetable.
Searching for plan that satisfies -
Include chicken in the dish.
Include snow pea in the dish.
Make a stir-fry dish.
Avoid failure of type
  SIDE-EFFECT:DISABLED-CONDITION:CONCURRENT
  exemplified by the failure 'The broccoli is now soggy'
  in recipe BEEF-AND-BROCCOLI.

Found recipe -> REC9 BEEF-AND-BROCCOLI

Recipe exactly satisfies goals ->
  Avoid failure of type
  SIDE-EFFECT:DISABLED-CONDITION:CONCURRENT
  exemplified by the failure 'The broccoli is now soggy'
  in recipe BEEF-AND-BROCCOLI.
  Make a stir-fry dish.

Recipe partially matches ->
  Include chicken in the dish.
  in that the recipe satisfies: Include meat in the dish.

Recipe partially matches ->
  Include snow pea in the dish.
  in that the snow pea can be substituted for the broccoli

Building new name for copy of BEEF-AND-BROCCOLI based on its goals.
Calling recipe CHICKEN-STIR-FRIED

Modifying recipe: CHICKEN-STIR-FRIED
to satisfy: Include chicken in the dish.
  Placing some chicken in recipe CHICKEN-STIR-FRIED

Modifying recipe: CHICKEN-STIR-FRIED
to satisfy: Include snow pea in the dish.
  Placing some snow pea in recipe CHICKEN-STIR-FRIED
Created recipe CHICKEN-STIR-FRIED

CHICKEN-STIR-FRIED

A half pound of chicken
Two tablespoons of soy sauce
One teaspoon of rice wine
A half tablespoon of corn starch
One teaspoon of sugar
A half pound of snow pea
One teaspoon of salt
One chunk of garlic

Bone the chicken.
Chop the garlic into pieces the size of matchheads.
Shred the chicken.
Marinate the chicken in the garlic, sugar, corn starch, rice wine and soy sauce.
Chop the snow pea into pieces the size of chunks.
Stir fry the snow pea for three minutes.
Remove the snow pea from the pan.
Stir fry the spices, rice wine and chicken for three minutes.
Add the snow pea to the spices, rice wine and chicken.
Stir fry the spices, snow pea, rice wine and chicken for a half minute.
Add the salt to the spices, snow pea, rice wine and chicken.

CHEF uses its anticipation of a problem to find a plan that avoids it.

When CHEF decides to use the beef and broccoli recipe as its starting point, it passes up other recipes in memory that might appear to have more in common with the input goals. It has, for example, a recipe with chicken and green beans that, in the absence of the beef and broccoli experience, it would have been happy to modify to account for the chicken and snow peas. But it prefers a plan that fixes the predicted problem, because the interaction between plan steps has a greater effect on the final structure of the plan than the goals to include certain ingredients.

The lesser goals are not ignored, however. If CHEF were asked for a stir fry dish with chicken and bean sprouts after storing its new CHICKEN-
STIR-FRIED plan in memory, it would anticipate the problem with the bean sprouts getting soggy, but would start with CHICKEN-STIR-FRIED, rather than BEEF-AND-BROCCOLI, because CHICKEN-STIR-FRIED deals with the problem of stir frying meats and crisp vegetables together and has more surface features in common with the current situation. Once a failure is anticipated, it becomes one of the features that is used to find a plan, but not the only one.

6.6 Learning Plans

A case-based planner learns plans in the sense that it stores the new plans it creates in its plan memory. Learning, in this context, means figuring out the features that should be used index the plan in memory and then storing the plan using them. The task is to organize experience such that it can be retrieved and reused at the appropriate time.

A case-based planner stores the most obvious result of planning, the plan itself, in a plan memory.

The stored plan is a series of steps and list of ingredients that has been built to satisfy some particular set of goals. This plan is not generalized in any way. To do so would be to lose information that could be used again, without gaining anything in terms of more general applicability.

For example, it’s as easy for CHEF to modify STRAWBERRY-SOUFFLE to use kirsch instead of strawberries as it would be for CHEF to use a generalized version of the same recipe. But if CHEF is asked for strawberry soufflé, perhaps with something else, then it can use the stored recipe directly, without having to adapt a generalized plan.

While a case-based planner need not generalize the plans it stores, it does need to generalize the features that are used to index them. Because of this, the plan can be suggested for use in a wide range of situations and can still retain the specific information that makes it more useful when applied to situations in which the goals that are being planned for are completely satisfied by the plan itself. By generalizing the indexes rather than the plan, a case-based planner is able to avoid much of the tradeoff between generality and power of application. The general indexes makes it applicable in many situations and the specificity of the plan makes it a powerful tool in those situations in which the match between the current goals and those satisfied by the plan is a good one.
In storing a plan in memory, a case-based planner does not generalize the plan itself. It instead generalizes the features that are used to index the plan in memory.

A case-based planner indexes plans by the goals that they satisfy and the problems that they avoid. The goals that a plan satisfies are states that have to be true once the plan has been run. The problems that a plan avoids are causal interactions that need to be overcome.

CHEF's goals include the ingredients, their final tastes, and textures, and the type of dish. These are the initial features that are used in indexing the plan. Problems that a plan avoids are represented by the TOP that describes the abstract causal situation linked to the specific failure state. In CHEF, a plan may solve several problems, just as it may satisfy several goals. These two sets of items are combined into a single list of features that will be used to index the plan in memory.

The fact that a plan avoids a particular problem can be treated by a case-based planner as a goal to avoid that failure that the plan associated with it satisfies. These goals to avoid failures can be treated as any other goal for use in indexing of plans in memory. They are considered more important than most other goals, however, because it is often easier to alter a plan to satisfy a new goal than to make the changes required to deal with an interaction between goals.

A plan is indexed in memory, then, by the goals that it satisfies and the problems that it avoids. It also has to be indexed by the features of a situation that are independent of the goals but do direct the planner to one plan or another. As mentioned before, the plan to call the front desk for a wake-up call is a good plan to get the planner up in the morning, but only in certain circumstances. These circumstances, which can be reduced to states that describe the world, also have to be used to index plans in memory.

In the CHEF program, plans are stored in a discrimination net [Charniak et al., 1987]. These goals and problems are ordered by their importance, with the higher priority features used at the higher levels of discrimination. Once the features are ordered, a plan is placed in a discrimination net, using the features as indexes. The ordering of features allows CHEF to somewhat limit the branching in this net by allowing discrimination from any one node to be made only on the basis of features of less importance than the last one used to index it.

As each goal is used to place a plan in memory, more general versions of the goals are also used. BEEF-AND-BROCCOLI is indexed by the fact that it includes beef and broccoli, but it is also indexed by the fact that it includes meat and vegetables in general. The level of generality of the goals used to
index plans is preset and corresponds to the level of generality that is used in defining the modification rules used in substituting one ingredient for another in a plan. As a result, the goal to include one type of ingredient can be used to find a plan for a similar ingredient.

CHEF stores new plans indexed by the goals that they satisfy and the problems that they avoid.

Once a plan is placed in memory this way it can be accessed by the planner when it is searching for a plan that satisfies similar goals and for a plan that avoids failures that it predicts may appear and wants to avoid.

For example, in building a strawberry soufflé recipe, CHEF has to deal with the fact that the liquid from the fruit prevents the soufflé from rising. It deals with this by adding more egg white to the recipe, reestablishing the relationship between liquid and leavening that the fruit unbalanced. The final recipe satisfies a set of goals having to do with making a soufflé, including strawberries, making the soufflé taste sweet and like berries and having a fluffy texture. These goals come from the input request and some are generated by the planner out of its understanding of what this type of plan should be like in general (Figure 6.7). The plan also has a token associated with it that indicates that it deals with the problem of the added liquid from the fruit interfering with the baking (Figure 6.8).

Created recipe STRAWBERRY-SOUFFLE

If this plan is successful, the following should be true:

- The batter is now baked.
- The batter is now risen.
- The dish now tastes like berries.
- The dish now tastes sweet.

The plan satisfies -
- Include strawberry in the dish
- Make a soufflé.

Figure 6.7: Goals Satisfied by STRAWBERRY-SOUFFLE

These goals are collected and used to index the plan in memory, the goal to make a soufflé and the goals to avoid the failure taking priority over all others.