6.4.3 Plan Modification

The RETRIEVER only finds and suggests past plans for new situations; it doesn’t modify these plans to fit the new goals. This is the job of the MODIFIER.

In order to modify a plan, the MODIFIER uses a library of modification rules that specify what steps to add to particular plans, given particular goals. The rules do not have to give complete plans for achieving any particular goal. They can just be the modifications that are needed to alter an existing plan to achieve that goal. The MODIFIER also needs to have information about items in its domain that tells it how to change those items to meet the conditions required by the more general modification rules. This information, in the form of special purpose critics, tailor the general modifications of a plan to the specific needs of the items required to achieve particular goals. Finally, the MODIFIER needs to know about what the plans it is modifying are supposed to be doing in general. This is needed so that it doesn’t violate the goals of the overall plan when it modifies it to satisfy a specific goal.

To alter old plans to meet new goals, the MODIFIER needs a set of modification rules, critics with knowledge of goal specific requirements, and general plan specifications.
For example, in the architectural domain, to add a window to an existing design, the MODIFIER would have to know the goal (add a window), the type of design (office, apartment or house), and the features of the particular window (the type of glass, the size, or the shape). The changes required for adding a window to the design for an apartment building are different from those that have to be made to a plan for a standard house. By storing modifications in terms of both the goal to be added and the type of plan begin altered, a MODIFIER can be sensitive to these differences. By also storing idiosyncratic steps that deal with the features of particular items in a domain, it can deal with those items within the context of a more general process of plan alteration.

Another example is in the domain of automotive design. Suppose the system wants different parts of a car it is designing to be colored red. If the part is an exterior metal piece, the alteration to the initial plan will involve changing the color of the paint that is used to cover the part. If the part is an interior plastic part, the change will involve altering the pigments used in the initial mixing of the plastic. The different initial plans determine different alterations in response to the same goal. No one plan for changing the color will do. Different alterations, associated with the different initial plans, have to be used for the different situations.

The RETRIEVER and MODIFIER together make up a basic case-based planner. The RETRIEVER takes a set of goals and finds the past plan that best satisfies them. The MODIFIER takes this plan and the goals that it fails to meet and modifies the plan to satisfy all of the input goals. To do this it needs to have plan modification rules, rules on actions that have to be performed in order to use certain items, and general information about the goals that types of plans are supposed to satisfy (Figure 6.2). For a given set
of goals, then, the RETRIEVER finds a good plan and MODIFIER makes it better.

6.4.4 Plan Storage

The original justification for case-based planning was learning. We want the planner's abilities to improve, based on its own experience. Within the confines of the planner that we have built so far, only one kind of learning is really possible. This is learning by remembering, i.e., by storing a plan that has been built in the planner's memory. This is sometimes called caching the plan.

The features that are used to store a plan must be the same as those used to access it. The knowledge that a planner needs to store a plan is exactly parallel to the knowledge it needs to find one. The indexes used to store plans, then, are the goals that the plan satisfies and the circumstances in which the goals occurred that affected plan choice.

The STORER stores in memory the results of the work done by the RETRIEVER and the MODIFIER. The STORER does nothing to help in the building of a present plan. Storing plans helps the planner save time when solving similar problems in the future. Later we'll see how these stored plans also help the planner learn from failure.

![Diagram of STORER](image)

**Figure 6.3:** The STORER

To place new plans in memory, the STORER needs to index them under the same features that the RETRIEVER uses to find them: the goals that they satisfy and the situations in which they are appropriate.
6.4.5 Plan Repair

Many planning systems stop once a plan has been generated. In the real world, this is impractical, because even the best-laid plans will fail once in a while. A real planner has to be able to repair failed plans. No matter how good the planner is, at one time or another it will have to confront problems that arise out of its own lack of knowledge and the limits of its own heuristics. Given that the planner is going to make mistakes, we have to give it some mechanism for repairing the faulty plans it builds. This mechanism will be called the REPAIRER.

The input to the REPAIRER has two parts: a faulty plan and some description of the fault. The fault is either a desired state that was not achieved or an undesired state that arose during the plan execution. How a planner gets this information can vary. It can actually run its plans and examine the results. It can run simulations of the plans and use this to diagnose errors. It can even ask an outside source if the plan will do what it wants it to. But no matter how it does it, a planner has to be able to notice and respond to its own failures.

To do its job, the RETRIEVER also needs to know why the fault arose. It needs a causal explanation of the failure. For example, the problem might be "The engine failed to start when I turned the key." The causal explanation might be "The engine failed to start because a wire leading to the starter has shorted out on the body where it passes behind the side-mounted air-filter. The heat exchange from the air-filter melts the insulation on the wire." The explanation is important because it specifies which states and actions participated in the failure. This in turn suggests which parts of a design have to be changed. Even if a specific method for fixing the particular problem doesn't already exist, one can be generated out of a method for dealing with the general problem (a side effect of one part violating the maintenance conditions of another) and the particular states (the side effect is excess heat, the maintenance condition is the fact of the insulation and so on). This could be used to suggest possible general kinds of repair, e.g., "recover from side effect" (find a way to drain off the heat from the air-filter), or "protect from side effect" (use a heat resistant insulation on the wire), or "bypass side effect" (reroute the wire).

So, along with a vocabulary for describing plan execution failures, the REPAIRER needs a set of repair methods that can be accessed with that vocabulary. These methods should be organized such that the description of a given failure will access only those repair methods that have a chance of repairing that particular plan fault. This relationship between problem and repair is like the relationship between goals and plans. Plans are indexed
under the goals they satisfy and repair methods are indexed under the types of failures that they deal with.

When a plan is repaired, the result is not only a plan that satisfies a set of goals, it is also a plan that avoids a particular problem. The original goals and plans interacted and caused a problem. The repaired plan is designed to cope with that interaction and to avoid the failure altogether. The REPAIRER therefore tells the STORER about the goals that the repaired plan satisfies and the failures that it avoids along the way. With this information, the STORER can index repaired plans in memory in such a way that the RETRIEVER can find them when a similar combination of goals occurs again.

With the addition of the REPAIRER, the STORER can now index plans by the problems that they avoid as well as the goals that they satisfy.

![Diagram of REPAIRER and STORER processes]

Figure 6.4: The REPAIRER

To summarize, the REPAIRER is invoked only when a plan fails. Its task is to repair the plan and tell the STORER how to characterize it so that it can be found again.

6.4. BUILDING ITS VOCABULARY

A vocabulary for describing the problems and the problems it can be found again again is needed. This vocabulary for describing the STORER is indexed by the REPAIRER and the problems it can be found again.

To repair failed plans, the REPAIRER uses strategies to find a replacement plan.

6.4.6 Learning to Repair Failed Plans

Even though STC and the STORER have not yet been built, this concept is still a useful one. This is the future. This is the system that is going to arise and be useful unless the circumstances are such that it cannot be used. A plan that solves a problem in a particular set of circumstances is called the ASSIGNER.

The ability to repair plans relies on the ability to repair the plan. The planner needs a failure strategy to repair the plan. A plan that caused the failure needs to be repaired. The ASSIGER and the STORER decide what to do next. The knowledge that is used by the planner is complex and requires the repair of the failure strategy.

To decide what to do, the ASSIGER needs to come off the plan and check whether a plan exists for which it can be repaired.

Take for example, the BABY PROBLEM:

To decide what to do, the ASSIGER needs to come off the plan and check whether a plan exists for which it can be repaired.
can be found again in a similar problem situation. The REPAIRER requires a vocabulary for describing plan execution problems and a set of strategies that are indexed by the problems that they solve. The repaired plan is given to the STORER to place in memory, indexed by the goals that the plan satisfies and the problems it avoids (Figure 6.4).

| To repair failed plans and describe them to the STORER, the REPAIRER requires a vocabulary of plan failures and repair strategies that are indexed by that vocabulary. |

6.4.6 Learning from Failure

Even though STORER indexes a repaired plan by the problems it avoids, there is still a stumbling block to reusing that plan for similar problems in the future. This is the RETRIEVER'S inability to figure out when the problem is going to arise again. The fact that a plan solves a particular problem is not useful unless the planner can anticipate that problem in the appropriate circumstances and use that prediction to find the plan in memory. Having a plan that solves a problem does the planner no good if it cannot recognize the circumstances in which that problem will arise.

The ability to predict when a problem is going to arise in the future rests on the ability to figure out why it happened in the past. To do that the planner needs a function that can decide which input features of a failed plan caused the failure to occur. Given these features, it can predict when the problem will arise again. This function, which assigns blame to features, is called the ASSIGENER. The job of the ASSIGENER is to look at a failed plan and decide what circumstances will be predictive of that failure in the future. The knowledge the ASSIGENER uses can vary in much the same way that the knowledge used by the REPAIRER can vary: it can be simple and unreliable or complex and robust.

| To decide which features in a situation are to blame for a failure, the ASSIGENER needs to be able to describe the causes of the failure. The more extensive its vocabulary for this description, the more exact its credit assignment will be. |

Take for example a situation in which John misses a free ride with Bill from the airport to the hotel because John has to wait for his baggage to come off the plane. Many factors led to the missed ride, including the haste with which Bill had to leave, but the main feature of John's plan that is
involved is the fact that he checked his baggage while traveling with others. When later performing a similar plan, that is, taking a flight with other people and checking baggage, John should recall the failure and plan for it by, say, taking only carry-on luggage.

So, the ASSIGNER's task is to mark features in a situation as predictive of the problems that arose in that situation. Like the STORER, the ASSIGNER has no effect on problems that the planner is currently working on. Instead, its job is to assure that the planner is able to predict when the current problem is going to arise again in later circumstances. Its output can be a set of inference rules that are to be checked in the early stages of planning, links going from surface goals to predictions, or a table of effects that matches features to predictions. The form of the output is not the issue here. The issue is that the planner, because the ASSIGNER is able to identify the features that predict a problem, is now able to anticipate that problem and use the goal of avoiding it to find a plan that does so (Figure 6.5).
6.4.7 Problem Anticipation

The ASSIGNER determines which features led to the failure. Another module, the ANTICIPATOR, uses this information to predict problems when new goals and situations come in, before any other planning is done. The ANTICIPATOR tells the RETRIEVER about these potential problems, so that the RETRIEVER can try to find a plan that avoids them (Figure 6.6).

![Diagram of the ANTICIPATOR](image)

Figure 6.6: The ANTICIPATOR

To anticipate a problem on the basis of surface features, the ANTICIPATOR needs the base of information built by the ASSIGNER.

For example, consider again poor John, who missed a ride because he had to wait for his baggage. The next time he travels, his ANTICIPATOR should remember the problems caused by waiting for luggage, and the RETRIEVER
should either find a past plan that deals with the problem or alter another
plan to do so, before John misses another ride.

Note the relationship between the ANTICIPATOR, ASSIGNER, and RETRIEVER. Once a problem plan has been repaired, it is stored in memory, indexed by the fact that it deals with a particular problem. The ANTICIPATOR is needed to enable the RETRIEVER to find the repaired plan when the same potential problem arises again. The ASSIGNER is needed to find out what the ANTICIPATOR should look for.

With the addition of the ANTICIPATOR, the RETRIEVER can search for plans on the basis of the problems that they avoid as well as the goals they satisfy

6.4.8 The Final Package

The basic case-based planner that grows out of the need to reuse plans and adapt them for new goals functions as follows: A set of goals is handed to the planner and sent directly to the ANTICIPATOR. The ANTICIPATOR, based on the knowledge built up by the ASSIGNER, makes any predictions of planning problems that it thinks will arise out of the current goals. The goals are then handed to the RETRIEVER along with the ANTICIPATOR’s predictions of problems that have to be avoided. The RETRIEVER uses both to search for a plan in memory that best satisfies the goals it is trying to achieve and avoids any problems that have been anticipated. The result is a past plan that matches some or all of the goals now being planned for.

This plan is sent to the MODIFIER, which adds or substitutes new steps to the plan in order to make it satisfy any of the planner’s goals that it does not yet achieve. Once modified, the plan is run and the results checked against the goals of the planner. If there is a failure, either because a desired goal has not been achieved or because an undesired state has resulted from the plan, the plan is given to the REPAIRER.

The REPAIRER builds a characterization of the failure and uses this to find and apply one of its repair methods. The repaired plan, along with a description of the problems that had to be solved along the way, are then sent to the STORER for placement into memory. The STORER indexes the new plan by the goals it achieves and the problems it avoids. Now the plan can be used again in similar circumstances in the future.

While the REPAIRER is repairing the plan, the ASSIGNER is deciding which input features interacted to cause the failure to occur. Once it has done this, it marks these features as predictive of the problem so that the
6.4. BUILDING IT FROM THE BOTTOM

ANTICIPATOR can anticipate the problem if it encounters the goals in a later input.

This planner does two things as it builds a plan. It is trying to satisfy a set of goals using its model of what plans are appropriate for different situations. But it is also testing that model of the appropriateness of those plans against the real world so that later planning will be easier and more reliable.

To serve its first function, the planner has to react to failures by repairing the present plan. To serve its second function, it has to alter its view of the world by adding new plans indexed by the problems they solve and by altering its predictions so that it can anticipate those problems and find the plans that avoid them.

The fact that planning and learning are so intimately connected in case-based planning is no accident. The power of a case-based planner is directly dependent on its ability to reuse plans, and the only way to reuse plans effectively is to take seriously the notion of learning which features in a planning situation determine when they are appropriate to use.

A case-based planner learns by remembering plans and deciding which features are important for indexing plans. The learning is not in the building of plans but in the storing and indexing of them.

A case-based planner learns by correctly indexing its planning experiences in memory.

Memories of past planning experience can also be helpful to a planner in dealing with the situation in which it predicts a failure but is unable to find a plan that deals with it. In these situations, the memory of a repair used to fix a past plan with a similar failure can often be used to repair the current plan before it is run. plan is being devised. Of course, not all repairs can be transferred between plans, just as not all old plans can be transferred to new situations.

In all three of these learning situations, the main task of the learner is to figure out which features a piece of information should be indexed under. Any generalization that is done takes the form of generalizing these features without changing the specificity of the information being stored. The plans that are placed in memory are the actual plans created by the planner. They are not generalized versions of these plans. The goals that are used to index them, however, are generalized so that the plans can be found in situations that are similar if not identical to those in which they were originally constructed.

The theory of case-based planning presented here is a theory of learning in each of these three areas. It is a theory of learning new plans, new problems