Planning and Goal Interaction:

The use of past solutions in present situations

Kristian J. Hammond

Department of Computer Science
Yale University
New Haven, Connecticut 06520

Abstract

This paper presents WOK, a cased-based planner that makes use of memory structures based on goal interactions. WOK generates original plans, (which take the form of recipes), in the domain of Szechuan cooking, by modifying existing plans that are stored and then retrieved on the basis of the goal interactions that they deal with. The paper suggests an organization and indexing strategy that allows the retrieval and use of plans that overarch sets of goals rather than just individual goal situations. It also demonstrates how episodic knowledge can be used to guide planning and avoid past failures.*

1 Introduction - Anyone can plan

At this point in time, nearly anyone in Artificial Intelligence can write a planner. The basic ideas exist as part of the knowledge we have as a field. The ideas that goals have plans associated with them; that some plans are stored as sets of coordinated steps or sub-plans; and that the achievement of most goals is in fact the achievement of many sub-goals all exist as givens in the field. What does not exist is the knowledge of how to deal with planning situations in which multiple interacting goals have to be planned for. While much work has been done in the identification of goal interactions such as CONFLICT, COMPETITION and CONCORD, [7], little to date has been done to incorporate these ideas into the knowledge system of a planner. The following then is a description of WOK (Well Organized Knowledge), a planner that makes use of goal interaction knowledge to categorize and organize plans in terms of the interactions that they deal with. WOK generates original plans in the domain of Szechuan cooking, by modifying existing plans that are stored and then retrieved on the basis of goal interactions specific to the domain (*e.g.* CONFLICTING-TASTES:ONE-DOMINATES and CONTRASTING-TASTES:EQUAL-FOOTING) rather than on the basis of specific contextual goals that the plans meet (*e.g.* the goal to include chicken or the goal to avoid hot tastes). WOK's processing is guided by the top down application of its memories of past successes and failures, stored in terms of the goal interactions that they deal with.

2 Planners and Plan Interaction

The problem of planning has been worked on since the earliest days of AI ([3], [1], [4]), but most of the stress has been on systems that plan for a single goal. Such systems are concerned primarily with means/ends analysis, reasoning about the sub-plans and sub-goals that must be accomplished in order for the top level goals to be satisfied. Some such systems have included critics or constraint mechanisms to deal with goal interactions such as goal conflict and goal subsumption [4] or include explicit references to particular interactions in their rule bases [6]. In the case of Sacerdoti's NOAH, the planner had to create a faulty plan before applying a special case critic to correct the fault, while in the case of Shortliffe's MYCIN, each case of interaction had to be coded into a specific rule.

More recently models have been developed that deal not with single goals and plans, but with sets of goals that have to be planned for in terms of their possible interactions ([8], [2]). Bob Wilensky has proposed a model that expands a simple planner to deal with meta-level goals that arise when goal interactions lead to difficulties. When a problem having to do with some goal interaction is noticed, such as a conflict between the goal to get the newspaper on a rainy day and the goal to stay dry, the resolution of the conflict itself is made into a goal and the planner goes to work on trying to satisfy it. Hayes-Roth and Hayes-Roth on the other hand propose a system in which plan interaction is dealt with by starting with a plan and opportunistically performing plans for the satisfaction of active goals as they arise.

Wilensky's model, while it does try to integrate the knowledge of meta-plans with the lower level knowledge concerning simple plans for single goals, still assumes that the planner will first generate individual plans for each goal and then notice the interactions. The planner then deals with meta-level goal interactions (such as the conflict between the paper and the rain mentioned above) by performing a simulation of the various plans for each of the individual goals in the interaction, until it finds a combination that satisfies all of the goals in that interaction. The Hayes-Roths' model addresses positive plan interaction, such as the piggy-backing of plans when the outcome of one plan aids in the satisfaction of some other active goal (as in the performance of one errand when another brought the planner to an advantageous location). It is not capable, however, of drawing on top-down knowledge to deal with the negative interactions such as conflict and competition.

In the main, these systems make use of a search though the space of combinations of individual plan interactions to find a single complex plan that will cover all or most of the goals that the system is trying to satisfy. The final plan is found after either an examination of many of the possible plan combinations, or by simply stumbling over an appropriate plan in the course of execution. While Wilensky does make use of goal interaction information to spawn new meta-goals for his planner to work on, he does not make use of this information to search for plans to achieve his original goals.

---

* This report describes work done at the Department of Computer Science at Yale University. It was supported in part by National Science Foundation grant IST 8120451.

3 WOK - A different sort of planner

3.1 Planning *for* rather than *against* interactions

The WOK project takes a somewhat different approach to planning. The program makes use of its knowledge of goal interactions to organize plans and tries to make use of plans that overarch and satisfy a set of goals rather than find a plan for each individual goal and try to combine them. WOK is designed to be an interactive program that is given a set of constraints by the user and then provides a recipe that meets them. The constraints come in the form of requests for certain ingredients, (*i.e.* chicken, water chestnuts, scallions), particular tastes, (*i.e.* hot, spicy, bland), and textures, (*i.e.* crunchy, gelatinous, chewy). The output is a natural language description of the dish and the recipe that has to be followed to make it. WOK functions by finding an existing plan or recipe that in some sense fits its current set of goals, and modifying it to provide a plan that precisely matches those goals. WOK does not ever try to build up a recipe from scratch. It instead is reminded of old recipes that deal with situations analogous to the one it is working on and alters them to fit its current needs.

3.2 Knowledge and the organization of knowledge

WOK begins with a knowledge base of recipes; descriptions of ingredients; descriptions of physical scenes that correspond to preparation and cooking steps such as CHOP or STIR-FRY as well as a set of primitive alteration steps that allow it to ADD, REPLACE and REMOVE ingredients from a recipe. Recipes take the form of sequential orderings of preparation and cooking steps which include listings of all ingredients used in the plan. Beyond this, the initial recipes include information pointing out the interesting tastes and textures that result from the execution of the plan (this normally includes the major ingredients as well as any strong spices) and information about any interesting interactions of tastes that occur in the recipe (*i.e.* the contrast between the savory taste of pork and the sweet taste of hoisin in PORK SHREDS WITH HOISIN or the conflict that is dealt with between the licorice taste of star anise and the salty taste of the cooked down soy sauce in ANISE CHICKEN).

Recipes stored in memory are indexed in three different ways. Each is indexed under the important foods and tastes that are included in the recipe, making it possible to retrieve a recipe for an individual ingredient or taste, such as CHICKEN or GINGER. Recipes are also stored under the individual preparation and cooking steps that they make use of. This makes it possible to retrieve recipes on the basis of some particular step that might be important in the current situation, such as finding a recipe that includes the MIX and FORM steps necessary to any ground meat dish.

These two means of indexing, while useful in the case of single goal requests such as those for *just* chicken or *just* ground beef, are of little value when a set of taste and ingredient goals are to be satisfied. Once the system has to deal with goals in interaction, it is important that it be able to retrieve plans on the basis of something other than a single ingredient or step. Given a request for a dish that includes chicken and oranges for example, it is less important to look at all of the chicken dishes or all of the recipes that have oranges, than it is to find a recipe that has already confronted the problem of combining a savory and sweet taste of equal strength. Because of this, all recipes are indexed under the taste interactions that they deal with. Thus PORK SHREDS WITH HOISIN is stored under a structure that contains knowledge about contrasting tastes where both tastes are of equal strength (CONTRASTING-TASTE:EQUAL-FOOTING) because it deals with a situation in which that interaction has been incorporated into a successful plan, in that it includes both the savory PORK and the sweet HOISIN. ANISE CHICKEN is stored under a similar structure for conflicting tastes where one taste overpowers the other (CONFLICTING-TASTE:ONE-DOMINATES) because it deals with this interaction by including both the salty taste of SOY-SAUCE and the licorice taste of STAR-ANISE. This indexing is by far the most important of the three when dealing with interacting goals in that it allows the use of recipes that solve problems which are are analogous to a current processing situation rather than just those that include the same ingredients or tastes.

3.3 The structure of interactions

WOK uses the categories defined by taste interactions as knowledge structures under which are stored plans that have been developed to cope with the interactions themselves. Rather than starting with a set of goals and planning for each until some goal interaction blocks normal processing, WOK begins by trying to identify the interactions between the goals it is planning for and searching for plans designed to deal with those interactions. The knowledge structures used are domain specific versions of Schank's Thematic Organization Packets or TOPs [5] and owe a great deal to the categorization proposed by Wilensky [7]. The categories themselves are made up of two components. First there is the relationship between the tastes alone, which includes interactions such as CONTRASTING-TASTE, CONFLICTING-TASTE and AGREEING-TASTE. There also is the effect of the interaction, which includes ONE-DOMINATES, EQUAL-FOOTING and BLENDING.

WOK makes use of nine separate categories of interaction and effect. The more important of these includes:

- **CONTRASTING TASTE : ONE DOMINATES (ex. hot and savory)**
- **CONTRASTING TASTE : EQUAL FOOTING (ex. sweet and sour)**
- **CONFLICTING TASTES : ONE DOMINATES (ex. soy sauce and hoisin)**
- **CONFLICTING TASTES : EQUAL FOOTING (ex. garlic and hoisin)**
- **DIFFERENT TASTES : BLEND (ex. garlic and nuts)**
- **DIFFERENT TASTES : BALANCED TASTES (ex. pork and trees ear)**

Each of these structures organizes four sorts of planning information relevant to the particular interaction.

- **Specific plans that deal with the interaction (*i.e.* particular recipes that handle taste conflicts or instances of an over abundance of agreeing tastes).**

- **General strategies for dealing with the interaction (*i.e.* add a contrasting taste to undercut a conflict between two others, spice up a basicly homogeneous dish that is weak or add a dull tasting buffer to a dish that has become too strong).**

- **Indexing information about what aspects of the plan are important to the interaction and should thus be used in storage and retrieval (*i.e.* index by the dominating taste in the interaction or presence of other interactions).**
- **Instances of failures of plans and strategies that are used to avoid similar mistakes. (*i.e.* Recipes that simply didn't taste particularly good)**

4 WOK - An example

4.1 The basic algorithm

The processing in WOK can be broken down into four major steps. First, the system must sort the goals given to it by the user and identify the taste interactions that will be useful to it in finding a plan to deal with that combination of goals. Second, it uses the abstract interaction, and the particulars of the goals to find a plan that overarches a major segment of the goals as well as the effects of their interaction. Third, the plan is modified to fit all of the current user goals. Fourth, the interactions of new tastes that have resulted from the modifications are checked for their similarity to past failures, and are changed if necessary. Once this is done the new recipe is indexed into the existing data base in terms of the new interactions that are dealt with by the plan. What each of these steps actually means can be best seen through a look at excerpts from one example of the system trying to build a recipe for the user request of a dish with CHICKEN and MANDARIN ORANGES. In the rest of this paper, boldfaced type will be actual output from the program itself.

4.2 Sorting goals and identifying interactions

The goals that the system is given are sorted by their relative importance in the present data base. Given that each recipe begins with knowledge of which tastes and ingredients are important, as well as which participate in the interactions that the recipe handles, it is easy to assess the relative importance of two ingredients or tastes by a comparison of the extent to which they have been important in the past. The importance of a taste or ingredient then is a dynamic value that is the function of how many recipes it is included in, how often it is considered important to those recipes by itself, and how often it is important when interacting with other tastes in those recipes. This last aspect of a taste or ingredient's importance tends to be the most significant, in that most of the indexing is done in terms of interactions, and a taste that has many interesting interactions with others will provide a richer set of indexing possibilities. For example, the hot taste of RED-PEPPER interacts with many more tastes in the recipes that the system begins with, than the taste of CHICKEN. A user goal to have RED PEPPER then would be considered more important that one to have CHICKEN.

Once the goals are sorted, the most important ones are compared in a pairwise fashion, and the interaction between them is found. This is done by a simple table look up of the combination of tastes associated with the goals. In the example of the chicken and oranges, the interaction is CONTRASTING-TASTE:EQUAL-FOOTING, which is to say, the tastes contrast, and they are of the same intensity. This memory structure is then searched for a possible plan, because it is under here that that plans dealing with the sort of interaction exemplified by the relationship between the chicken and orange are stored.

4.3 Indexing and search

Once an interaction is identified, it guides the processing that follows. In the case of CONTRASTING-TASTE:EQUAL-FOOTING, the structure itself knows that both of the tastes are important to use in indexing, and that acceptable recipes include those that match both tastes in the current interaction or those that match one ingredient of the interaction but not both tastes. This contrasts with the structure CONTRASTING-TASTE:ONE-DOMINATES which knows that the dominant taste is more important than the dominated and that this must be matched for a recipe to be used. In the case of the CHICKEN and ORANGE example, the recipe that is found, PORK SHREDS WITH HOISIN SAUCE, is indexed by the tastes alone, savory and sweet, under the interaction CONTRASTING-TASTE:EQUAL-FOOTING.

The goals are ranked as follows:
Goal to include chicken
Goal to include mandarin orange

Searching for recipes including direct relations between ingredients and tastes.

Searching CONTRASTING-TASTE:EQUAL-FOOTING

I have found PORK SHREDS WITH HOISIN SAUCE.
The recipe includes: Pork, scallion, soy sauce and hoisin.

The recipe was chosen because the relation in which the savory taste of the pork contrasts with the sweet taste of hoisin sauce is similar to the relation between the two goals to include chicken and to include mandarin orange .

4.4 Modification of plans

Once a recipe is found, it has to be modified to cover the goals of the original request. As in the case of this example, the recipe may not include any of the ingredients requested, though it does handle the effect of the interaction between them. Given that the system knows how it found the recipe, it also knows what the mapping between the original goals and the existing plan is. The fact that it found the recipe because of the similarity between the interaction of the PORK and HOISIN and that of the CHICKEN and ORANGE gives it the information that the CHICKEN and ORANGE map directly onto the PORK and HOISIN. This information allows it to know to do a simple replacement in this case. In cases where a new ingredient does not satisfy all of the important goals of the one being removed the system has to add a further ingredient that will satisfy the now active goal.

The following goals still have to be met:
Goal to include chicken

Goal to include mandarin orange

Mapping chicken to pork...Replacing
Mapping mandarin orange to hoisin sauce...Replacing.

4.5 New interactions and avoiding past failures

Once a new recipe is built, the system examines the new taste interactions that have developed, in an effort to avoid any mistakes that it has made in the past. It does this by looking at the interactions between the important tastes in what remains of the original recipe, (as stated before, these are explicitly noted in the recipe itself), and the important tastes, (in the global sense of having been important in past recipes), that have been added. The purpose of this step is twofold. First, the new interactions have to be understood by the system so that it can index the new recipe in terms of them. More importantly for the recipe itself, the system is also making sure that no new interaction has come about that is similar to any past failure. Failures are actual recipes that do not succeed in dealing with the interactions they include. If such a failure is found (by going though the same sort of search that was used to find the original recipe, but with the interactions in the current recipe rather than the user goals), then the interaction associated with it is again used to provide a new plan to deal with the problem. In this example, a problem is discovered with the new ORANGE and the old SCALLION, and the resulting interaction, CONFLICTING-TASTES:ONE-DOMINATES provides an alternate plan of finding a new dominating taste for the ORANGE. The new taste is found by looking for a contrasting taste in any recipe, but the search is guided by the context of the current situation.

Thinking about the relations between the following tastes:
The savory taste of the chicken.
The sweet taste of the mandarin orange.
The fresh, vegetable taste of the scallion.

There is a problem with the relationship between the Mandarin orange and the Scallion. I am reminded of the failure in the ORANGE AND OLIVE SALAD.

In the ORANGE AND OLIVE SALAD the relationship between the Onion and the Mandarin orange was considered to be a failure.

In this recipe -
The fresh, vegetable taste of the onion contrasts with and dominates the sweet taste of the mandarin orange.

The plan used in this recipe - STAND-ALONE - is the same as the plan in the failed recipe.

Trying general plans from TOP
CONTRASTING-TASTES:ONE-DOMINATES

TOP has 2. plans: STAND-ALONE and REPLACE-OB1

Plan STAND-ALONE has already failed in the recipe for ORANGE AND OLIVE SALAD.

Avoiding past failure - Trying REPLACE-OB1 -
This plan is to replace the dominating taste with another. The relation is preserved, with a new taste.

Plan is to replace the Scallion.

Looking for item in TOP
CONTRASTING-TASTES:ONE-DOMINATES

Found a possible contrast - Red pepper.

In CHICKEN WITH PEANUTS -
The hot and spicy taste of the red pepper contrasts with and dominates the savory taste of the chicken.

Replacing the Scallion with Red pepper.

Once all new interactions have been validated, the new recipe is indexed in the data base in terms of those interactions.

5 Conclusions

By organizing plans by the goal interactions that they deal with, the WOK planner is able to access and use complex plans that overarch a set of goals, rather than just meet single goals. The memory structures related to these interactions are able to guide the planning process in many ways. They provide instances of both successful plans and failures, general strategies for dealing with the related goal interaction and indexing information which guides search and the later mapping between current problems and past solutions. This information allows the system to find appropriate plans on the basis of goal interactions, modify them to meet current constraints, avoid repeating past failures and make use of a various number of general planning strategies that are organized applicable to the specific interaction. The system avoids the use of special purpose critics and time consuming simulations by anticipating the interactions between goals and finding an overarching plan rather than waiting for the problems of interactions to interrupt processing.

References

[1] Fikes, R., and Nilsson, N.J. "STRIPS: A new approach to the application of theorem proving to problem solving." *Artificial Intelligence 2, 1971.*

[2] Hayes-Roth, B., Hayes-Roth, F. "A Cognitive Model of Planning" *Cognitive Science 3*(4), 1979.

[3] Newell, A., Shaw, J.C., and Simon, H.A. "Report on a general problem-solving program" In *Proceedings of the International Conference on Information Processing.* UNESCO, UNESCO House, Paris, 1959.

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

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

[6] Shortliffe, E.H. *Computer-based medical consultations: MYCIN.* American Elsevier, New York, 1976.

[7] Wilensky, R. *Understanding Goal-Based Stories.* PhD thesis, Yale University, 1978. Research Report #140.

[8] Wilensky, R. *META-PLANNING.* Technical Report M80 33, UCB College of Engineering, August 1980.