Goal Conflict Concerns

Marc

Computer Science Division
University of California at Berkeley
Berkeley, CA 94720
U.S.A.

Computer Science Department
Technion, Israel Institute of Technology
Haifa
Israel

Abstract

A major problem for a knowledge based planner is the determination of potential plan failures due to goal conflict. Given a potential plan, it is computationally intractable to compare every effect of that plan against all planner goals. A commonsense planner named KIP, Knowledge Intensive Planner, is described. KIP determines user plans for the UNIX Consultant System, KIP uses goal conflict concerns to focus attention on those effects and goals which are likely to cause goal conflict. A concern refers to those aspects of a plan which should be considered because they are likely sources of plan failure. Violated Default Concerns allow the planner to use default knowledge effectively in cases where the planning situation has not been specified completely. Unintended Effect Concerns allow KIP to plan effectively when plans are used in novel situations. User goals are often only inferred in response to a threat from a potential plan. A new concept termed interests, is introduced. Interests are general states that KIP assumes are important to the user. While interests refer to general states of the world, goals refer to a concrete state of the world.

1. Introduction

Human problem solving is a knowledge intensive process. People know much about many plans, and also know about the many ways which these plans can fail. However, people have the uncanny ability to consider only a small pan of this planning knowledge while problem solving. They select one or two potential plans to solve current problems among the many plans they have used in the past. They can determine which aspects of a selected plan might cause plan failure in the current planning situation without examining every possible problem that might occur. Early planning programs (Fikes71, Ncwell72, Sacerdoti74) could consider every possible plan failure, since they planned in a domain with a limited number of plans and goals. However, a brute force approach is not computationally tractable in a knowledge rich domain.

A planner which is able to effectively use a large body of knowledge about a knowledge rich domain is called a *commonsense planner* (Wilensky83). Such knowledge includes a general understanding of planning strategy, detailed descriptions of plans, the conditions necessary for these plans to execute successfully, and descriptions of potential goal conflicts these plans might cause.

In this paper, I describe a commonsense planner called KIP, Knowledge Intensive Planner. KIP is being developed for UC, the UNDx Consultant system (Luria85, Wilensky 84a, 86). This planner has a large knowledge-base of information bout the UNIX operating system. The parser and goal ualyzer (Mayfield 86) of UC pass KIP a set of goals, and KIP tries to find appropriate plans for those goals. UC is a conversational system, and if necessary KIP can query the user for more information. Nevertheless, KIP tries to provide the best plan it can with the information provided by the user. This plan is then passed to other UC components, which use the plan to generate a natural language response to the user.

There are two types of plan failure: (1) *condition failure* - plan failure due to an unsatisfied condition necessary for the plan to execute and (2) *goal conflict failure* - plan failure due to the effect of the plan conflicting with a goal. In this paper, we describe how potential plan failures due to goal conflicts between an effect of a potential plan and a goal are detected. (For more on condition failure detection see Luria86, 87). Detecting failures due to goal conflict is more complex than detecting failures due to condition failure. A potential plan might fail due to a limited number of conditions. In contrast, any of the effects of the same plan could potentially conflict with one of the many explicit and background goals of an agent. Since a commonsense planner is faced with a combinatorial explosion of potential goal conflicts, it cannot consider each potential goal conflict as a source of plan failure. Therefore, an algorithm is needed to limit those potential goal conflicts which should be considered.

In the next section, we provide an example of a plan which causes a goal conflict. This example reflects the difficulty in the goal conflict detection problem. We then examine the properties of an algorithm that addresses this problem and describe those issues that must be addressed by any algorithm that detects goal conflicts. In the following section, an algorithm for detecting potential goal conflicts is described.

2. An Example of Goal Conflict Detection

When given a set of user goals, KIP should detect any conflicts between goals the user has specified and other user goals of which KIP is aware. Additionally, KIP creates a potential plan for the user's goals. KIP should detect goal conflicts between the effects of the potential plan and other previously unconsidered goals of the user. Effects can conflict with other goals by causing states which are incompatible with these goals, or by making plans to achieve these goals impossible. The problem of detecting goal conflicts is difficult, since KIP must consider conflicts between the effects of the plan, and the explicit and background goals of the user. If there is a conflict, KIP should try to resolve the conflict. Conflict resolution may occur by either modifying the plan, changing the order in which plans for various goals are executed, or choosing a new plan. For example, suppose the user asks the following question:

(a) How do I move a file named junk to a file named filel?

KIP might select the USE-MV-COMMAND plan. KIP creates an individual instance of this plan for the particular problem situation of moving the file named junk, say the USE-MV-COMMANDI plan. This plan is to execute the command mv junk filel. KIP needs to examine the USE-MV-COMMAND1 plan in order to detect any goal conflicts between an effect of the USE-MV-COMMANDI plan and one or more of the goals of the user, both explicit and inferred.

For example, KIP might detect the following potential problem with the USE-MV-COMMAND 1 plan: One effect of this plan is that if filel exists, it will be overwritten. Let us call this result the *destination file deletion effect.* This effect conflicts with the user's background goal of having access to his file named filel. There is a conflict between the destination deletion file effect of the USE-MV-COMMAND1 plan and the user's goal of having access to the file named filel. This conflict occurs because once the USE-MV-COMMMANDI plan is executed, the user will no longer be able to access the file.

Detecting the goal conflict is difficult since KIP should know about a number of potential effects of USE-MV-COMMANDI. For example, KIP might also consider the following effects of the USE-MV-COMMANDI plan:

If the file named junk does not exist, the message is printed:
mv: junk: Cannot access: No such file or directory

The protection of the file named junk is the same as that of the file named filel.

If the user does not have permission on the current directory, the message is printed:
mv: junk: rename: Permission denied.

If the file named filel is write protected, the user is asked:
override protection 444 for filel?

KIP might consider the following effects that the USE-MV-COMMANDI plan inherits from its parents in the hierarchy of plans:

The directory inode will be updated.

The disk arm will move due to a directory update.

KIP also needs to know about many other background goals of the user that could conflict with these effects. For example, KIP might also consider these background goals of the user:

Try to limit disk space usage.

Execute commands in a way that maintains a low load average.

Have a small number of files in each directory.

Keep your password secret.

None of these background goals of the user conflict with the effects of the USE-MV-COMMANDI plan.

3. Exhaustive Search as an Algorithm for GCD

The *goal conflict detection problem* (GCD) refers to the problem of detecting those goal conflicts that require goal conflict resolution. This problem occurs when KIP has created a potential plan for a goal of the user. KIP should determine if any effect of this plan might cause a conflict with a goal of the user. Here we examine the properties of an algorithm which addresses the GCD problem.

A weak method for solving the goal conflict detection problem would be to compare every potential effect of a particular plan with every explicit and background goal of the user. According to this method, if KIP knew about 5 effects of a plan and knew about 50 explicit and background goals of the user, KIP would need to make 250 tests for conflict. However, exhaustive search is inefficient in a knowledge rich domain. In order to avoid checking for conflicts between every effect of a plan and every potential goal of a user, we require some knowledge efficient method of identifying which potential conflicts should be considered.

Secondly, a human planner usually knows right away that a certain plan will fail. A human planner does not appear to consider every possible goal of the user as a possible source of conflict with every effect of a plan. Rather, the human planner considers only those effects and goals which will often cause the plan to fail.

Thirdly, KIP may not be aware of the values of many of the conditions of a particular plan. Most previous planning research assumed that the values for all the conditions is known. However, in UC, and most other knowledge rich domains, when a user describes a planning problem which is then passed to KIP, the values for many conditions are usually left out. All users would believe that normal goals, like the user wants to preserve the contents of his files, would be assumed by the consultant. A naive user might not be aware of many of the effects of UNIX commands that require a more sophisticated knowledge of UNIX. An expert user would believe that the consultant would make certain assumptions requiring this more sophisticated knowledge of UNIX. It would be undesirable to prompt the user for this information, particularly for those values which are not important for the specific planning situation.

Therefore, KIP should rely on default situation knowledge in order to detect potential goal conflicts. For example, KIP might have knowledge that unless there is information to the contrary, it is likely that a file has read and write permission. If exhaustive search is used as an algorithm for GCD, a comparison of every effect with every goal might be difficult due to the KIP's dependence on default knowledge. Since much of the knowledge about a particular situation may be unknown, each of the individual comparisons for conflict might entail much effort and uncertainty.

Fourthly, many goals of the user are inferred by KIP, rather than being described by the user in a particular problem situation. These goal inferences are based on KIP's long-term knowledge about the user's long term *interests* in the general state of the world. *Interests* are general states that KIP assumes are important to the user. An interest differs from a goal in that one can have interests about general states of the world, while goals refer to a concrete state of the world. For example, preserving the contents of one's files is an interest, while preserving the contents of the file named filel is a goal. KIP's knowledge base includes many interests that KIP assumes on the part of the user. Goals are generated only when expressed by the user, or by KIP itself during the planning process.

The reliance on long-term knowledge about interests makes exhaustive search for goal conflicts difficult. An individual goal is often only inferred by KIP when there is some action that might threaten an interest of the user. If KIP were to compare every effect of a plan with all the goals of the user, it would first need to examine each interest of the user. It would also need to determine if an individual goal should be inferred in the situation due to the particular interest. Checking each interest would be a very inefficient use of knowledge. Very few of these interests will give rise to goals in a particular situation, and even fewer will become causes of goal conflict.

Therefore, a knowledge efficient algorithm that addresses the GCD problem should consider only a limited number of potential goal conflicts and ignore other potential goal conflicts completely. Such an algorithm should utilize default knowledge and instantiate user goals when user interests are threatened.

4. Concerns

In the previous sections, we have described the difficulty and importance of the Goal Conflict Detection problem. Therefore, we have introduced a new concept in KIP, called a *concern* which addresses this problem.

A concern refers to those aspects of a plan which should be considered because they are possible sources of plan failure. A concern describes which aspects of a plan arc likely to cause failure.

There are two major types of concerns, *condition concerns*, and *goal conflict concerns. Condition concerns* refer to those aspects of a plan that are likely to cause plan failure due to a condition of the plan that is needed for successful execution. (These are fully described in Luria86, 87). *Goal conflict concerns* refer to those aspects of a plan which are likely to cause plan failure due to a potential goal conflict between an effect of a plan and a goal of the user. Goal conflict concerns are represented as three way relations between the selected plan, the conflicting effect, and the threatened goal.

The conditions about which KIP is concerned are always conditions of a particular plan. Goal conflict concerns, however, relate plans to user goals and to other pieces of knowledge that are not part of the plan. Examples of this knowledge include background goals which may or may not be threatened by the plan. Since these background goals are usually not instantiated until such a threat is perceived, goal conflict concerns often refer to conflicts between a potential plan and an interest of the user. *Stored goal conflict concerns,* refer to concerns about conflicts of interest. These are concerns about the selected plan conflicting with an interest of the user. If KIP detects a conflict-of-interest concern, then KIP must determine if it should infer an individual goal on the part of the user that reflects this interest. If KIP decides to infer this individual goal, then a *dynamic concern* between the selected plan and the individual goal is also instantiated.

Some goal conflicts are more likely to occur than other goal conflicts, and some goal conflicts are more important than others if they do occur. The representation of goal conflict concerns reflects this difference by assigning a varying degree of concern to the stored concerns in the knowledge base. There are many factors that determine the degree of concern about a conflict-of-interest. The planning knowledge base designer needs to determine how likely a conflicting effect is to occur, how likely it is that the user holds the threatened goal, and how important this goal is to the user.

In the present implementation of KIP, information regarding concerns of potential plans is supplied by a human expert with a great deal of UNIX experience. In principle, however, the information might be supplied by an analysis of data of actual UNIX interactions.

5. An Example of Goal Conflict Concerns

Before describing the other types of goal conflict concerns, and KIP's complete algorithm for dealing with these concerns, we consider a simple example of the use of goal conflict concerns. We describe the order of the steps of this algorithm. Suppose the user asks the following question:

(b) How do I change my password?

KIP is passed the goal of changing the password of the user. KIP's knowledge base contains a stored plan for the goal of changing a password, namely, the USE-PASSWD-COMMAND plan. In addition to the stored condition concerns for this plan, KIP identifies one stored conflict-of-interest concern for this particular plan, the *password authentification* problem. If the user changes his password on one machine, it may not be changed on another machine. As the effect of the USE-PASSWD-COMMAND plan, the user's password is changed on his current machine. The effect conflicts with the user's interest of having the same password on all machines on which he has an account. Let us call this interest the *identical password interest.* This user interest is a subgoal of the user being able to remember his password.

Concerns are stored between the effect that is likely to cause goal conflict and the interest with which it is likely to cause goal conflict. In this example, the stored goal conflict concern which KIP has identified is a three-way relation between the USE-PASSWb-COMMAND plan, the effect that the password is changed on this machine, and the identical password interest.

KIP next evaluates the identical password interest in this particular planning situation. KIP uses knowledge about the particular user to determine if an individual goal should be inferred in this situation which reflects the identical password interest. For example, if KIP knows specifically that the user has an account only on the current machine, then KIP does not assert an identical password goal for the user. Consideration of this goal conflict concern stops. KIP may have information that the user has accounts on many machines, or may make this assumption based on default knowledge from the user model. In either of these cases, KIP then asserts an individual identical password goal on the part of the user.

KIP next evaluates whether the goal of the user is the intended effect of the selected plan. According to KIP's input, the user has specified that his goal is to change his password. KIP's knowledge base stores the fact that the USE-PASSWD-COMMAND plan is the best plan to accomplish this password-change goal. Thus, KIP detemines that the goal of the user is the intended effect of the plan. KIP assumes that the concerns indexed under the USE-PASSWD-COMMAND plan are not concerns about the intended goal of the plan. Therefore, KIP assumes that the conflict between the password-changing on the current machine and the identical pasword interest is a real goal conflict, not an artificial goal conflict.

KIP next evaluates this potential goal conflict. In this case, the concern about multiple passwords is marked as having a high degree of concern, and therefore a goal conflict is inferred. This potential goal conflict is then passed to the goal conflict resolution mechanism.

6. Taxonomy of Goal Conflict Concerns

In this section, we describe a number of different types of goal conflict concerns. These different types address important issues for goal conflict detection in unique situations. In the following section, we describe the goal conflict concern algorithm. We will describe how these types of concerns are used together in the context of that algorithm.

6.1. Default Concerns vs. Violated-Default Concerns

The previous concerns have all been *default goal conflict concerns.* Default goal conflict concerns are those concerns with which the planner is usually occupied, given a set of defaults used to make assumptions about the world. When these defaults are violated, new concerns arise which I call *violated-default goal conflict concerns.* For example, suppose the user asks the following questions:

(c) How do I print out a file on the lineprinter?

(d) How do I print out a very long file on the lineprinter?

In example (c), KIP selects the USE-LPR-COMMAND plan. Since no defaults have been violated, only default concerns are accessed. KIP finds no goal conflict concerns for this plan, and it is returned as a plan for accomplishing the user's goal.

In example (d), KIP also selects the USE-LPR-COMMAND plan. However, in this example, KIP should also access the violated-default concerns of the plan. Since the size of the default file is usually assumed to be short, the fact that the user has specified that the file is a long one violates a default. Because this default is violated, KIP accesses the violated-default concerns for the plan that reflect this violated default. KIP accesses a conflict between the effect of printing out a long file and the interest of being considerate to other users of system resources.

Thus, KIP always accesses the default goal conflict concerns of a particular plan. A plan's violated-default concerns are only accessed when a default is violated. However, only those violated-default concerns that reflect the violated default are accessed.

Accessing only the concerns that reflect the violated default makes the implementation of violated-default concerns difficult In example (d), for instance, the violated-default concern is detected by accessing a category of plans that manipulate long files. Let us call this category the *long-file-plans* category. Since the default of the file being short is violated, a USE-LPR-COMMAND l plan is created. The USE-LPR-COMMAND l plan is dominated by both the USE-LPR-COMMAND plan and the category of long-file-plans. The individual plan inherits all the concerns of both parents. USE-LPR-COMMAND 1 thus inherits conflict-of-interest concerns from both the USE-LPR-COMMAND plan and the long-file-plans category. KIP knows that concerns arising from the long-file-plans category are violated-default concerns. The relationship between long-file-plans category and USE-LPR-COMMANDl is created in order to reflect the violated default.

One advantage of this implementation is that general concerns about non-default situations can be stored in one general category. For example, the long-file-plans category and its concerns are also used by KIP to detect other similar conflicts. KIP can use this category to detect conflicts regarding the use of system resources when compiling a very large program, typesetting a long paper, or sending a very large file over the network.

There can also be more exacting descriptions of violated-default concerns in a more specific category. For example, sending a 50,000 byte file to the lineprinter might not be considered excessive and therefore would not generate a concern. However, sending the same size file to the laser printer, which takes much longer to print, would generate a concern. Therefore, a specific concern category of plans which print large files on a laser printer would be necessary to represent this information. Thus, if the user wants to print a file that KIP knows is 50,000 bytes long, printing the file on the laser printer would cause a concern. Sending the file to the lineprinter, however, would not cause a concern.

6.2. Intended Effects vs Unintended Effects Concerns

When KIP is unable to find a stored plan that solves the goals of the user, it uses another plan that is used for some other similar goal. For example, suppose the user asks the following question:

(e) How can I free up disk space?

If KIP has no stored plan for this goal, it might select the USE-RM-COMMAND plan to accomplish the goal of the user. As an effect of that plan, disk space of the file that is removed is marked as free. One of the problems with using this plan is that it conflicts with the user's interest in preserving his files. The conflicting effect of using the USE-RM-COMMAND plan on a particular file is that the file is removed. This effect conflicts with user's goal of preserving the individual file. Let us call this conflict the *preservation/removal* conflict. In order to detect this goal conflict, a concern should be accessed between the removal of the file and the preservation of the file. The preservation/removal goal conflict should be passed to the goal conflict resolution mechanism.

However, suppose the user asks the following question:

(f) How can I remove a file?

In example (f), KIP would also select the USE-RM-COMMAND plan in order to satisfy the user goal. The USE-RM-COMMAND plan is defined as a plan in service of the goal of removing a file. In this example, however, the user actually intends to remove a file. Therefore, the goal of preserving this file should not be threatened and the preservation/removal conflict should not be detected. Therefore, in example (f), KIP should not access a concern about the conflict-of-interest between removing a file and preserving a file.

This type of goal conflict between an intended effect and a general interest is termed an *artificial* goal conflict. Artificial goal conflicts should not be passed to the goal conflict resolution mechanism. This problem often occurs when a planner does not properly evaluate the threatened interest with respect to the query goal.

In order to avoid detection of artificial goal conflicts, KIP differientiates between *intended effect concerns* and *unintended effect concerns.* Intended effect concerns refer to conflicts-of-interest in which the selected plan is being used for its usually intended effect. Unintended effect concerns refer to conflicts-of-interest in which the selected plan is being used for some other effect of the plan. KIP always accesses a plan's intended effect concerns. A plan's unintended effect concerns are only accessed when the user goal is not the intended effect of the plan.

In example (f), since the USE-RM-COMMAND is being used for its intended effect, the concern about file deletion is not even considered. In example (e), however the USE-RM-COMMAND plan is being used for an effect other than deleting a file. Therefore, all unintended effect concerns are considered, including the concern that using the USE-RM-COMMAND plan will conflict with the user's goal of preserving his files.

In KIP, if a particular action is used as a plan for more than one goal, two or more different plans are created. For example, once KIP knows that a plan for freeing up disk space is to use the rm command, it creates a USE-RM-COMMAND-FOR-DISK-SPACE plan. This plan has a different intended effect than the USE-RM-COMMAND-FOR-DELETING plan. Therefore, this plan will have different intended effect concerns and unintended effect concerns than the USE-RM-COMMAND-FOR-DELETING plan.

Therefore, when KIP has selected a plan for the goal which it has been intended, KIP must check all the intended effect concerns for that particular plan. If KIP has selected a plan in order to satisfy a goal for which the plan has not been intended, then both the intended effect concerns and the unintended effect concerns must be checked.

6.3. Other Types of Concerns

There are a number of other different types of goal conflict concerns that cannot be fully described due to space limitations. These include:

Expressed Goal Conflict Concerns - dynamic concerns that are expressed by the user or created by the goal analyzer. These concerns reflect the concern of the user that a potential plan may cause a goal conflict.

Effect Goal Conflict Concerns - stored concerns between effects and user goals. These concerns reflect knowledge that certain effects give rise to goal conflicts independent of plans that cause these effects. Previously discussed concerns have all been *plan goal conflict concerns.*

7. KIP's Algorithm for Dealing with Goal Conflict Concerns

In the diagram below I have expanded on those parts of the KIP's planning algorithm in which goal conflict concerns play an important role.

7.1. Concern Retrieval

After KIP detects the goals of the user, it selects a potential plan and creates an instance of that plan. KIP then checks for any violated defaults in the particular planning situation by comparing the values of properties in the planning situation, that have been specified by the user, against the default values for those properties. For each violated default, KIP determines the most specific stored violated default concerns for that violated default. Some violated defaults may generate concerns regarding conflicts due to an effect that is not part of the potential plan. Therefore, the conflicting effects of goal conflict concerns are matched against the effects of the potential plan. KIP discards all concerns whose effects are not effects of the potential plan.

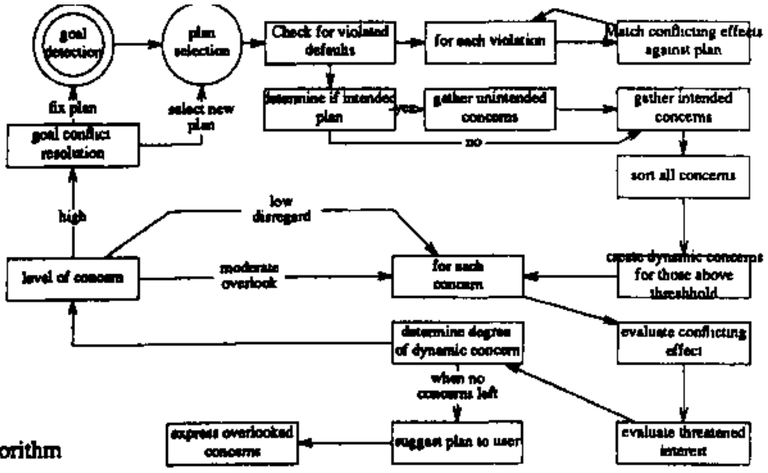
KIP next evaluates whether the user goal is the intended effect of the selected plan. If the goal is not the intended effect of the plan, then both the intended and unintended goal conflict concerns are gathered. If the user goal is the goal for which the plan was intended, then only the intended effect concerns are gathered.

Once intended, unintended and violated default concerns are gathered, it sorts them based on the degree of concern. KIP then decides on a threshold level for concern. This level is based on the planning situation. For example, if the plan is the normal plan for these goals, a high threshold will be chosen. A lower threshold is chosen when the plan has not been used before. The concerns which are below the threshold level are discarded.

7.2. Concern Evaluation

KIP then creates dynamic concerns for each of the stored concerns. It evaluates the concerns according to the degree of stored concern. KIP first evaluates the conflicting effect by determining if the conditions necessary for the effect are true in KIP's model of the world. Secondly, KIP evaluates the threatened interest to determine if the interest is important in this particular problem situation. If KIP determines that the interest is important, than the interest is inferred as a goal. If not, then the concern is disregarded. In either case the interest evaluation is remembered so that other concerns which are related to this interest are not reevaluated.

During this evaluation, KIP assigns a new degree of concern to the dynamic concern based on the particular planning situation. However, many of the values necessary for this evaluation will not be known and must be provided from unc-


goal detection
plan selection
Check for violated defaults
for each violation
Match conflicting effects against plan
determine if intended plan
yes
gather unintended concerns
gather intended concerns
no
fix plan
select new plan
goal conflict resolution
sort all concerns
high
low disregard
level of concern
moderate overlook
for each concern
create dynamic concerns for those above threshhold
determine degree of dynamic concern
evaluate conflicting effect
when no concerns left
express overlooked concerns
suggest plan to user
evaluate threatened interest

KIP's Concern Algorithm

ertain default knowledge. Therefore, the degree of concern of the dynamic concern is calculated by using both the degree of concern of the stored concern and the degree of certainty in the default knowledge. For example, consider a case where KIP evaluates a dynamic concern which has a high degree of concern, and the default knowledge claims that the interest is an unlikely goal. In this case, KIP decides that the degree of concern of the dynamic concern is moderate.

7.3. Concern Treatment in the Planning Process

Once KIP has evaluated a concern it can proceed in one of three ways, depending on the degree of that particular concern. If the degree of concern is *low,* KIP can choose to *disregard* the concern. *Disregard* means that the concern is no longer considered at all. KIP can try to revise other parts of the plan, and suggest the plan to the user with no reservations.

If the degree of concern is *high,* KIP can choose to *elevate* the concern to a source of plan failure. In this case, KIP determines that it is very likely that the plan will fail. KIP tries to fix this plan in order to change the value of this condition, or tries to find another plan.

The most complex case is when the degree of concern is *moderate.* In this case, KIP can choose to *disregard* the concern, or *elevate* it to a source of plan failure. KIP can also choose to *overlook* the concern. Once KIP has developed a complete plan for this problem, it is once again faced with the need to deal with the *overlooked* concern. If the plan will work, except for the overlooked concern, KIP can again choose to *disregard* the concern, or *elevate* it to a source of plan failure. At this point, KIP can also choose to suggest an answer to the user. Depending on the degree of this overlooked concern, KIP may choose to express the concern to the user in the answer.

7.4. Implementation and Representation

KIP is implemented in Zetalisp on a Symbolics 3670. Concepts are represented in the KODIAK knowledge representation language (Wilensky 84b). In particular, knowledge about UNIX commands has been organized in complex hierarchies using multiple inheritance. Therefore, when searching for stored default concerns of a particular plan that uses a particular UNIX command, KIP must search through a hierarchy of these commands. This is also true when looking for default violations, KIP searches up the hierarchy, and retrieves the stored stored concerns or default violations in this hierarchy.

Stored goal conflict concerns are presently implemented by creating a different CONCERN concept for each concern. Also, a 3-way HAS-CONCERN relation is created between each concern, the conflicting effect and the threatened interest or goal which are cause for concern. Degrees of concern are implemented by creating a HAS-CONCERN-LEVEL relation between the particular concern and the degree of concern. Degrees of concerns are presently implemented as numbers from one to ten. Dynamic condition concerns are implemented as instances of these stored concerns.

Defaults are implemented in the current version of KIP by attaching default values of conditions to the plans themselves. Context dependent defaults are implemented by exploiting the concretion mechanism of UC, which tries to find the most specific concept in the hierarchy. Therefore, since KIP retrieves the most specific plan in the knowledge-base, it automatically retrieves the most specific defaults.

Violated default concerns are implemented by creating a different VIOLATED-DEFAULT-CONCERN concept for each violated default concern. A HAS-VIOLATED-DEFAULT-CONCERN relation is added between the concern and the stored default which is violated. Therefore, when KIP has found the default that has been violated, it looks for the violated default concerns that are referenced by this default.

Particular concerns have been entered into the database of UNIX plans through a KODIAK knowledge representation acquisition language called DEFABS. These concerns are all based on my experience using UNIX and on discussions I have had with other UNIX users in our research group. We are currently investigating a way to enter this concern information using the UCTeacher program (Martin, 1985) a natural language knowledge acquisition system. Eventually, KIP may incorporate a learning component that would allow KIP to detect the frequency of certain plan failures and to store these as concerns.

8. Trace

```
User: How do I move a fil« named junk.lisp to a file
named filel?
KIP is passed: goal of moving file junk to the file
filel
file junk.lisp belongs to user
file junk.lisp is a lisp file
file filel belongs to user
Selecting plan USE-MV-COMMAND
Creating USE-MV-COMMAND1
Checking against defaults
junk.lisp is a lisp file violates default file as
text file
violated default concerns:
lisp file is in proper format for current lisp
no violated default concerns match conditions of
USE-MV-COMMAND1
Determining if USE-MV-COMMAND1 is used for its
intention
goal of moving file junk to the file filel
is intention of USE-MV-COMMAND1
Skipping unintended concerns
Gathering intended concerns
filel will be deleted (5)
Sort concerns
All above threshold
Create dynamics concerns
concernl filel will be deleted
evaluating concernl
default knowledge - filel probably does not exist
concernl - value 3
overlook concernl
plan works except for concernl
pass concernl to expression mechanism
UC:  To move the file junk to the file filel,
type mv junk.lisp filel.
However, if filel exists it will be deleted.
```

9. Relationship to Previous Research

Early planners such as STRIPS (Fikes71) did not address Goal Conflict Detection as a separate problem. Conflicts were detected by the resolution theorem prover. The theorem prover compares a small set of add or delete formulas, and a small set of formulas that described the present state and the desired state of the world. If an action deleted the precondition of another action in the plan sequence, backtracking allowed the planner to determine another ordering of the plan steps. ABSTRIPS (Sacerdon'74), modified STRIPS to avoid these interacting subgoal problems by solving goals in a hierarchical fashion. Conflicts in ABSTRIPS were also noticed by the theorem prover. However, since the most important parts of the plan were solved first, they occurred less often and fewer paths were explored.

Sacerdoti's NOAH (Sacerdoti77) program separated the detection of conflicts from the rest of the planning process using his *Resolve-Conflicts critic.* This critic detects one particular kind of conflict, in which one action deletes the precondition of another action. We refer to this type of conflict as a deleted precondition plan conflict. The critic resolves the conflict by committing to an ordering of steps in which the action which requires the precondition is executed first. The ordering of steps is usually possible since NOAH uses a *least commitment strategy* for plan step ordering. By separating the detection of goal conflicts from the rest of the planning process, NOAH needs to search fewer plan paths than earlier planners.

In order to detect conflicts NOAH computes a TOME, a table of multiple effects, each time a new action is added to the plan. This table includes all preconditions which are asserted or denied by more than one step in the current plan. Conflicts are recognized when a precondition for one step is denied in another step. In order to construct this table, NOAH must enter all the effects and preconditions for each of the steps in the plan every time a new step is added to the plan.

NOAH'S separation of Goal Conflict Detection Phase from the rest of the planning process was an important addition to planning research. However, NOAH'S approach is problematic in a number of ways. First, it only detects conflicts that occur as a result of deleted preconditions. Other conflicts, such as conflicts between effects of a plan and other planner goals, cannot be detected using this method. Most of the examples in this paper are part of this category of conflict. If many planner goals were included in a TOME, as would be necessary in real world planning situations, this method would be computationally inefficient. Therefore, the same problems that were discussed earlier in regard to exhaustive search also apply to this method. A TOME is (1) computationally inefficient, (2) not cognitively valid, (3) unable to deal with default knowledge, and (4) assumes that all user goals are known, i.e. would have to evaluate every planner interest in a particular planning situation.

By using concerns, KIP is: (1) computationally efficient - each plan has a relatively small number of concerns regarding potential plan failures, (2) cognitively valid - concerns correspond to the commonsense notion that people can determine which aspects of a selected plan might cause plan failure in the current planning situation without examining every possible problem that might occur, (3) able to deal with default knowledge and consider new problems when certain defaults are violated, and (4) able to consider potential conflicts with long term planner interests and instantiate goals which reflect these interests when they are threatened.

10. Summary

A major problem for a knowledge based planner is the Goal Conflict Detection problem. People only consider a small number of potential goal conflicts. Previous planning programs, however, have accessed all effects of a plan and all potential goals of the user. Goal conflict concerns were introduced in order to address this problem. KIP only needs to access a small number of goal conflict concerns in order to determine if a likely goal conflict exists. Violated Default Concerns allow the planner to use default knowledge effectively in cases where the planning situation has not been specified completely. Unintended Effect Concerns allow KIP to plan effectively when plans are used in novel situations.

11. References

Ernst, G. and Newell, A. 1969. *GPS: A Case Study in Generality and Problem Solving.* New York: Academic Press.

Fikes, R. E., and Nilsson, N. J. STRIPS: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence,* Vol. 2, No. 3-4, pp. 189-208. 1971.

Luria, M. "Commonsense Planning in a Consultant System" *Proceedings of 9th Conference of the IEEE on Systems, Man, and Cybernetics,* Tuscon, AZ. November, 1985.

Luria, M. "Concerns: How to Detect Plan Failures." *Proceedings of the Third Annual Conference on Theoretical Issues in Conceptual Information Processing.* Philadelphia, PA. August, 1986.

Luria, M. "Concerns: A Means of Identifying Potential Plan Failures." *Proceedings of the Third IEEE Conference on Artificial Intelligence Applications.* Orlando, Florida. February, 1987.

Martin, J., 1985. Knowledge Acquisition Through Natural Language Dialogue, *Proceedings of the 2nd Conference on Artificial Intelligence Applications,* Miami, Florida, 1985.

Mayfield, J., 1986. When to Keep Thinking, *Proceedings of the Third Annual Conference on Theoretical Issues in Conceptual Information Processing.* Philadelphia, PA. 1986.

Newell, A., and Simon, H. A. *Human Problem Solving.* Prentice-Hall, Englewood Cliffs, N. J. 1972.

Sacerdoti, E., Planning in a Hierarchy of Abstraction Spaces, *Artificial Intelligence* Vol. 5, pp. 115-135,1974.

Sacerdoti E. *A Structure for Plans and Behavior* Elsevier North-Holland, New York, N.Y. 1977.

Wilensky, R. *Planning and Understanding; A Computational Approach to Human Reasoning.* Addison-Wesley, Reading, Mass., 1983.

Wilensky, R., "KODIAK: A Knowledge Representation Language". *Proceedings of the 6th National Conference of the Cognitive Science Society,* Boulder, CO, June 1984.

Wilensky, R., Arens, Y., and Chin, D. Talking to Unix in English: An Overview of UC. *Communications of the Association for Computing Machinery,* June, 1984.

Wilensky, R., et. al., UC - A Progress Report. University of California, Berkeley, Electronic Research Laboratory Memorandum No. UCB/CSD 87/303. 1986.

Sponsored by the Defense Advanced Research Projects Agency (DoD), Arpa Order No. 4871, monitored by Space and Naval Warfare Systems Command under Contract N00039-84-C-0089.