June 1992

A Reconsideration of Preconditions

Christopher W. Geib
University of Pennsylvania

University of Pennsylvania Department of Computer and Information Science Technical Report No. MS-CIS-92-50.

A Reconsideration of Preconditions

Abstract

This paper is part of an attempt to introduce intentionality of the actor to planning decisions. As a first step in this process the usual representations for actions used by planning systems must be reevaluated. this paper argues for the elimination of preconditions and qualification conditions from action representation in favor of explicit representation of intention, situated reasoning about the results of the action and reactive failure mechanisms. The paper then describes a planning system that has explicit representation and use of intentions and uses action representation that do not have preconditions.

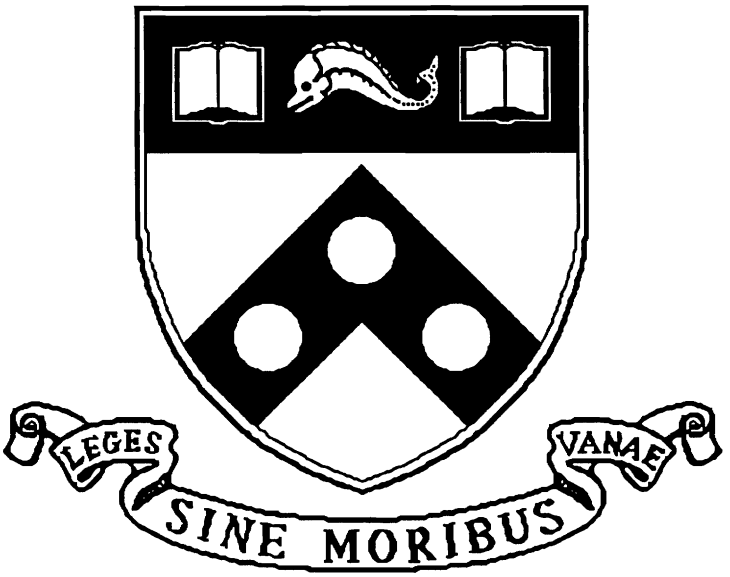
Comments

University of Pennsylvania Department of Computer and Information Science Technical Report No. MS-CIS-92-50.

A Reconsideration of Preconditions

MS-CIS-92-50
LINC LAB 228

Christopher W. Geib



University of Pennsylvania
School of Engineering and Applied Science
Computer and Information Science Department
Philadelphia, PA 19104-6389

June 1992

A Reconsideration of Preconditions

Christopher W Geib

Department of Computer and Information Science,
University of Pennsylvania
200 South 33rd Street,
Philadelphia PA 19104-6389, USA
Internet: geib@linc.cis.upenn.edu

Abstract

This paper is part of an attempt to introduce intentionality of the actor to planning decisions. As a first step in this process the usual representations for actions used by planning systems must be reevaluated. This paper argues for the elimination of preconditions and qualification conditions from action representation in favor of explicit representation of intention, situated reasoning about the results of the action and reactive failure mechanisms. The paper then describes a planning system that has explicit representation and use of intentions and uses action representations that do not have preconditions.

1 Planning and intention

The manner in which an agent chooses to accomplish a given task is often determined not only by the actual action or goal, but also by the agent's other intentions. For example, breaking the door off its hinges is a way of satisfying the goal of having a door open, however, this is not the first way one would think of to achieve that goal.

We can see the role intentions play in our planning deliberations in two ways. First, they can be seen as the ends of our deliberations that is to say, those things that we want to achieve in the future. Second, they can play a constraining role in the manner in which we chose to satisfy other intentions. Bratman[3] gives a description of these two different facets of intentions and their role in planning.

The goal of this work is to try to provide a principled treatment of intentions within planning. Toward this end we must reevaluate the role of preconditions in action representations. Preconditions have often implicitly encoded intentions the agent must hold for an action to achieve its expected effects. For example, in STRIPS[6] the action PICKUP(a) had as a precondition that the block "a" must be clear. This is not logically required for moving blocks; it is not even physically required. This condition must be true only if the agent has intentions not to move other blocks or if the objects on "a" make the action physically impossible for the agent in that world state or if the objects on "a" would cause a result that is undesired in the situation. Notice that all of these possibilities are dependent on the situation in which the action is being taken.

In short, preconditions have encoded situation dependent information (often specifically intentional information) in a universal manner. If we are to have a principled treatment of intentions we must extract these intentions and represent them explicitly. Explicit encoding of intentions and the use of situated reasoning will allow us to eliminate preconditions from action representation and will provide us with explicit intentions to use in making planning decisions. Section two of this paper will give arguments against various kinds of preconditions; section three will describe a system that plans without the use of preconditions. Section four will suggest future directions for this research.

2 Eliminating Preconditions

In the introduction of this paper I suggested that it is possible to completely eliminate preconditions in action representation. First, we must ask ourselves why we want to do this. In the case of this work, the reason is clear. As we saw early in this paper some of the standard preconditions for lifting blocks in the blocks world can be seen as encoding negative intentions about moving other blocks. Since our goal is a clear and uniform treatment of intentionality in planning, it is important for us to be explicit in our representation and use of intentions.

The second question is whether it is possible to eliminate all preconditions. They have a long and successful history in the planning literature, and there are a number of problems that are conveniently solved by them. If we are to eliminate preconditions we must find other solutions to these problems. The next sections will discuss various definitions of preconditions, discuss arguments against them, and sketch other solutions to the problems they solve.

2.1 Preconditions vs. Qualifications

At this point, we need to set out some definitions. *Preconditions* are usually defined as those conditions which must be true before an action is taken. This definition is unclear at best, and so a further distinction is often made, namely the distinction between preconditions and *applicability conditions* or *qualifications* [11].

Traditionally preconditions define when an action can be completed successfully while qualifiers describe the situations in which an action (if completed successfully) is relevant to a goal. The basic difference between preconditions and qualifiers is in how they are treated by the planner. If a precondition for an action is false, an agent might attempt to make it true in order to carry out the action. While if a qualifier for an action is false, the action itself should be considered inappropriate or irrelevant. The next two sections will examine ways in which preconditions and qualifiers define the conditions required for the success or relevance of actions and how they might be better replaced by other mechanisms.

2.2 Preconditions

A precondition might guarantee the success of an action in a number of ways. First, we might say that a condition P is a precondition of an action α in that it must be true before α is taken in order for α to have its intended effects. For now let us leave aside the question of intentionality. Instead, we must ask how do we determine the effects of the action? Any specific instance of an action will have a number of effects; some of these effects happen every time the action is performed while others are only the serendipitous result of the situation in which the particular instance happened.

For example, our moving blocks example. Moving a block that is under another block can have a number of effects. The block on top may fall and break or it may remain balanced and move with the lower block. The question is, when defining the action's effects exactly which of these effects should we consider? Obviously, we do not want to talk about all of the possible results of an action; we would like to be able to say, preconditions define those conditions required for the action to result in its "usual" effects. This however dooms us to the very slippery slope of defining what is "usual."

At this point let us reintroduce the question of "intended" effects. Even if we are able to define the usual results of an action, we must question which of these effects are intended by the agent. Given the history of this debate [3] [5] [7] it is far from clear that we can answer this question. It is certain that, we cannot answer this question without appeal to the intentions of the agent, and this information can only be determined at the time of an action's execution. In our previous example, we can easily imagine instances in which an agent intends each of the possible results. Therefore, in order to complete a definition of this type, we would need to examine the intentions of the agent each time the action is performed. Finally we must ask, do the preconditions reflect only those conditions that must be true for the action to result in only the intended effects? Certainly, the variability in intentions even within one agent makes this type of definition impossible.

Second, we could say state P is a precondition of action α if it is necessary that P hold before performing α in order to guarantee that α not have any unintended effects. This definition has many of the same flaws as the last one we considered. It again calls for us to define all of the effects of the action or at least some fixed subset which are usual.

After determining the usual effects, we must decide which are unintended by the agent. As before, the intentions of the agent can only be determined at the time of the execution of the action, however, the problem of defining

the unintended effects of an action goes a step further than that of the intended effects. Unintended effects fall into two distinct categories, known versus unknown. That is to say, if an effect is not defined as a usual result of the action does that make it unintended or simply unknown?

An explicit representation of intentions will solve this problem. If we have explicit representations of positive and negative intentions, the problem of unintended versus unknown effects is trivially solved. Those negative intentions the agent holds are unintended. This gives us a clear distinction between unintended and unknown effects.

Finally, we might define a state P as a precondition of action α in that P must be true before α is undertaken in order that α can be completed successfully. As an example, a precondition of lifting a block is that the block not be too heavy to lift. That is to say, the action lift cannot be successfully completed if the block is too heavy. It is often the case with these preconditions that the agent cannot know if the precondition holds unless he or she attempts the action, witness the weight of the block or the presence of ink in a pen for writing. How else could we know if the preconditions hold if we haven't taken the action?

This problem of foreknowledge has lead traditional planning theorists to suggest that while we are aware of these sorts of conditions we do not worry about them unless the action fails, for example, the famous "potato in the tail pipe" of McCarthy. These conditions are not really preconditions at all. They are more accurately explanations for the action's failure. Therefore, this information should be used to explain the action's failure after the fact.

2.3 Qualifiers

As we mentioned before, qualifiers are used to determine the relevance of an action. I argue, an action's relevance is determined by the world state and the intentions of the agent rather than any other conditions. That is to say an action α is *relevant* to achieving a state, S, just in the case α can achieve S in the current world state and S is not already true in the world. Other systems have defined an action α to be relevant only if its qualifiers are all true.

As with preconditions, there are a number of kinds of qualifiers, and as before, I will take each in turn and discuss why it can be eliminated. First, a state Q might be a qualification condition of an action α if it is logically required for the performance of α. For example, a logical qualification condition of the action "kill the King of France" is that there be a King of France.

However, notice that this action can be determined to be irrelevant by our definition. Since there is no King of France it is impossible for any action to achieve a state where he is dead. Therefore by our definition there is no relevant action. Therefore, there is certainly no reason to include qualifiers of this kind.

Second, a state Q might be a qualification condition of an action α if Q is a physical qualification on the action. This sort of physical qualification could take on two forms. First, a physical qualifier might be a relevance condition. For example, a qualification condition on unplugging an object is that the object be plugged in to begin with. This means, unplugging is irrelevant if the object is not plugged in. Again this sort of painfully obvious condition is rendered unnecessary by our definition of relevant actions. If the object is plugged in and we want to change that state then any action that achieves it will be relevant. Otherwise we need not consider actions.

However, a second kind of physical qualifier exists. These are similar to the preconditions for successfully completing an action. For example, it must be the case that an object is not irrevocably fixed to the ground if one is to lift it. These sorts of conditions have all of the same properties as their precondition counterparts, and so we will treat them in a similar manner. Since knowledge of the condition of the qualifiers usually requires performing the action, we can again eliminate these conditions in favor of dealing with the failure of the action. It is interesting to note that qualification conditions of this variety may in fact be preconditions where the actions necessary to achieve them are deemed to be too expensive by the agent.

There is a final kind of qualification condition, arbitrary rules. An example from everyday life is traffic regulations. It is not logically required that a traffic light be green before moving through an intersection, and we can all attest to the fact that it is not physically required that the light be green. However, we have a rule that we are not supposed to undertake an intersection crossing action while the traffic light over the intersection is red.

This is certainly the weakest type of qualification condition. As we know, these conditions are regularly ignored by human agents when they believe the reasons for the rule are inoperative. As with most preconditions, these qualification conditions are situation dependent rather than universal in nature, and therefore should be eliminated in favor of situated reasoning using the intentions of the agent.

3 ItPlanS

The Intentional Planning System (ItPlanS) is a first step toward a planner that gives intentions the role in planning they deserve. It is a planning system that currently operates in the blocks world. ItPlanS is similar to hierarchical planners in that it recursively expands intentions, developing lower level intentions until an intention is found that can be immediately satisfied by a single action. This action is taken and the process is repeated. The only structure maintained across each iteration is a list of the top level intentions. This means that the decompositions for a given intention can vary from iteration to iteration allowing ItPlanS to adapt its plan to a changing world environment. Since ItPlanS is only finding the next action based on the world state, the system's knowledge, and its intentions, it might be more accurately viewed as an action theory rather than a planning system. The following subsections give a detailed account of the system's construction and operation.

3.1 Intentions in ItPlanS

In ItPlanS positive intentions are represented as predicates like ON(a,b) or BROKEN(a) from the blocks world. These intentions are kept in an ordered intention list. Since ItPlanS only reasons about a single agent and it will take steps to achieve any intention it adopts, this representation is equivalent to INTEND(ACHIEVE(ON(a,b))).

It is also possible to have negative intentions. That is to say, it is possible to have an intention not to cause certain states to come about. For example, in the blocks world I might not want to break blocks. I can state this as a negative intention; I don't want to cause a state where BROKEN(X) holds for any X. To represent this ItPlanS has a second intention list containing only negative intentions. It is important to notice that negative intentions are described as predicates just as positive intentions are. Negative intentions can be interpreted as INTEND(NOT(ACHIEVE(predicate))). There is however, a question about this interpretation; is this equivalent to INTEND(NOT(HOLD(predicate)))? Notice that if the system is the only agent of change in the world, these two options are the same. It is only in the case where there are other agents in the world that these two diverge in meaning.

The difference is in the licensing of preventive action.
INTEND(NOT(ACHIEVE(predicate))) would normally only prevent the

agent from acting to cause the predicate to become true. The question is, will it license preventive action? Should the system take active measures to prevent the occurrence of some predicate by the action of some other agent? In ItPlanS, I have made the assumption that regardless of the existence of other agents in the world, the system will not take measures to prevent the other agents from acting in a manner that conflicts with its negative intentions. In short, ItPlanS remains agnostic about the existence of other agents, but will not take action to prevent possible interference.

Given this representation, one could argue that existing planning systems are already dealing with intentions; this is only partially true. Usually systems only deal with positive intentions; they do not represent the negative intentions of the agent explicitly, and therefore lack the power of ItPlanS.

3.2 Action representation

Since action representations in ItPlanS have no preconditions, it is impossible for ItPlanS to use traditional backchaining on preconditions to produce plans. Therefore, ItPlanS is designed after the decomposition methods of hierarchical planners, and so makes a distinction between two different types of actions in its ontology, primitive and complex. *Primitive actions* are actions in the traditional sense. They have a physical realization; that is, they perform some operation on the world. *Complex actions*, on the other hand, exist in order to perform the decomposition of the system's intentions into subintentions in the same way that actions in hierarchical planners decompose goals into subgoals. It is also possible to understand the distinction between complex actions and primitive actions as the distinction between actions the system must "reason" about in order to perform versus those actions the system can perform "without thinking."

3.2.1 Primitive Actions

As noted, in ItPlanS primitive actions are the basic actions the system is capable of performing on the world. In the case of the blocks world there are only three of these: GRASP, LETGO, and GOTO. The selection of these actions and this number is arbitrary. The actions GRASP and LETGO have the obvious effects on the end effector and they have no arguments. The GOTO action, on the other hand, moves the robot arm about the blocks world, and does take an argument. This argument specifies the desired location of the end effector at the end of the movement. For example, the

action GOTO(FRONT(a)) moves the end effector to the area in front of the block labeled "a" [1] .

GOTO also has the effect of setting a focus for the special function NEAR, a relative distance measure. Executing a GOTO(NEAR) brings the effector close to the object in focus. Notice there is no argument to NEAR. The argument is obtained from the last GOTO action. This means the series of actions GOTO(FRONT(b)), GOTO(NEAR) would at least have as its consequences AT(FRONT(b)) and NEAR(b).

Each primitive action is defined by two sets of data. The first is a set of likely outcomes of performing the action; we will call these *possible results.* Possible result lists are designed to provide a concise, declarative statement about what the system believes might result from a given action. For example, a possible result of a GRASP action is INHAND(X) where X is a block. It is important to notice that, the possible results of an action may not occur every time the actions does. For example, INHAND(X) is true after a GRASP action if and only if AT(FRONT(X)) and NEAR(X) were true in the world before the action was performed.

The system also has a set of *result rules* which describe the effects the system believes the action will have in given situations. Result rules are designed to provide the information needed to simulate the effects of primitive actions. These rules can be looked at as conditional add and delete lists for the action. Each result rule has three parts. First, the condition, is a list of predicates, the conjunction of which must be true in order for the rule to be applicable in simulating the action. Second, is a list of predicates that should be added to the world model by the action and finally is a list of predicates that should be deleted from the world model as a result of the action. For example, if we take our previous example the system would have a rule that stated that if AT(FRONT(X)) and NEAR(X) are true and a GRASP is executed then INHAND(X) will result. The system should also have a rule which states if INHAND(a) is true then if GOTO(FRONT(b)) is performed then block "a" is no longer where it was.

Since result rules define what the system believes about how actions work in the world, they may be in error. It is not required that they be a correct axiomatization of the laws of physics or that they be limited to those results listed in an action's possible result list. Of course, for any action the add lists of the result rules should be a superset of the possible results, but even

[1] See the section on Specifying locations for a description of the special function FRONT.

this is not required. The distinction between the possible results and the result rules is maintained for exactly this possibility. If the possible results list of an action names a result not listed in one of the result rules, this tells us that the system believes some predicate often follows from an action without knowing the exact conditions under which it will follow.

The objective of primitive actions is to attempt to define a small set of actions from which larger behaviors can be built. The choice of actions as primitive is arbitrary and can certainly be changed. Assumably as an agent becomes more expert at a specific type of problem they will develop more primitive actions to simplify the problem. Note, primitive actions are the only actions that actually effect the state of the world; all complex actions are realized as a series of primitive actions.

3.2.2 Complex Actions

Complex actions are not really actions at all. While they share many things with the primitive actions, complex actions can best be viewed as functions from a single intention to an ordered set of intentions. In effect, they map from an intention to a set of subintentions that need to be achieved in order for the original intention to be achieved.

For example, the representation for the action STACK(X,Y) maps the intention of ON(X,Y) to the list of intentions [INHAND(X), HANDOVER(Y), HANDEMPTY] which, when achieved in order, will result in a STACK operation. Notice, the action's definition does not specify how to achieve any of these subintentions; any way to achieve them will be equally valid. Complex actions also have a possible results list, and again the possible result lists are not required to be accurate. As with primitive actions, arguments to complex actions denote locations in the universe.

As was mentioned above, in ItPlanS, the results of a complex action are seen as nothing more than the sum of the results of the primitive actions chosen to satisfy its subintentions. Since the actions taken can vary from occurrence to occurrence, there is no way to have complete beliefs about the results of a complex action. So while complex actions have possible results, they do not have result rules.

At this point, I will once more state the obvious. None of the actions has any form of preconditions. Any action can be undertaken at any time and at any condition of the world. Of course, it will often be the case that the action will not have the expected outcome if taken randomly, however, this does not preclude performing the action.

3.3 Specifying Locations

In order to provide a uniform treatment of block stacking to arbitrary heights, it was necessary to develop a method for representing space. To this end, I have defined a number of functions from objects in the world to areas of space around them. This treatment is not intended to be complete or rigorous, nor is it to be a substantive part of the work to follow. It is to be seen as a tool to solve some of the problems of designating locations in the world and may undergo serious revision.

The functions I defined are suggested by the English prepositions LEFT, RIGHT, FRONT, BACK, OVER, UNDER; these functions map from the argument/object to the spatial area described. For example, the action GOTO(LEFT(b)) would move the end effector to some arbitrary location to the left of block "b." As a result the world now contains the statement AT(LEFT(b)) reflecting the changed position of the effector.

3.4 The Algorithm

ItPlanS' algorithm works in the following manner. The agent starts each iteration with two lists, one of positive intentions and the other of negative intentions. The agent linearly searches the list of its positive intentions to find the first intention not satisfied in the current world state. If there is no primitive action that has this intention in its possible results list the system finds a complex action that does. Using the complex action's mapping function the algorithm creates a new list of positive subintentions. This list is searched for its first unsatisfied intention and the process is repeated until an intention is found that can be satisfied by use of a single primitive action.

For example, if we look at figure 1, we see that initially in the blocks world we have three intentions. The first of these, ON(a,b) is already true in the world. Therefore, the next intention the system needs to satisfy is ON(c,d) since it is not true in the world already. ItPlanS checks its knowledge base and determines that a STACK action has as a possible result ON(c,d). Since STACK is a complex action, ItPlanS next considers the list of subintentions a STACK action results in namely: INHAND(c), HANDOVER(d), HANDEMPTY. ItPlanS then checks to see if any of these intentions are already satisfied in the world, and discovers that, in fact, INHAND(c) is. Therefore the next intention to be satisfied is HANDOVER(d). Notice, the process continues until the intention NEAR(d) is reached. At this point a primitive action can be used to satisfy the intention namely

GOTO(NEAR).

Notice, the system, in looking for the first unsatisfied intention at each level of decomposition, is constantly reacting to the actual state of the world so that if an accident occurred and the last action performed did not have the consequences the agent expected, the agent can correct it immediately. The agent can also attend to any other intention that has suddenly been invalidated.

Having found an intention that can be satisfied by a single primitive action, ItPlanS selects one primitive action to achieve this intention.[2] The system creates a minimal model of the world relevant to the action. Then ItPlanS simulates the performance of the action on the world by selecting the action's most specific result rule whose condition is true in the world, and adding and deleting propositions from its model as appropriate to derive the model of the world after the action is taken. If the resulting world model does not contain any of the negative intentions of the agent the action is carried out. This completes the selection of one action by the system. The system then repeats this process for the next action, discarding all but the top level of the intentional structure.

It may be argued that the system never commits to a single plan. This could lead to incoherent action in which the system follows one possible plan for a step and then undoes that action. In ItPlanS, the coherence of action relies on the meta-stability of the top level intentions of the system and the relative predictability of the world. I assume that intentions are inherently persistent, and this longevity will prevent trashing of the type described.

In fact, the absence of commitment is a positive side effect for the system. It is this lack of commitment to a single plan that allows the system to utilize positive changes in the environment when they happen without a need to alter the adopted plan in any significant manner.

3.5 Negative Intention Conflict

There are a number of outstanding issues left out of this algorithm. Foremost is what does the system do if all of the explored primitive actions violate one of the negative intentions of the system? ItPlanS has a recovery mechanism for these situations which relies heavily on the accuracy of the action's result rules. What has happened in these cases is all of the examined actions have been determined to cause problems. What the system needs to do is to

[2]It is possible for more than one primitive action to satisfy the selected intention.

decide on the action it would "like" to take and then remove the causes of the problematic result of the action.

For example, suppose block "b" is stacked on block "a" and further suppose that I am grasping block "a" but have not yet moved it. Let us also assume that I have an intention to move block "a" to a new position, and I don't intend to break anything. The only action I have available to me that will reposition block "a" is the action GOTO. Finally, let us suppose that I know that if I perform a GOTO action, block "b" will fall off and break. At this point I am stuck. The only action to achieve my ends will violate my negative intentions. specifically moving block "a" will cause a state where block "b" is broken violating my negative intention about breaking things.

Obviously, in this case, what I should do is determine the cause of this conflict, (in this case it is the presence of a block on top of what I want to move) and eliminate this cause (unstack block "b" from block "a"). In effect, this is exactly the process ItPlanS follows. In the example, there is only one possible action that can achieve my intention. In general, however, there may be many, so the first step ItPlanS takes in this process is to decide on which possible primitive action it will attempt to remove impediments to.

Having decided on an action ItPlanS looks up the result rule that is applied in the simulation to produce the conflict. Our goal in this process is to prevent this result rule from being applied to the model of the world. This result rule was chosen because all of the predicates in the conditional portion of the rule were true in the world. If one of the predicates in the condition of the result rule were false the rule would not be used to simulate the action and therefore the conflict would not arise.

Let us return to our example to see how this works. The result rule that would be applied for this use of GOTO would be something of the form [[INHAND(X), ON(Y,X)], [ON(Y,table), BROKEN(Y)], [ON(Y,X), ON(X,Z)]]. Where the first element is the condition, the second is the add list, the third is the delete list, "X,Y,Z" are all variables, and "table" is a constant. In order for this rule to be applied, it must be the case in the world that INHAND(X) and ON(Y,X) are both true. In our specific example, we know that INHAND(a) and ON(b,a) so the rule is applicable. In order to keep this rule from being applicable to the simulation we need to violate one of the elements of the conditional. That is, we need it to be that case that either INHAND(X) or ON(Y,X) is not true in the world.

At this point ItPlanS chooses one of these conditions to invalidate, inverts the condition and achieves this inverted proposition as a positive intention.

This will remove one of the conditions from the world making the rule invalid. Having done this, the original GOTO action can now be carried out without the undesired consequence of BROKEN(b). Of course, this method requires the agent's knowledge base to contain information about how to invert actions. In our example, the agent must know that the way to undo ON(b,a) is to achieve ON(b,table).

We must also produce a method for deciding on which condition to invalidate. It would be unfortunate if we chose to invert one of the intentions we just achieved when there was another possibility. As a partial solution to this problem, the system maintains a list of those intentions that are "above and to the left" of the current. That is to say, the system keeps a list of all the intentions it has tested in the decomposition process which have been determined to be satisfied. In figure 1. these intentions are underscored. It will obviously be preferable to invalidate an intention that is not on this list, since they are necessary for the achievement of the system's intentions and have been achieved already. Therefore, ItPlanS makes every effort to protect these intentions.

If it is impossible to find an intention that is not on the list of "protected" intentions then one from that list must be selected and inverted. This is the worst possible case as this intention must be marked as special to prevent the system from trashing and undoing the work already done. This method roughly corresponds to goal advancement in traditional planning systems.

4 Results and Future work

I have implemented the ItPlanS system as it is described above. The system performs exactly as we wanted it to do. Unless the system is given as a negative intention BROKEN(X) it will ignore the traditional preconditions of clearing off blocks before picking them up. However, when given the appropriate negative intention the system finds it must clear off the blocks before lifting them, showing that the intentions of the agent and the world state can be used to eliminate preconditions for actions. While the system as it is described does succeed in solving the most of the traditional problems in planning, there are obviously a number of issues that need further study.

4.1 Choices and ItPlanS

One of the most important issues that I want to look at in this work it the role that intentions play in the planning process. If we take Bratman

[3] seriously then intentions play two roles. One of these is constraining possible plans. Bratman talks about making choices about how to get to Tanner library and the fact that his intentions to leave a car for his wife constrain this decision. This is exactly the type of concern I want to look at, the role that other intentions play in determining the ways of satisfying our present intentions.

In ItPlanS, there are actually three different places that this kind of information will be needed in order to make choices about what actions to take. The first and most obvious is the action decomposition process. There is nothing in this formalism that suggests that for a given intention there is only one action which satisfies it. This leaves us with two major questions. First, if an intention needs to be broken down into subintentions and there is more than one action to accomplish this task which one should ItPlanS chose? Obviously, we want ItPlanS to consider its list of future intentions, and attempt to determine if some of these actions are incompatible with some or all of the system's possible actions. I have had some thoughts on this subject but as I have said this is largely an area for future work. Second, is the question of multiple primitive actions fulfilling an intention. Are some primitive actions better than others for some intentions? And if so how should ItPlanS decide this issue?

The second place intentions should impact the choices ItPlanS makes is in the choosing of actions when there is a negative intention conflict. As noted, as the first step in that process, ItPlanS must chose one course of action to pursue. This choice is very similar to the previous case. However, in this case, there is more information to be considered, for example, the ease with which one can invert one of the result rule's conditions, and the other results of inverting one of the condition.

Finally, the choice of which condition to invert in a result rule in the event of a negative intention conflict must involve the system's other intentions. As stated earlier, ItPlanS already considers those intentions that have been satisfied when making this choice. Of course, ItPlanS should also consider its future intentions when making this decision. It may be the case that one of the possible conditions is significantly easier to change or will involve less change in the world. Or it might be the case that inverting one of the conditions will interfere with the achievement of one of the system's later intentions. Obviously this would not be a good choice.

In the end, the intentions of the system play a role in just about all of its decisions. Some of these roles have been well specified in the existing work. However, at least in these three cases, there is more work to be done.

4.2 Limiting Model Size and Complexity

Another area for future work is in the construction of the models of the world ItPlanS uses to simulate the results of its actions. I suggested that ItPlanS makes a "minimal" model of the world in which to simulate the action. I argue the size and content of the model depend on the action being taken and the problem space.

One of the chief problems in planning has been the maintenance of large models of the world. Therefore, ItPlanS limits the models created to simulating a single action and should limit the size and complexity of these models to information believed to be relevant to the action. For example, when performing a GOTO action in the blocks world the color of the block in unimportant and should not be included in the simulation. On the other hand, the ON relation is very important when considering a GOTO action.

Therefore, in order to generate the model for a specific action, ItPlanS should start with the set of objects directly involved with the action and construct the transitive closure of the relevant predicates over this initial set. Let us return to our example, if we assume that ON is the only relevant predicate, the the initial set of objects in our model is just the block "a" since the action we are going to carry out is a GOTO and INHAND(a) is true. Taking the transitive closure of the ON relation will get us all of the relations that hold between anything "a" is on and anything on "a."

While I believe this to be a satisfactory solution to this problem, I have not yet implemented it in the ItPlanS system. Therefore, I have no empirical validation for the sufficiency for this procedure yet. Of course, this solution method implies that ItPlanS must have declarative knowledge of those relations relevant to the results of actions, however, this does not strike me as unrealistic in any way.

4.3 Reasoning in the World Model

There is a second problem with the modeling of action's effects in ItPlanS. The simple add and delete lists in result rules are not powerful enough to express all of the beliefs we would like the agent to hold as a result of an action. For example, we want to be able to express the fact that all of the blocks on top of a block that is moved will fall, not just the one stacked immediately on top of it. This type of inference calls for a limited reasoner to be placed in the action simulator. The bounds and limits for this type of reasoning will most likely involve some kind of naive physics. Here again, I

have not given the problem much thought, but it is clear that something of this type is required. The need for this kind of limited reasoning will have to be explored even if only to find a sufficient solution in order to look at the more central issues of this work.

4.4 Belief Inconsistency and Inaccuracy

Finally there is a large area to be explored in terms of the requirements for consistency and accuracy in the agents beliefs. In ItPlanS the system's beliefs about the outcomes of actions are not required to be correct; the possible results lists may be in error as may the result rules. However, the system as it stands now assumes that the agent's beliefs ARE, in fact, correct and that all actions are completed successfully.

There are a number of issues that must be looked at in this light. Some of these I have alluded to before. Since the possible results of an action are not required to be a subset of the addlists of the result rules it is possible for the agent to know that an action causes a result but not know why.

Also the issues of the monitoring to check for the completion of actions as well as issues of failure recovery. For example, when should ItPlanS decide that the action it has just carried out not been successful? How should it react to this failure?

These are just some of the obvious limitations associated with the incompleteness of the systems knowledge. While I again have not given substantial thought to these problems, they are significant and further work needs to be done to confront them.

5 Conclusion

The ItPlanS system, while it embodies many of the elements we have talked about, I feel, is nothing more than a platform for this work. Research into the role of intentions in action selection has been sadly lacking in the area of planning. The relevance and role played by intentions in action choices is the main trust of this work and while ItPlanS begins to address some of these issues there is a great deal of work left to be done.

References

[1] Agre, P.E. *The Dynamic Structure of Everyday Life*, Technical Report 1085, MIT Artificial Intelligence Laboratory, 1988.

[2] Agre, P.E. and Chapman D. Pengi: An Implementation of a Theory of Activity. *Proceedings of the Sixth National Conference on Artificial Intelligence*, Seattle, Washington, 1987.

[3] Bratman M. *Intentions, Plans, and Practical Reason*, Harvard University Press, Cambridge, Mass. 1987.

[4] Chapman D. Planning for Conjunctive Goals. *Artificial Intelligence*, 32:333, 1987.

[5] Davidson D. *Essays on Actions and Events*, Oxford University Press, New York, New York. 1980.

[6] Fikes R.E. and Nilsson N.J. Strips: a new approach to the application of theorem proving to problem solving, *Artificial Intelligence*, 2:189, 1971.

[7] Harman G. *Change In View* MIT Press, Cambridge, Mass. 1986

[8] McDermott D. Planning and Acting. *Cognitive Science*, 2:71, 1978.

[9] Sacerdoti E.D. Planning in a Hierarchy of Abstraction Spaces, *Artificial Intelligence*, 5:115, 1974.

[10] Schoppers M.J. Universal plans for reactive robots in unpredictable environments, *IJCAI '87*, 1039, 1987.

[11] Tate A. Generating project networks. *IJCAI '77*, 888, 1977.

World State:

ON(a,b), INHAND(c), AT(OVER(d)), on(g,h),...

Positive Intentions:

ON(a,b), ON(c,d), ON(e,f)

Negative Intentions:

broken(_)

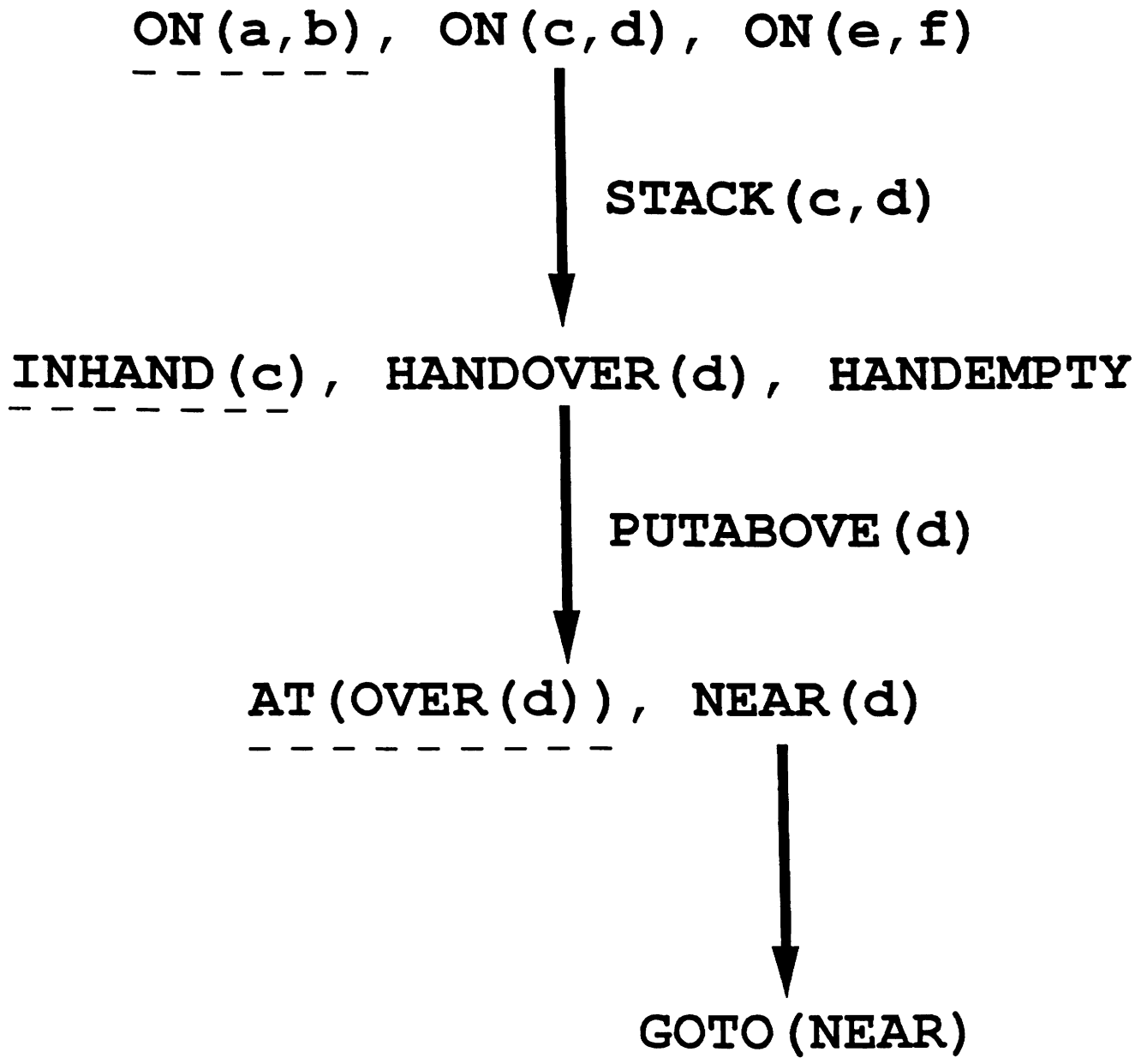


figure 1