Planning and Execution in Incompletely Specified Environments*

R. T. Chien and S. Weissman

Coordinated Science Laboratory
University of Illinois at Urbana-Champaign
Urbana, Illinois 61801
U.S.A.

Abstract

This paper describes a number of new ideas for planning and for the execution of plans in a world which is not completely specified. Such situations often occur in real world environments. Most of the ideas contained here have been implemented in a system for computer-aided decision-making for flight operations.

1. Planning and Execution in Incompletely Specified Environments

In conventional computer applications, data is input and manipulated according to predefined plans specified by the programmer. Programs are written to provide solutions to problems for which algorithms are known. While the numerical data may be changed easily, altering the goal of the program may require extensive modification. This type of approach may be unacceptable in systems where the exact problem specification is not known at the time of programming.

In order to make programs which are able to solve a wider variety of problems, much time has been spent constructing systems which attempt to make the computer "understand" the subjects with which it is dealing. Many of the systems are planners, programs which take a goal as an input and generate a plan which can be executed, at which time the given task will have been satisfied.

Most of the existing high level planners such as STRIPS [4] and PLANNER [6,14] will report a success only when a detailed plan has been developed. These planners have primarily been applied to simple domains in which all relevant aspects concerning the state of the world are known to the planner. In these domains, nothing can change without the execution of a system initiated action.

Many of the planning systems are based upon the idea that a problem may be divided into a series of subgoals (or preconditions). Any one of a number of techniques or operators can be employed In order to try to solve the subgoal. A sequence of correct operators would determine the solution. However, much of the deduction may depend upon the presence of certain data in the database. If this data were not known at the time of planning, then an entire section could fail. In many cases information may be missing because of incomplete modeling of the world due to the domains complexity. But in other cases, the overall concept may have been modeled, but the specific piece of datum may not be "known" to the planner, much as a person may not know what is going on in the next room.

This work was supported in part by the Joint Services Electronics Program *(U.S,* Army, U.S. Navy, and U.S. Air Force) under Contract DAAB-07-72-C-0259 and in part by the Air Force Systems Command under Contract AF F3315-73-C-1238.

In order to increase the complexity of problems and domains which can be accommodated, it is necessary to extend the capabilities of the deduction languages and systems in the following areas:

The deduction mechanism must have the capability to construct plans in a dynamic environment. By dynamic, it is meant that movements of objects or changes In the world can occur without being merely consequences of system actions. In this type of situation, it may be futile to formulate a detailed plan based upon specific data when a dynamic alteration of this data may totally invalidate the remainder of the plan. Planning for all of the possible alternatives would, in most cases, be unfeasible due to the large number of future states possible. One alternative would be to have a planner which would "know" that it exists in the real world. The planner would have the ability to vary the complexity of the plans generated according to the situation. This would lead to a case where there would be no necessary distinction between the planning and execution phases. In certain cases, the plans which would be generated would be of a more general nature, reflecting an outline of the important steps and tasks to be done. As the execution progresses, new information would be received and added to the database. This would allow more details of the plan to be computed as execution continued.

A deductive mechanism should be able to operate in an environment in which there is a lack of Information. This would correspond to the real world situation where a human being has to make an intelligent evaluation missing some of the facts. This capability has to be attained before the dynamic planning and execution can occur. This is because while it may be recognized that a certain aspect of the world may be expected to exhibit dynamic action, it may not be known what the value would be when needed. Planning may have to continue without the definite information. The planner must have the ability to gather new information. Among the possible ways in which this could be accomplished are: have the system develop a question or allow the mechanism to seek out information by inspecting the environment using any sensory equipment available. If it is realized that while certain possibly relevant information is not known during the periods of initial planning, it may be possible to plan if it is realized that the information is to become available at some future time. In this case, it may be necessary to analyze certain possible future states of the world. For this, there must be a mechanism for storing global knowledge and models concerning this unknown information. It may also be desirable to incorporate probabilities and costs into the deductive and decision making procedures.

2. Related Research

Research concerning the application of general deduction mechanisms to real world problems which are dynamic in nature and/or are incompletely specified have been extremely limited. It appears that many of the existing systems are incapable of being extended to handle these types of problems without extensive modifications.

PLANNER [6,14] allows strategies and relationships to express as procedures called theorems. These theorems are executed in order to try to satisfy the problem which consists of a series of goals. The control structure is based upon a depth first search with backtracking. It appears that it would be difficult to express the idea that certain facts may not be known at a given time. PLANNER understands only one type of failure, that being when a goal cannot be satisfied. If, however, a goal is not satisfied not because it is 'Wong*' but because some of the necessary data are missing, then a different type of failure has occurred, a type which PLANNER cannot understand. When dealing with this "unspecified information", it is necessary to maintain several alternatives from which one may be chosen. Storing this type of information is very difficult in PLANNER.

When evaluating a theorem, PLANNER treats each of Its steps or subgoals as equal. During the planning, all of these subgoals are of equal importance. If one fails, the theorem fails. But, it appears that in reality, subgoals have different levels of importance. This means that more time should be spent in order to satisfy a key subgoal than a relatively minor one. PLANNER'S depth first control structure would not allow the consideration of all major subgoals first.

Many of the philosophies in PLANNER are also contained in QA4 [10]. The context mechanism which is available in QA4 would allow the storage of alternative plans resulting from different possible values of unspecified information. QA4's limited effectiveness, as with PLANNER, arises from the backtracking philosophy which is embedded in both of the systems. The introduction of new types of failures makes backtracking an undesirable search technique. As in PLANNER, the inflexibility of the recommendation lists means that once a sequence of theorems is formed, it cannot be altered or edited. This appears to be inappropriate when desiring to alter control as new information is determined.

CONNIVER's [7,13] main advantages over PLANNER and QA4 are freedom from compulsory backtracking, the inclusion of a context mechanism, and flexible possibilities lists. The possibilities list, which specifies the next procedure to be tried, can be inspected or edited at any time. The control structure is based upon the frame concept [1] which allows a total deduction environment to be maintained and continued. This allows great flexibility in specifying how a theorem is to be evaluated. Despite its advantages, it appears that CONNIVER has not yet been applied in systems which require the integration of planning and execution, such as those dealing with dynamic situations in uncertain environments.

Much of the work which has been done concerning the problems found in executing and planning have been outgrowths and extensions of the STRIPS [2,3,4,5] system. STRIPS is used to generate a plan which could be solved through the application of a sequence of operators. An operator can be executed when all of its preconditions have been satisfied. The PLANNEX [6] system takes a complete STRIPS plan and monitors its execution. Using this system, actions may be deleted from the plan if it is determined that their consequences are not needed. It can also recognize if necessary Initial conditions are absent, which would lead to a replan mode. It Is also possible to take solutions which have been generated and generalize them. These MACROPS [5] are saved and can be used to satisfy future tasks or subtasks. STRIPS only succeeds when a complete plan has been generated. The presence of unspecified information would in most cases lead to a failure. STRIPS would respond to this type of failure by searching for an alternate plan.

Recent results have demonstrated that STRIPS-like systems can be made more efficient by employing a hierarchical approach [9,11,12]. These systems have been constructed using the principle that preconditions of an operator are of varying importance and some should be examined and satisfied before others. By trying to satisfy the preconditions which are most basic or are harder to achieve first, irrelevant operators can be eliminated sooner. The preconditions are assigned a criticality or rank. The higher valued preconditions represent the tasks which must be satisfied first. So, in ABSTRIPS [11], (TYPE (box room)) has a higher rank than (INROOM (box room)) and both have a higher rank that (INROOM (ROBOT room)). When a problem specification is received, the criticality is set to a maximum value (which would contain all predicates representing unchangable information). Preconditions with criticality below this value are initially ignored. A plan is constructed using whatever techniques are appropriate to the system and domain. The plans produced will satisfy all of the final conditions, but the operators specified will only be satisfied through their most critical preconditions. As the criticality is lowered, new preconditions are introduced for the already specified operators. As these precondtions are satisfied, new operators are introduced forming a more detailed plan. When the criticality has been set to its lowest value and all preconditions have been satisfied, a complete, detailed plan will have been constructed. While this type of planning has proved to be more efficient that STRIPS, of more interest are the types of plans which are generated. Some of the high-criticality plans have many of the desired attributes of a partial plan outline. The plans do not contain every necessary detail, but rather only the major steps which must occur. These approaches have not been used to satisfy problems in domains which are dynamic or incompletely specified. In [8] Minsky describes a framework for a representation of knowledge which would permit the inclusion of situation dependent default values. The scope of the world model which is considered at any time is a function of the present environment.

3. Planning in an Incompletely Specified and/or Dynamic Domain

Systems which are to operate in an incompletely specified, real world environment must of necessity operate differently than the existing planners. There must be a realization that knowledge concerning some of the relevant portions of the world may be unavailable during some of the stages of planning. This may be because the unknown portion of the world is outside of the system's monitoring capability and/or may be changing as time progresses. Because of this, the planner must realize that in many cases it is not feasible, if not futile, to insist upon the construction of a completely detailed plan before the initiation of execution. The system must have the knowledge that missing information may be obtained in various ways, such as through observation or questioning. The system may have to plan around some of the missing information by making reasonable assumptions.

When a problem is specified to a planning or execution system, it may be done in several ways. The most common method is to specify aspects of the world which must exist after the plan has been executed. This is generally accomplished by specifying a goal state. It may also be desirable to specify possible intermediate states which may have to be satisfied in a certain order. Most of the existing planners place little emphasis on how the goal state is to be achieved. There is little or no concern about whether the plan which has been generated is optimal or near-optimal according to any criteria. However, in more realistic situations, people strive for a more efficient plan even though their analysis may not include a formal statement of what is best, A planner should also be able to accept a statement of what criteria should be used*

In these types of problem specifications, there are really only three general methods which can be used to insure that a portion of the goal is satisfied. First, the goal could already be true in the world and represented in the system's world model. Second, the goal could be true in the world model but not explicitly represented in the system's model and it could be deduced that the goal is a logical consequence of available information in the model. Third, it may be that the goal is not true in either the model or the world but it is possible to perform actions which will alter the world in such a manner that the goal will be satisfied. The existing general purpose planners satisfy goals using these general techniques.

But all of the situations above are predicted on the concept that all relevant Information is directly known or could be deduced. But these are clearly not realistic assumptions. The world model which the system maintains could be deficient in many ways. Some information could be missing due to the necessary simplifications which must occur when modeling a complicated domain. However, if some aspect of the world were expected to be important for planning, it would surely be represented. And some of the information could be missing because it is just not known, no matter how relevant it may be. The latter case is of major interest because this type of unspecification occurs in realistic problems when a portion of the world is beyond monitoring capability or when dynamic situation alter previously known values.

One major problem is how to recognize this type of missing information, which will be referred to as "unknown" information. When a precondition to an operator is encountered, it Is imperative to know whether it belongs to a previously mentioned category or is unknown information. No matter how complete the model, certain information may be absent if the system is solely responsible for collecting and storing the information. But by using semantic knowledge about the world, it may be possible to determine something about how to satisfy the preconditions. The Initial attempt to satisfy a precondition involves examining the database in order to see if the precondition is satisfied because it is already true. As soon as it is determined that the needed fact is not in the database, the system checks to determine if the concept is unspecified. In many cases it is possible to determine that a precondition is not satisfied by inspecting the database for contradictory information. If this is the case, then the system can conclude that the Information is specified and that some action is needed. Sometimes, however, sufficient information is not available to make the determination and the system may have to postpone the decision. Initially, research concerning unknown information will be confined to predicates whose restrictions are limited. In these cases the alternate values and contradictions can be expressed in a fairly straightforward manner.

When the database is being referenced, it is not enough to find a specific fact represented. The dynamic properties of the domain have to be considered. Some of the attributes may change dynamically in a random or predetermined manner. This would affect the confidence in the truth or falsity of a fact.

In order to ascertain the value for an item of unknown information, it is necessary to activate some type of input. This may include any sensory device available, such as a camera for observation. It may also take the form of a response to a question. In any case, the system must know the appropriate methods available. It must also be aware of when types of information can be obtained.

As has been stated, a planned solution to a task is a sequence of actions whose execution would alter the world from an initial state to a goal state. The proper actions are determined by evaluating operators. The form of the operators is shown in Figure 1. The operators represent allowable actions in a domain. They are STRIPS-like in representation but have direct counterparts in PLANNER and CONNIVER. Each operator has a set of preconditions which must be satisfied before the action can be executed (either real-time or during planning). The operators have ADDITION and DELETION lists. These represent the aspects of the world which are expected to be altered when the action is executed. These are only used to update the system's world model. The system is responsible for making any observations necessary to insure that the changes in the world correspond to the expected changes.

Each of the preconditions, additions and deletions has a number associated with it. This is the criticality of a predicate. There is an upper limit for the criticality in a domain. This is for concepts which cannot be altered by any system action. The concepts represented by predicates with lower criticalities can be changed. The criticalities roughly represent the order in which the preconditions must be satisfied. If there are two preconditions with different criticalities, the one with the highest criticality is satisfied first; the lower precondition will be satlsfiable in some manner.

```
(TO task)
     (ACTION action)
          (PRECONDITIONS
              (criticalityp1 (condition1))
              (criticalityp2 (condition2))
                        .
                        .
              (criticalitypn (conditionn)))
          (DELETION
              (criticalityd1 (deletion1))
              (criticalityd2 (deletion2))
                        .
                        .
              (criticalitydi (deletioni)))
          (ADDITION
              (criticalitya1 (addition1))
              (criticalitya2 (addition2))
                        .
                        .
              (criticalityaj (additionj)))
```

Figure 1

So, when considering an operator, the criticality is set to some value. Any precondition with criticality below this is not considered at that time. If all of the pertinent preconditions have been satisfied when the criticality is "n", it is said that the operator is satisfied through criticality n. If all of the operators in a plan are satisfied through criticality 1, then a complete, detailed plan should have been generated.

A significant research area concerns how to satisfy the prerequisites and the development of a control structure which would facilitate the efficient construction of intelligent plans. The conventional methods for satisfying the prerequisites have been described above. Research is being conducted concerning the possibility of considering a prerequisite to be satisfied by "assumption" at certain levels of criticality and stages of planning. In these cases, a precondition may be assumed to be satisfied (or satlsfiable) at an early planning stage. The system must be aware of the assumptions being made and have reasons for these actions. To date, several classes of assumptions and their reasons have been formulated. The most basic is a low-criticality precondition. In most cases it is possible to assume that a precondition with a criticality below some cutoff can be satisfied. This is because this type of precondition was to be constructed as an easily done detail. The criticality cutoff could be determined by the stage of planning, the domain used, as well as the system's knowledge of what tasks could always be accomplished.

Another type of assumption is called a logical assumption. This type of assumption originally would occur when unknown information was involved. The various possible values would have been examined. If, for each case it was determined that the precondition could be satisfied, the precondition would be assumed to be logically satlsfiable. The information derived from searching all of the possibilities should somehow be saved so that an assumption could be made if the proper conditions are met.

In many cases a precondition can be assumed to be satlsfiable if certain relationships exist between the precondition of the operator being evaluated and the precondition of any operator used to satisfy the original precondition. This is called a dominance assumption. An operator, 0P1, is defined to dominate another operator, 0P2, if all of 0P2's preconditions are among the preconditions of 0P1 with the same relationship restricting any uninstantiated variables. Now, If a precondition of 0P1 may possibly be satisfied by the application of 0P2, then the precondition may be assumed to be satlsfiable by dominance.

In the previous two cases, the assumptions which could be made are very dependent upon the planning environment which exists when a precondition is encountered. Hopefully, this will lead to a system which has a better knowledge of what it is trying to accomplish at any given time as well as a knowledge of situation methods of dealing with the preconditions.

The last type of assumption which has thus far been considered has been called a linkage assumption. This type of assumption arises because the lack of knowledge concerning the exact order of execution of a plan satisfying part of a top level goal will cause unknown information to exist. In this case the information is known in the real world but may be altered during planning of another plan. In many cases the overall goal will be divided into subplans, each of which is developed independently. The exact order in which the plans will be executed is not known at the time of planning. When trying to satisfy certain preconditions, a planner may examine the real-world database (as opposed to a local, planning database). But information found in the real world database may be altered by other subplans by the time the subplan is actually executed. The system must have knowledge of whether database entries are expected to change. For those that are expected to change, the appropriate assumption may be that the precondition will be satisfiable during the execution phase.

The last type of assumption mentioned is basically restricted to attributes which are expected to change. There are some types of information which are not expected to change (even though they may). In these cases, the system may use values found in the latest real world model. But these should be noted in the plan being produced.

A problem concerns when the system should initiate execution. This also involves the question of how detailed should the planning be. In theory, the execution could be initiated during any stage of planning. However, a more realistic approach would have the execution begin when a "reasonable" plan has been developed. In some cases the initial course of action may be so well defined (or may be the only alternative) that the system may decide to start execution before the initial planning has terminated. Observations, which are a type of execution, could be performed at any time during planning or execution if the proper conditions occur in the real world.

After the plan outlines satisfying portions of the top level goals have been generated, it is necessary to link them into a coherent plan outline. To do this, an order of execution must be determined and intermediate connecting programs must be developed. The subplans are classed according to the criticalities of the tasks which are satisfied. An attempt is made to link the highly rated subplans to the initial world. When a "shortest" linkage is found to a subplan, this subplan is assumed to be executed next. Linkage is attempted between the remaining subplans and the world model of the. most recently assumed executed subplan. This continues until all of the subplans have been linked.

Among the major problems which still must be considered for effective linking include: how to distinguish between the case when two subplans of different criticalities have the same initial environment condition and should be executed consecutively and the case when the lower criticality subplan must be executed with subplans of its own criticality to avoid making other subplans undoable; what types of searches are necessary to promote the most "efficient" linkage for the entire plan? Surely a most efficient first linkage is not enough.

It would be very desirable to have the system be aware not only of what part of the plan it is executing but also the knowledge needs and preconditions of other subplans. In the present formulation, an attempt is being made to obtain this type of performance. When observations are needed and the necessary environment does not exist, demons will be established so that, should the opportunity arise, the observations could be made. Certain aspects of the world could be protected as preconditions for future subplans, top-level goals as well as any currently active operators. If a subplan tries to undo another subplan's initial assumptions, the planner being constructed will be warned of this.

In some cases, a plan may be executing and during an observation, some previously unknown information is acquired. This new information may lead the system to re-evaluate the manner in which the overall goal is to be satisfied. This may involve satisfying a subgoal with a different operator and/or altering the order in which subplans are executed. For this type of performance to occur, the system must be aware of some of the important possible alternatives. Investigation has been concerned with two possible cases: new information leading to alternate plans, and new information leading to short cuts.

When trying to satisfy a goal, the system may be considering several possible approaches, A situation which can occur is that after planning has been successful through a certain criticality, a failure due to unknown information is encountered. In the previous cases discussed, it was possible to "assume" that the precondition could be satisfied, but this is not always the case. If the unknown precondition is found to be satisfied, the system should be able to pause and consider the new information. If a new subplan were found which was expected to be superior, the linkages may have to be reformulated. In most cases, it is not expected that this would alter the composition of other subplans. When the subplan is being reformulated, other unknown failures may still be encountered which may dissuade the system from pursuing these paths.

The possibility of a short-cut also may occur when a failure due to unknown information is encountered during planning. In this case, the operator which was being considered when the failure occurred is later found to be potentially useful in the finally constructed plan. To determine this, the system examines the failure and subplan produced and asks the questions:
1) Is the operator which was being examined when the failure occurred still potentially useful in that the change the action would have yielded was realized in some manner later on in the plan (or more precisely, did planning continue until a subplan to realize the action goal was obtained)?
2) If the failed precondition were assumed to be true, would all of the operator's preconditions be satisfied or satisfiable to a certain criticality?

If these conditions are met, then the plan should be altered to indicate that if the precondition is found to be true, an alternate shorter subplan could replace part of the plan without affecting any of the other linkages or subplans. It is hoped that this type of planning will not only make the system more responsive to new information which is received, but will also allow the system to demonstrate a better understanding of what it is trying to do at all levels of planning and execution ncluded as unspecified parameters. It is sometimes possible to judge the superiority of one plan over another with only this information.

The preceding discussion was primarily concerned with planning when missing some relevant information. Another problem which arises is to construct a plan which is most compatible with the overall mission. To accomplish this, it is necessary that the system be aware of the overall mission, including an overall plan of how the mission is to be accomplished. The system must also be aware of both the pilot and system capabilities; who is to complete a task and how is it to be done. The system should then be able to develop a plan which would conserve the most essential on-board capabilities and would least interfere with the pilot's actions.

4. Conclusions

Systems which are to be able to plan and execute solutions to real world problems must be able to plan in incompletely specified environments. The system should have the knowledge that certain relevant information may not be known at all stages of planning. The system should be able to form a plan outline indicating the "major" steps. For various situations, it should be able to assume that certain tasks are satisflable, deferring planning until a time just before execution. The system should have the ability to initiate execution before planning has terminated. As new information is received, the plan and/or flow of execution should be modifiable.

When planning, the system should be cognizant of the overall mission outline. Using knowledge of pilot and system capabilities, the plan which is constructed should be compatible with the mission objective.

References

[1] Bobrow, D. and Wegbreit, B., "A Model for Artificial Intelligence Programming Languages," Third International Joint Conference on Artificial Intelligence, Stanford, California, August 1973.

[2] Fikes, R., "Failure Tests and Goals in Plans," Artificial Intelligence Group Technical Note 53, Stanford Research Institute, March 1971.

[3] Fikes, R., "Monitored Execution of Robot Plans Produced by STRIPS," Artificial Intelligence Technical Note 55, Stanford Research Institute, April 1971.

[4] Fikes, R. and N. Nilsson, "STRIPS: A New Approach to the Application of Theorem Proving to Problem Solving," Artificial Intelligence, Vol. 2, Nos. 3/4, 1971.

[5] Fikes, R., Hart, P. and Nilsson, N., "Learning and Executing Generalized Robot Plans," Artificial Intelligence, Vol. 3, No. 4, 1972.

[6] Hewitt, C., "Description and Theoretical Analysis (Using Schemata) of PLANNER: A Language for Proving Theorems and Manipulating Models in a Robot," Ph.D. Thesis, Department of Mathematics, Massachusetts Institute of Technology, 1972.

[7] McDermott, D. and Sussman, G., "The CONNIVER Reference Manual," Artificial Intelligence Memo No. 259, Massachusetts Institute of Technology, May, 1972.

[8] Minsky, M., "FRAME-SYSTEMS: A Framework for Representation of Knowledge," to be published, November 1973.

[9] Nilsson, N., "A Hierarchical Robot Planning and Execution System," Stanford Research Institute Report Project 1187, April 1973.

[10] Rulifson, J., Derksen, J. and Waldinger, R., "QA4: A Procedural Calculus for Intuitive Reasoning," Artificial Intelligence Technical Note 73, Stanford Research Institute, November 1972.

[11] Sacerdoti, E., "Planning in a Hierarchy of Abstraction Space," Third International Joint Conference on Artificial Intelligence, Stanford, California, August 1973.

[12] Siklossy, L. and Dreussi, J., "An Efficient Robot Planner Which Generates Its Own Procedures," Third International Joint Conference on Artificial Intelligence, Stanford, California, August 1973

[13] Sussman, C, "Why Conniving is Better than Planning," FJCC, 1972.

[14] Sussman, G., Winograd, T. and Charniak, E., "MICRO-PLANNER Reference Manual," Artificial Intelligence Memo 203A, Massachusetts Institute of Technology, December 1971.