557855 N90-20930 P22

Time and Truth in Plans[*]

Lyman R. Hazelton, Jr.
Flight Transportation Laboratory
Department of Aeronautics and Astronautics
Massachusetts Institute of Technology

1 Introduction

Planning denotes the formulation of a detailed scheme, program, or method worked out beforehand for the accomplishment of a goal. It involves the analysis of the desired goal and its division into sub-goals which are subsequently treated in the same way until a set of primitive objectives is obtained. A rational plan is prepared by a reasoner for execution by one or more actors or agents who perform actions to achieve the objectives. A reasoner is a cognitive system (human or machine) capable of some level of logical deliberation.

The formulation of complex plans is an arduous process. Computers have been employed to assist in the creation of plans almost from their inception. One of the first uses of computers, shortly after World War II, was to solve large linear optimization problems for military planners. However, the actual creation of a plan by a computer did not occur until Sussman's work [1] in 1975. The reason for this delay is that planning is a cognitive process not directly involving computation; that is, planning requires symbol manipulation. Also, the composition of a plan is highly domain specific, so it is difficult to make a general planning program [2]. Domain dependent plan generation is *rule based*, and the rules vary from domain to domain. The creation of plans by a computer had to await the development of a mature symbol manipulation language, such as *Lisp*.

2 The Single Actor Assumption

Sussman, and most of those who followed him, made a very powerful simplifying assumption regarding the domain of a planner. This assumption is that all actions in the domain are taken by a single actor, the plan executor. Furthermore, that actor is

[*]Research supported by NASA contract number NGL 22-009-640.

restricted to taking only one action at a time. This supposition constrains the plan to consist of a *serial sequence* of primitive objectives. Because the sequence is non-overlapping, there is no requirement for the planner to be aware of time in the general sense. Time in this restricted domain is only measured by the conclusion of a task; the clock time necessary to complete the task is unimportant. The assumption also guarantees that the environment is not hostile. Because there is only one actor, and it carries out the plan, there is no mode by which the plan may fail. This obviates *execution monitoring* and plan *repair.*

The central reason for the employment of the single actor assumption is the lack of a unified representation of the knowledge required by the more complex environment. The standard representations employed by classical planners are sufficient to represent domain knowledge alone. Information concerning causal or temporal relationships is not domain knowledge in the normal sense. Rather, this kind of information is an example of more abstract knowledge regarding relationships between facts known about the domain. For example, let us analyze the statement, "After snow plowing is completed, the runway will be returned to service." This statement is actually *three* statements: (1) Snow plowing of the runway will be completed, (2) the runway will be returned to service, and (3) the period of validity of statement (2) follows that of statement (1) temporally. Notice that statement (3) is a statement about statements (1) and (2), without regard to their domain meaning or content. Making such meta-statements requires a mechanism within the representation language which allows reference to *statements* that have previously been made.

The domain of classical planners also provides the reasoner with accurate information about the initial state of the domain and a precise goal in terms of the required final state.

3 The Real World

Although such restricted environments do exist in reality, they are very rare and usually contrived. In the "real world", there are multiple actors who can accomplish tasks simultaneously. Some of the actors may be random or chaotic or even hostile to the planner — therefore a plan or a part of a plan may fail. There may be clock time constraints on the accomplishment of a task, as well as relative temporal constraints *between tasks.* Additionally, "real world" plans may depend on causality, a notion completely missing in classical computer planners. A planner in the real world must not only *create* a plan. It must *schedule*[1] the events of the plan, *monitor* the execution of the plan and detect failures, and *repair* the plan and schedule in light of any failures that occur.

In "real world" environments, domain information may be incomplete or erroneous. The planner is required to discover spurious data and to be able to make reasonable assumptions about missing or incorrect knowledge. Finally, "real world" planners often must operate in a process management domain, in which there is *no* final goal, but rather a continuous requirement to control and keep a system "functioning". A planner with such a task must analyze the current and predicted future states of the system and even *create* goals.

4 Forward Chaining Inference

Computer reasoners are based on *rules.* For the purposes of this discussion, a rule is a generalized statement of deduction. Rules are composed of two parts, an antecedent consisting of one or more conditions, and a consequent consisting of one or more declarative statements regarding changes in the reasoner's knowledge. The

[1]Scheduling is the assignment of specific times to the objectives determined by a planning process.

consequent of a rule is valid if and only if all of the conditions in its antecedent are valid.

When a new statement about the reasoner's knowledge is asserted, the reasoner examines the antecedents of its rules to see if any of them are applicable. Those that are deemed appropriate are queued (together with contextual information, in a structure called a situation) for more detailed examination on the reasoner's agenda. When the reasoner completes its analysis of a rule it gets the next rule from the agenda, until the agenda is empty. The reasoner then has an up to date picture of the domain until some new observation is perceived.

5 Truth Maintenance

Knowledge about the world can be classified as *observed* fact and *inferred* fact.

Observed fact is information provided to the reasoner from some direct observation of the domain.

Inferred fact is knowledge produced by the reasoner via the application of its rules.

As its observations of the world change, a reasoner must change its inferred beliefs in order to preserve a correct representation of reality. To accomplish this, the reasoner must retain information regarding the evidence supporting each fact that it believes. In practical applications, this information is kept in a tree structure. The nodes of the tree represent known facts, while each arc represents an inference relation or *dependency* (see Figure 1).

The base nodes in the tree represent observed facts. All other facts in the tree are inferred facts. When an observation changes, all the inferred facts reachable by

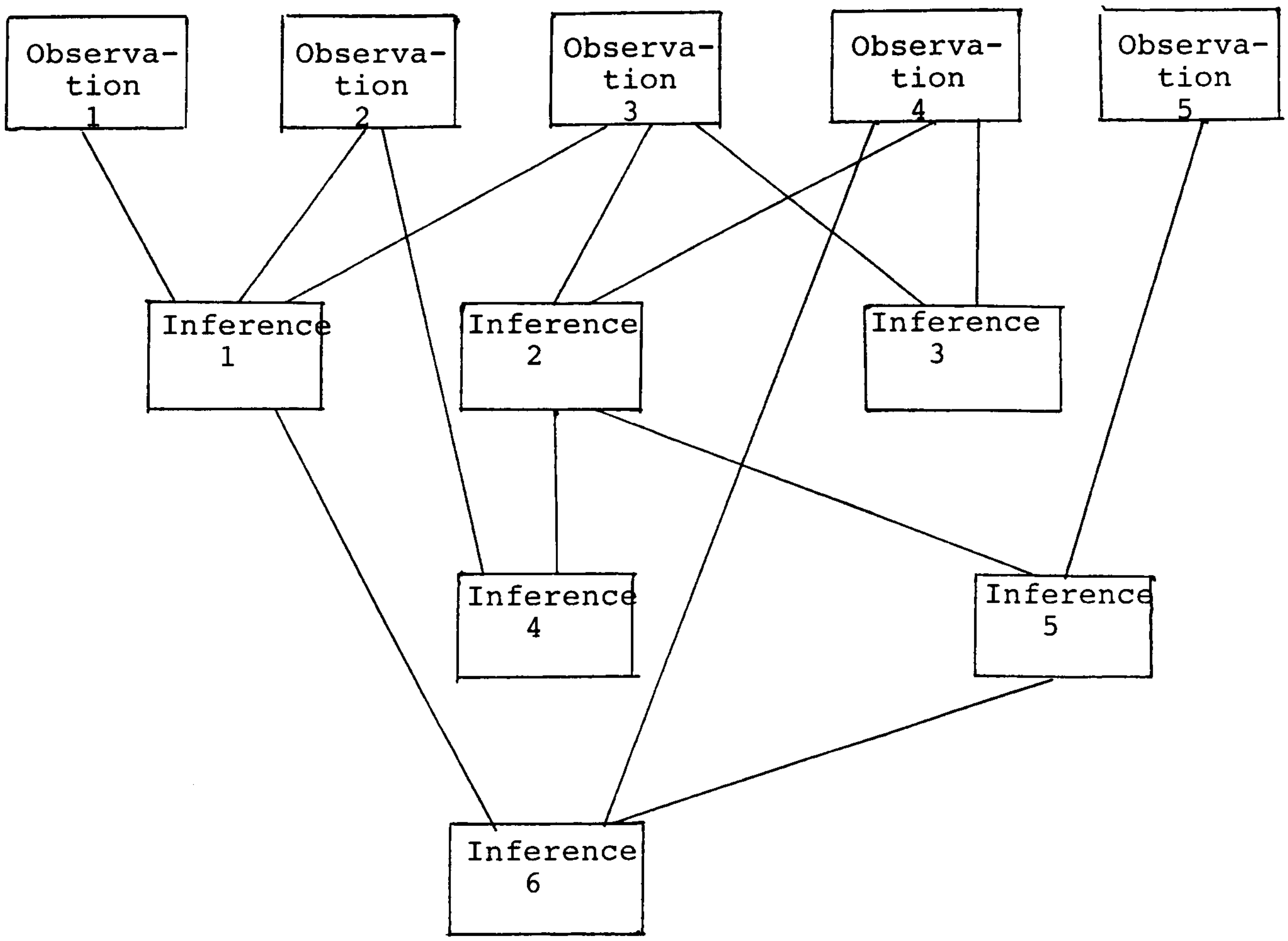

Figure 1: Evidentiary Tree

going *down* dependency arcs from the modified observation node must be investigated for correctness. Those found to be incorrect under the new observation must either be corrected or denied (i.e., deleted from the tree). All inference nodes which are dependent on a changed or denied node must likewise be inspected.

The process of maintaining a correct set of beliefs through the use of evidence is called *Truth Maintenance* ([3] and [4]). It should be noted that the process of truth maintenance is a form of *constraint propagation.* It is said that the truth of a node is constrained by the truth of its supporting nodes. The validities of all inferred facts are constrained by the validity of the facts that were used as a basis for their deduction. The inspection and update process propagates down the tree. The system of programs that implements this constraint propagation process is called a *Truth Maintenance System* or *TMS*.

6 Assumptions and Truth Maintenance

When a reasoner is faced with incomplete information, it must resort to making tentative assumptions about the questionable domain. The reasoner then proceeds to infer the consequences of those assumptions on its representation of reality. This is another classification of knowledge:

> *Assumed fact* is knowledge "guessed" in order to proceed with a chain of reasoning when the reasoner detects that it is missing some information.

A *contradiction* occurs whenever a rule deduces a fact that conflicts with another fact already known to the system. Should a contradiction be detected, the reasoner is forced to *change* one or more assumptions. Changing an assumption requires truth maintenance similar but not identical to that required when changing an observation.

Doyle [3] implicitly defines all non-deduced knowledge in his truth maintenance system as assumption. There is a disadvantage to this view. It unnecessarily increases the number of candidates when a contradiction is found. I prefer to view information that the reasoner *observes* as different than that which is *assumed* because of some lack of knowledge.

The process of changing an assumption when a contradiction is detected requires two steps. The first is to *find* an appropriate candidate to change. This step, first explored by Stallman and Sussman [5], is called "dependency directed backtracking". The second step is to remove the offender and replace it with a different assumption, if possible.

An important question needs to be addressed when reasoning with assumptions: *When* is it appropriate to make an assumption? Clearly, just because we *can* make an assumption does *not* mean that we *should* make it. Making unnecessary assumptions is expensive and can be confusing. We need to have a criterion which determines the *need* to make an assumption. I claim that the only time that we need to make an assumption is when we want to know some missing information. Thus, when we are in the middle of deducing something new about our world and we find that we are missing some fact required in the chain of reasoning, we try to make an assumption about what is unknown that is congruous to what we do know. Operationally, this means that we need a *special* set of rules for making assumptions.

These assumptive rules have the same general form as normal rules. However, they are different in three significant ways.

1. They are only invoked during the evaluation of the antecedent of a normal rule. This insures that assumptions are only made when they are required.

2. If all of the antecedents of an assumptive rule are satisfied, then the consequent asserts the assumption through the TMS, as usual. However, it is *not* necessary to search for other rules which will use the assumption and enqueue situations on the agenda. This is because if there were another rule requiring the assumption, the assumption would already have been made.

3. Assumptive rules can be *disabled* or *enabled* by the system. There are circumstances under which the reasoner can detect that a particular assumption cannot logically be made, and the associated assumptive rule(s) should not even be considered. If the circumstances change, the restriction may be lifted.

To understand why assumptive rules should be disabled, consider what must take place if an assumption results in a contradiction. First, note that contradictions can occur for two reasons:

1. Suppose that facts $\mathcal{A}$ and $\mathcal{B}$ are used by rule $\mathcal{R}_1$ to deduce fact $\mathcal{C}$. Now suppose that the resaoner observes that **not** $\mathcal{C}$ is true, and that $\mathcal{A}$ and $\mathcal{B}$ are true. In this situation, either the user is mistaken about one of the facts, or the rule $\mathcal{R}_1$ is incorrect. There is no way for the reasoner to determine which of these premises is true, so it should stop and explain the situation and ask for a correction.

2. Suppose, again, that facts $\mathcal{A}$ and $\mathcal{B}$ are used by rule $\mathcal{R}_1$ to deduce fact $\mathcal{C}$. Now suppose that the reasoner observes that $\mathcal{A}$ is true, and that the system has no knowledge about $\mathcal{B}$. The fact that $\mathcal{A}$ is true will cause the reasoner to attempt to run rule $\mathcal{R}_1$. When the antecedent requiring $\mathcal{B}$ is evaluated, the reasoner will come up empty handed and then endeavor to make an assumption about $\mathcal{B}$. If there is an appropriate assumptive rule $\mathcal{A}\mathcal{R}_1$ which succeeds, assumption $\mathcal{B}'$ is

asserted. The $\mathcal{C}'$ resulting from $\mathcal{R}_1$ will *also* be an assumption. Now, if **not** $\mathcal{C}'$ is true, a contradiction will occur.

Contradictions involving assumptions need to be looked at in greater detail. In the example above, clearly the reasoner should deny both $\mathcal{C}'$ and $\mathcal{B}'$. In addition, it should mark the assumptive rule $\mathcal{AR}_1$ as unusable (i.e. disabled) as long as **not** $\mathcal{C}'$ is known to be true. It can then consider other assumptive rules that may result in an alternative hypothesis for $\mathcal{B}$. If there are no more assumptive rules for $\mathcal{B}$, then processing of $\mathcal{R}_1$ must be terminated.

7 Time and Truth Maintenance

The truth maintenance systems of Doyle and DeKleer do not concern themselves with an environment that changes with time. In these systems, either a statement is supported by current data, or it is not.

When new data are obtained, truth maintenance systems change the implications that were previously derived based on old knowledge. Consider the following rules for a resource allocator:

1. You may only allocate a resource if it is not already allocated (in use).
2. If you do not know that a resource has been allocated, you may assume that it is available (not allocated).

In a system with truth maintenance, if the allocator wants to assign a resource, it looks for one which is uncommitted. Suppose that it only finds resources that are already committed or about which it is ignorant. It then *assumes* that one of the uncommitted resources is available. Based on the evidence of that supposition, the resource can be allocated. When the resource is allocated, the assumption that it is available is no longer true, and the TMS removes it. But the assumption of availability was the evidence supporting the action of allocating the resource in the first place, so it must remove the fact that it has allocated the resource, too. This is a variant of the famous statement, "This statement is not true." The system wants to state that the resource is available and unavailable at the same time. While the TMS may be able to detect circularities of this kind, it cannot properly resolve them without modification.

The reason that the TMS gets into trouble in the resource allocation problem is that it has no temporal knowledge. All of its statements must be true at all times. To correct this difficulty, it must be able to reason not only *why* a statement is true, but *when* it is true, as well.

Interest in temporal reasoning has increased recently. Notable work has been done by James Allen ([6] and [7]), Tom Dean ([8] and [9]), and Yoav Shoham ([10] and [11]). Prior to their work, the most that one could do was to "time tag" each fact in the system with the clock time at which it was asserted.

The real break through in Allen's work is the idea that most temporal references about facts are *relative* to temporal references of other facts. Allen further postulates that temporal references are not points in time, but intervals. That is, there is a period of time, which I like to call the "validity interval" of the fact, during which the fact is true.

Very few, if any, facts are actually tagged with a clock time in Allen's system. Instead, most validity intervals are determined by constraints derived from relationships to other facts. For example, two facts may be connected by an "after" relation: "The runway may be returned to service *after* the plowing is completed." The significance of this connection is that if the period of validity of the plowing operation is extended (i.e., the plowing takes longer than was expected), the runway must be returned to service later. The period of validity of the plowing operation constrains the validity period of the return to service. The result is another constraint propagation network, but a rather complex one. While it has been shown [12] that the maintenance of the full temporal network described by Allen is at least NP-hard, the approach has interesting implications to truth maintenance systems.

In the truth maintenance constraint propagation process described above, the arcs in the network identify a single relationship: implication. According to Allen, there are thirteen different ways that one temporal interval can be related to another. Further, there is a transitive algebra that allows one to compute the relationship between two intervals that have known relationships to a common interval. This means that it is not necessary to keep *all* of the arcs that could connect the nodes in the net, but only a sufficient number of them to allow the others to be computed as necessary. The computation of a transitive relation is claimed to be fast, so it seems reasonable to keep only a minimum network.

Every fact known to the system must have an associated temporal validity interval. Facts which are true at all times have a special temporal interval "ALWAYS". While we could also support facts which are never true with another special interval, it is more efficient to simply exclude them all together. Note that a statement of the form "X is never true" is a true statement over the interval ALWAYS.

In a planning domain, it is not enough to simply change the rules to incorporate temporal intervals and use Allen's transitive operators. To see this, consider the resource allocator previously discussed. In a temporal environment, the rules must be modified to include validity interval information:

1. You may only allocate a resource during a time interval $\mathcal{T}$ if it is not already allocated at any time during $\mathcal{T}$.

2. If you do not know that a resource has been allocated during a time interval $\mathcal{T}$, you may assume that it is available for allocation during $\mathcal{T}$.

All that this addition does is to restrict the interval of time from all time to the interval $\mathcal{T}$. While it is a necessary addition to the rules for operation in a temporal environment, the reader will see by stepping through the previous exercise that it is not sufficient to solve the problem.

The solution to the problem can be realized by observing the logic employed by a human scheduler. The human is aware of *two* times while constructing the schedule. One of the times is the period during which the resource is to be allocated, called $\mathcal{T}$ above. The other time is the actual time that the plan is being constructed, the planner's *now*. The planner is not only aware of the time sequence of the proposed schedule, but of the sequence of events during plan construction. In a temporal environment, contradiction can only happen between facts which have overlapping temporal validity intervals. In the same way that new information can change the truth of implied facts in the TMS, new information can change the temporal validity intervals of facts that were already known in the temporal system. This process goes beyond simply computing the relationship between two facts in the network.

As an example, let us re-examine the resource allocation problem once again. Suppose the allocator wants to assign a resource during a time interval $\mathcal{T}$. It begins by looking for an uncommitted resource. Suppose, as previously, it finds only resources which are committed during $\mathcal{T}$ or about which it is ignorant concerning allocation status during $\mathcal{T}$. It can then *assume*, at time τ_1, that one of these unknown resources is available during $\mathcal{T}$. Note that τ_1 is the time that the availability assumption is made, and has *nothing* to do with the proposed schedule time $\mathcal{T}$. The temporal validity interval of the assumption is from τ_1 to forever. On the basis of the assumption, the resource can be allocated during $\mathcal{T}$ at time τ_2, which is later than τ_1. This allocation produces a fact that the resource is not available for the period $\mathcal{T}$, with a temporal validity interval from τ_2 to forever. The effect of this new fact is not, however, to completely remove the assumption that the resource was available from τ_1 to forever. Rather, it removes the validity of that assumption from τ_2 to forever. This leaves the assumption that the resource is available valid from τ_1 to τ_2. Thus the evidence for the allocation step remains valid.

This mechanism is still a form of constraint propagation, since the new information about the plan constrains the temporal validity interval of old plan information. While it is possible that this change may effect the validity intervals of yet other facts, in practice it does so only rarely.

8 Conclusions

In this report, I have described some of the logical components of a rule based planning and scheduling system. I have pointed out a deficiency in the conventional truth maintenance approach to this class of problems and suggested a new mechanism which overcomes the problem.

This extension of the idea of justification truth maintenance may seem at first to be a small philosophical step. However, it embodies a process of basic human reasoning which is so common and automatic as to escape conscious detection without careful introspection. It is vital to any successful implementation of a rule based planning reasoner.

References

[1] Sussman, G. J., *A Computer Model of Skill Acquisition*, Elsevier, 1975.

[2] Chapman, D., *Planning for Conjunctive Goals*, M.I.T. Artificial Intelligence Laboratory Technical Report 802, 1985.

[3] Doyle, J., *A Truth Maintenance System*, Artificial Intelligence, Volume 12, 231-272, 1979.

[4] deKleer, J., *An Assumption-Based Truth Maintenance System*, Artificial Intelligence, Volume 28, 127-162, 1986.

[5] Stallman, R. M., and Sussman, G. J., *Forward Reasoning and Dependency Directed Backtracking in a System for Computer Aided Circuit Analysis*, Artificial Intelligence, Volume 9, 1977.

[6] Allen, J. F., *Maintaining Knowledge About Temporal Intervals*, Communications of the ACM, **26**(11), 832-843, 1983.

[7] Allen, J. F., and Kooman, J. A., *Planning Using a Temporal World Model*, Proceedings of the International Joint Conference on Artificial Intelligence, IJCAI-1983, 1983.

[8] Dean, T., *Intractability and Time-Dependent Planning*, Proceedings of the 1986 Workshop on Reasoning about Actions and Plans, 245-265, Morgan Kaufmann, 1987.

[9] Dean, T., and Kanazawa, K., *Probabilistic Causal Reasoning*, The Fourth Workshop on Uncertainty in Artificial Intelligence, 73-80, 1988.

[10] Shoham, Y., *Reasoning About Change*, M.I.T. Press, Boston, MA, 1987.

[11] Shoham, Y., *Time for Action*, Proceedings of the International Joint Conference on Artificial Intelligence, 954-959 & 1173, 1989.

[12] Kautz, H., and Vilain, M., *Constraint Propagation Algorithms for Temporal Reasoning*, Proceedings of the Fifth National Conference on Artificial Intelligence, AAAI-1986, 1986.