AI Planning: Systems and Techniques[1]

James Hendler, Austin Tate, and Mark Drummond

This article reviews research in the development of plan generation systems. Our goal is to familiarize the reader with some of the important problems that have arisen in the design of planning systems and to discuss some of the many solutions that have been developed in the over 30 years of research in this area. In this article, we broadly cover the major ideas in the field of AI planning and show the direction in which some current research is going. We define some of the terms commonly used in the planning literature, describe some of the basic issues coming from the design of planning systems, and survey results in the area. Because such tasks are virtually never ending, and thus, any finite document must be incomplete, we provide references to connect each idea to the appropriate literature and allow readers access to the work most relevant to their own research or applications.

A long-standing problem in the field of automated reasoning is designing systems that can describe a set of actions (or a plan) that can be expected to allow the system to reach a desired goal. Ideally, this set of actions is then passed to a robot, a manufacturing system, or some other form of effector, which can follow the plan and produce the desired result. The design of such planners has been with AI since its earliest days, and a large number of techniques have been introduced in progressively more ambitious systems over a long period. In addition, planning research has introduced many problems to the field of AI. Some examples are the representation and the reasoning about time, causality, and intentions; physical or other constraints on suitable solutions; uncertainty in the execution of plans; sensation and perception of the real world and the holding of beliefs about it; and multiple agents who might cooperate or interfere.

Planning problems, like most AI topics, have been attacked in two major ways: approaches that try to understand and solve the general problem without the use of domain-specific knowledge and approaches that directly use domain heuristics. In planning, these approaches are often referred to as domain dependent (those that use domain-specific heuristics to control the planner's operation) and domain independent (those in which the planning knowledge representation and algorithms are expected to work for a reasonably large variety of application domains). The issues involved in the design of domain-dependent planners are those generally found in applied approaches to AI: the need to justify solutions, the difficulty of knowledge acquisition, and the fact that the design principles might not map well from one application domain to another.

Work in domain-independent planning has formed the bulk of AI research in planning. The long history of these efforts (figure 1) has led to the discovery of many recurring problems as well as to certain standard solutions. In addition, there have been a number of

A long-standing problem . . . is designing systems that can describe a set of actions . . . that can be expected to allow the system to reach a desired goal.

0738-4602/89/$4.00 ©1990 AAAI

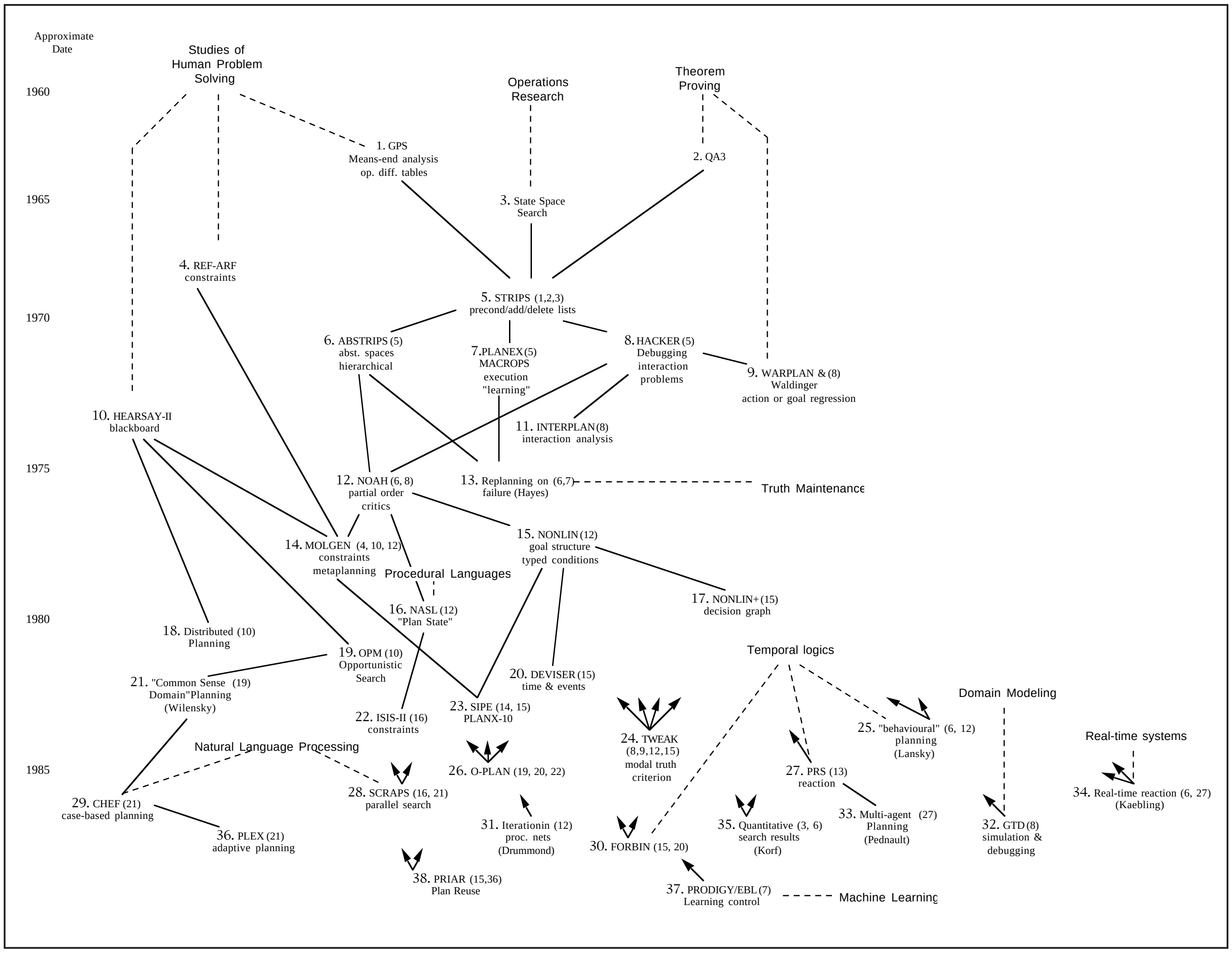
Approximate Date
1960
1965
1970
1975
1980
1985
Studies of Human Problem Solving
Operations Research
Theorem Proving
1. GPS Means-end analysis op. diff. tables
2. QA3
3. State Space Search
4. REF-ARF constraints
5. STRIPS (1,2,3) precond/add/delete lists
6. ABSTRIPS (5) abst. spaces hierarchical
7. PLANEX (5) MACROPS execution "learning"
8. HACKER (5) Debugging interaction problems
9. WARPLAN & (8) Waldinger action or goal regression
10. HEARSAY-II blackboard
11. INTERPLAN (8) interaction analysis
12. NOAH (6, 8) partial order critics
13. Replanning on (6,7) failure (Hayes)
Truth Maintenance
14. MOLGEN (4, 10, 12) constraints metaplanning
Procedural Languages
15. NONLIN (12) goal structure typed conditions
16. NASL (12) "Plan State"
17. NONLIN+ (15) decision graph
18. Distributed (10) Planning
19. OPM (10) Opportunistic Search
20. DEVISER (15) time & events
21. "Common Sense (19) Domain"Planning (Wilensky)
22. ISIS-II (16) constraints
23. SIPE (14, 15) PLANX-10
Temporal logics
24. TWEAK (8,9,12,15) modal truth criterion
25. "behavioural" (6, 12) planning (Lansky)
Domain Modeling
Real-time systems
Natural Language Processing
26. O-PLAN (19, 20, 22)
27. PRS (13) reaction
28. SCRAPS (16, 21) parallel search
29. CHEF (21) case-based planning
30. FORBIN (15, 20)
31. Iterationin (12) proc. nets (Drummond)
32. GTD (8) simulation & debugging
33. Multi-agent (27) Planning (Pednault)
34. Real-time reaction (6, 27) (Kaebling)
35. Quantitative (3, 6) search results (Korf)
36. PLEX (21) adaptive planning
37. PRODIGY/EBL (7) Learning control
Machine Learning
38. PRIAR (15,36) Plan Reuse

Figure 1. A Brief Chronology of Some Well-Known Planning Systems.

The numbers in parentheses represent systems on which each planner has directly built (also shown as solid lines where possible). The dotted lines represent some of the important outside areas influencing the development of planning systems.

attempts to combine the planning techniques available at a given time into prototypes able to cope with increasingly more realistic application domains. (Table 1 lists some of these efforts and the domains to which they were applied.)

The goal of this article is to familiarize the reader with some of the important problems that have arisen in the design of planning systems and some of the many solutions that have been developed in the over 30 years of planning research. We broadly cover the major ideas in the field of AI planning and attempt to show the direction in which current research is going. We define some of the terms commonly used in the planning literature, describe some of the basic issues coming from the design of planning systems, and survey results in the area. (Because of the recurrence of many themes throughout the planning literature, we survey the field on the basis of areas of interest rather than in terms of its chronological development.) Such tasks are virtually never ending, and thus, any finite document must be incomplete. Thus, in addition to our discussion of the issues, we provide references to connect each idea to the appropriate literature and allow readers access to the work most relevant to their own research or applications.

Planning Terminology

An AI planning system is charged with generating a plan that is one possible solution to a specified problem. The plan generated will be composed of operator schemata, provided to the system for each domain of application. This section briefly considers the meaning of each of these terms and how they relate to one another.

A *problem* is characterized by an initial state and a goal state description. The *initial state description* tells the planning system the way the world is right now. The *goal state description* tells the planning system the way we want the world to look when the plan has been executed. The world in which planning takes place is often called the *application domain*. We sometimes refer to the goal state description as simply the goal. In many systems, a goal can be transformed into a set of other, usually simpler, goals called *subgoals*.

Operator schemata characterize actions. (The terms action and event are often used interchangeably in the AI planning literature and are here.) Schemata primarily describe actions in terms of their preconditions and effects. Plans are built from these operator schemata. Each operator schemata characterizes a class of possible actions by containing a set of variables that can be replaced by constants to derive operator instances that describe specific, individual actions. When the distinction doesn't matter, we use the term operator to stand for both operator schemata and operator instances. An action that the planner considers to be directly executable is referred to as a *primitive action* or, simply, a primitive.

Planner	Domain
STRIPS (Fikes & Nilsson 1971	Simple Robot Control
HACKER (Sussman 1973)	Simple Program Generation
NOAH (Sacerdoti 1977)	Mechanical Engineers Apprentice Supervision
NONLIN (Tate 1977)	Electricity Turbine Overhaul
NASL (McDermott 1978)	Electronic Circuit Design
OPM (Hayes-Roth & Hayes-Roth 1979)	Journey Planning
ISIS-II (Fox et. al. 1981)	Job shop Scheduling (Turbine Production)
MOLGEN (Stefik 1981a)	Experiment Planning in Molecular Genetics
SIPE (Wilkins 1983)	Aircraft Carrier Mission Planning
NONLIN+ (Tate & Whiter 1984)	Naval Logistics
DEVISER (Vere 1983)	Voyager Spacecraft Mission Sequencing
FORBIN (Miller et. al. 1985)	Factory Control

Table 1. Numerous Planning Systems Have Attempted to Integrate Available Planning Techniques and Apply Them to Application Domains.

Pickup(x)

Precondition: ONTABLE(x) ^ HANDEMPTY ^ CLEAR(x)

Delete List: ONTABLE(x) HANDEMPTY CLEAR(x)

Add List: HOLDING(x)

Figure 2. A Typical Strips Operator.

The operator Pickup contains a precondition formula and add- and delete-lists. Thus, Pickup can be used to lift an object that must be on the table, the hand must be empty, and the object must be clear (from the preconditions). After pickup, the object is not on the table and is not clear, and the hand is no longer empty (from the delete-list); also, the hand is known to be holding the object (from the add-list).

The terminology of Strips operators is commonly used throughout the AI planning literature (Fikes, Hart, and Nilsson 1972a, 1972b). These operators, first used in the early planning program Strips, describe an action with three elements: a precondition formula, an add-list, and a delete-list (figure 2). An operator's *precondition formula* (simply, the operator's preconditions) gives facts that must hold before the operator can be applied. The add-list and delete-list are used in concert to simulate action occurrence. If an operator's preconditions hold in a state, then the operator can be applied. Applying an operator means acting on its add-list and delete-list to produce a new state. The new state is produced by first deleting all formulas given in the delete-list and then adding all formulas in the add-list. Although newer planning systems do depart from this approach, the terminology of Strips operators is fairly standard, and we use it often in this article.

A *plan* is an organized collection of operators. A plan is said to be a solution to a given problem if the plan is applicable in the problem's initial state, and if after plan execution, the goal is true. What does it mean for a plan to be applicable? Assume that there is some operator in the plan that must be executed first. The plan is applicable if all the preconditions for the execution of this first operator hold in the initial state. Repeated analysis can determine whether all operators can be applied in the order specified by the plan. This analysis is referred to as *temporal projection*. The first state considered in the projection is the problem's *initial state*. Repeated

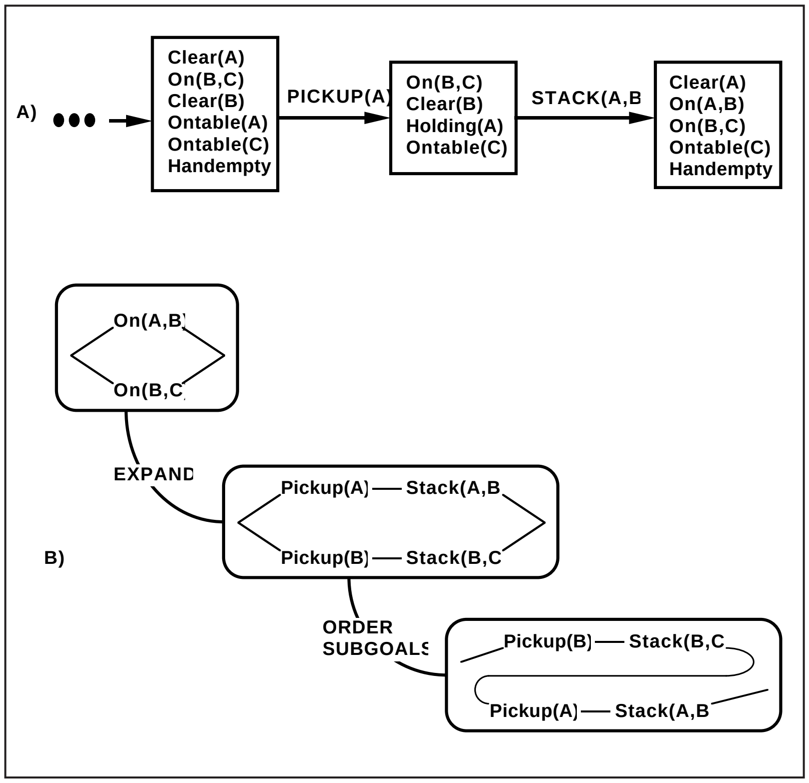

Figure 3. Partial Solutions to the Blocks World Problem (ON A B) & (ON B C).

Figure 3a (based on Strips) shows a typical state space and a sequence of operators representing a partial solution. Figure 3b (based on Noah) shows the plan as a partial ordering of actions representing a partial solution.

operator applications produce intermediate state descriptions. If the goal is true at the end of this projection, the plan is a solution to the specified problem.

Thus, the input to a typical AI planning system are a set of operator schemata and a problem that is characterized by an initial state description and goal. The output from the planner is a plan that under projection satisfies the goal. The process connecting the input and output is known by various names. Common names are plan generation, plan synthesis, and plan construction. A planner is called domain independent in the sense that the plan representation language and plan generation algorithms are expected to work for a reasonably large variety of application domains. Although this view of planning is slightly restrictive, it suffices for the purposes of this overview.

A planner is organized so that it defines a search space and then seeks a point in this search space that is defined as a solution. Planners differ about how they define their search space. Some (most of the pre-1975 planners) define points in the search space as *states* of the application's world at various times. This world state can be traversed by applying any applicable operator to some chosen state, leading to a different point in the search space. A *problem solution* can be defined as a sequence of operators that can traverse the search space from some initial state to some state defined as satisfying the goal. Thus, a state-space plan contains descriptions of states of the world the plan is to be executed in, and move operators that correspond to the actions to be carried out to execute the plan. A state-space solution is trivial to recognize: It is quite simply a path from some initial state to a state satisfying the goal specification (figure 3a).

Other (most of the post-1975) planners define points in the search space as partially elaborated plans. One point in this space is changed into another using any applicable planning transformation such as the expansion of an action to a greater level of detail, the inclusion of an additional ordering constraint between actions to resolve some interaction between effects of unordered actions, and so on. Given some initial (skeleton) plan defining a point in this partial plan search space, a sequence of plan transformations must be applied that lead to a fully detailed plan that satisfies the goal (figure 3b). A plan is considered complete when all the goals can be realized as a set of primitive actions (totally or partially ordered depending on the planner).

Systems that search through the space of partial plans have typically represented plans as an action ordering in which the actions, described by operators, are strung together with temporal ordering relations. The most famous example of such plans is Sacerdoti's (1975) procedural nets (see Planning and Search). *Action-ordering plans* directly describe the relationships among actions rather than through states and predicates contained within states.

In an action-ordering representation, a plan need not completely specify the conditions that each action affects. The plan can simply order two actions without specifying intermediate states. In contrast, *state-based plan structures* typically require complete specification of these intermediate states. This format is particularly difficult for use in describing complicated causal and temporal relationships between actions. Thus, many complex domains are quite difficult to encode using a state-based approach (Lansky 1987), and action-ordering approaches have become the generally used technique.

Planning and Search

As seen from the previous discussion, planning is essentially a search problem. The program must traverse a potentially large search space and find a plan that is applicable in the initial state and produces a solution to the goal when run. This search can be quite difficult because the search space can contain many interactions between different states or partial plans. These interactions lead to a surprising amount of complexity; for example, establishing the existence of a precondition in a partially ordered plan can require exponential computation (Chapman 1987), and the problem of finding an optimal plan in even a simple blocks world domain has recently been shown to be NP-hard (Gupta and Nau 1990).

Planning involving conditional effects has been shown to be undecidable (Chapman 1987); that is, a plan generated in such domains cannot be guaranteed to succeed without some type of execution-time additions. This issue is discussed in Planning and Execution. Therefore, the problems of organizing the heuristic search, choosing what to do in cases of failure, and generally finding ways of making more informed choices have been among the most discussed in the planning literature. (A good quantitative discussion of planning viewed as heuristic search can be found in Korf [1987].)

Early approaches to planning sought to apply legal moves to some initial states to search for a state that satisfied the given goals. There was a great deal of overlap with game-playing work. Heuristic evaluation functions were employed to rate the various intermediate states produced to estimate their closeness to a goal state (for example, A* [Hart, Nilsson, and Raphael 1968] and the graph traverser [Doran and Michie 1966]). This approach, however, was found wanting because of the difficulty in designing working heuristics and the usually exponential growth of the search space.

To reduce the number of intermediate states considered, Newell and Simon (1963) introduced *means-ends analysis*, a heuristic that involves considering only those activities that could satisfy some outstanding goal. Operators were preferred that would cause the current state to more closely resemble the goal state when run. This technique was used as the basis of the search in many of the early planners that used state-space-based plans.

With the introduction of procedural nets in Noah (Sacerdoti 1975), the search problem changed. In this system and its descendants, the search space consists not of a set of world states but of a space of partial plans. For any nonprimitive action in the current network, the planner can consider any known method of reducing this action to a set of other actions or primitives. Planning in such a system consists of choosing appropriate reductions from among the sets of possibilities and ordering actions to eliminate harmful interactions. One important technique introduced for searching partial plans is least commitment plan representations. (Some people use the term least commitment to refer only to the ordering of plan steps in a partial-order planner. We use the term in the broader sense, referring to any aspect of a planner that only commits to a particular choice when forced by some constraints.)

Such representations are used to allow a set of plans to be represented in a single state of the search. Examples of such representations include the use of a partially ordered plan to represent a number of possible action orderings prior to a commitment becoming necessary, as in Noah (Sacerdoti 1975), or the posting of constraints on objects referred to in the plan rather than the making of an arbitrary selection, as in Molgen (Stefik 1981b).

To search the space of partially ordered plans, many solutions have been offered. Some systems do not search through the possible alternatives at all. Instead, selections are made on the basis of locally available information, and a commitment is made to a particular solution path. (Of course, this approach means that some problems cannot be solved.) This technique has been most successful where strong domain heuristics can be used for making choices. Where such heuristics are not available, a more general solution is to use backtracking to allow backing up to occur when the goal cannot be reached based on some earlier choice. The planning system simply saves the state of the solution at each point at which there are alternative ways to proceed and keeps a record of the alternative choices. The first is chosen, and the search continues. If there is any failure, the saved state at the last choice point is restored, and the next alternative is taken (if there are no alternatives, backtracking continues over previous decisions). Simple stack-based implementation techniques can be used for this process (as done in Prolog).

The Nonlin program (Tate 1977) introduced a variant of depth-first backtracking for use in planning. Because good local information is often available to indicate the preferred solution path, it is often appropriate to try the best choice indicated by local heuristic infor-

. . . planning is essentially a search problem.

This technique . . . metaplanning . . . requires the planner to reason not only about the goal but also about the various techniques available for generating the plan.

mation before considering the many alternatives that might be available should the local choice prove faulty. Taken to the extreme, depth-first search gives something of the flavor of such a search strategy. However, gradual wandering from a valid solution path could entail backtracking through many levels when a failure is detected. An alternative is to focus on the choice currently being made and try to select the local choice that seems most promising. This process continues while all is going well (perhaps with some cutoff points to take a long, hard look at how well the process is going). However, if a failure occurs, Nonlin considers the entire set of alternatives that were generated (and ranked by a heuristic evaluator).

This basic technique was further refined to provide the most widely used approach to controlling search in planning. Any backtracking system based on saved states and resumption points (whether depth first or heuristically controlled) can waste much valuable search effort; a solution can have several unrelated parts. If backtracking on one part has to go back beyond points at which work was done on an unrelated part, then all effort on the unrelated part will be lost. Many planning systems avoid this problem by using a variant of the backtracking methods used in Nonlin. These systems do not keep saved states of the solution at choice points. Instead, they record the dependencies between decisions, the assumptions on which they are based, and the alternatives from which a selection can be made. They then use methods for undoing a failure by propagating and undoing all the dependent parts of the solution. This process leaves unrelated parts intact irrespective of whether they were worked on after some undone part of the solution. Examples of work using this technique include Hayes (1975); Stallman and Sussman (1977); Daniel (1983); and, to some extent, Stefik (1981a, 1981b).

One alternative that was suggested to the use of backtracking-based search is the *opportunistic planning system*. These systems do not take a fixed (goal-driven or data-driven) approach to solving a problem. Instead, a current focus for the search is identified on the basis of the most constrained operation that can be performed. This can be suggested by a comparison of the current goals with the initial world model state, a consideration of the number of likely outcomes of making a selection, the degree to which goals are instantiated, and so on. Any problem-solving component can summarize its requirements for the solution as constraints on possible solutions or restrictions of the values of variables representing objects being manipulated. They can then suspend their operation until information on which a more definite choice can be made becomes available (as in Molgen [Stefik 1981b]).

A number of planning systems have an operator-like representation of the different types of plan transformations available to the planner. A separate search is made to decide which of these transformations is best applied at any point; this search happens before decisions are taken about the details of the particular application plan being produced, as in Molgen (Stefik 1981a; Wilensky 1981b), OPM (Hayes-Roth and Hayes-Roth 1979), and PRS (Georgeff and Lansky 1987). This technique, primarily used in conjunction with opportunistic planning, is often referred to as *metaplanning* because it requires the planner to reason not only about the goal but also about the various techniques available for generating the plan.

Besides the basic method of reducing the search space by selection of relevant operators, many other methods have also been used in planners. Some examples include giving goals levels of priority and considering the highest-priority goals first, as in Abstrips (Sacerdoti 1973) and Lawaly (Siklossy and Dreussi 1975), or rejecting states or plans that are known to be impossible or in violation of some rule, as in Warplan (Warren 1974). The latter was refined to include the use of domain constraints (Allen and Koomen 1983) and temporal coherence (Drummond and Currie 1988, 1989) to provide heuristics for rejecting possible actions. A variant on this approach, using a domain model to simulate the results of planning operators, was used in Simmons's and Davis's (1987) generate-test-debug planner. Deviser (Vere 1983), Sipe (Wilkins 1983), Nonlin+ (Tate and Whiter 1984), and O-Plan (Currie and Tate 1985) use checks on resource usage levels, time constraints on actions, and other resource bounds to eliminate some possibilities. A recent approach has involved the use of parallelism to briefly examine many potential choices concurrently. Heuristics are used that allow the effects of various interactions detected by the parallel search to suggest possible choices or rule out potential plans, as in Scraps (Hendler 1987). Another approach is the use of a naturally occurring locality, which can occur in a domain, to reduce the search burden by partitioning the planning search space into smaller, localized search spaces, as in Gemplan (Lansky 1988).

Conjunctive Goal Planning

In addition to the problem of controlling the search, the order in which several simultaneous goals are tackled can have a marked effect on the efficiency of the search process. Some early planners, for example, could repeatedly loop on the same goals or get redundant solution (in the sense that the final plan contained steps that could be removed without negating the plan's reaching of the goal) when the goals were attempted in the wrong order. The solving of such *conjunctive goal plans* has been the basis of much of the modern planning research. Two somewhat orthogonal approaches have been taken toward dealing with this problem: ordering the various goals by levels of importance and analyzing and avoiding the interactions caused by interactions between conjunctive goals.

The use of levels to partially overcome this problem (it is not a complete solution) was introduced in Abstrips (Sacerdoti 1973) and Lawaly (Siklossy and Dreussi 1975). These systems separated the goals into levels of importance or priority, with the abstract and general goals being worked on first and the concrete or detailed levels being filled in later. A solution was formed at the most abstract level, and the lower, more detailed levels were then planned using the preset skeleton plan formed at the upper levels. No backtracking to the higher levels was possible. Later systems (for example, Nonlin [Tate 1977]) treated the abstraction levels as a guide to a skeleton solution but were able to replan or consider alternatives at any level if a solution could not be found or if it was indicated that a higher-level choice was faulty.

Other hierarchical systems use the abstraction levels as one guide to the ordering of goals but have other mechanisms that can also be considered. Some systems are able to determine when a particular choice (at whatever level) is sufficiently constrained to be a preferable goal to work on at any time (for example, Molgen [Stefik 1981a, 1981b]). A relatively recent approach included attempting to build models that can concurrently plan at different levels of the hierarchy. A

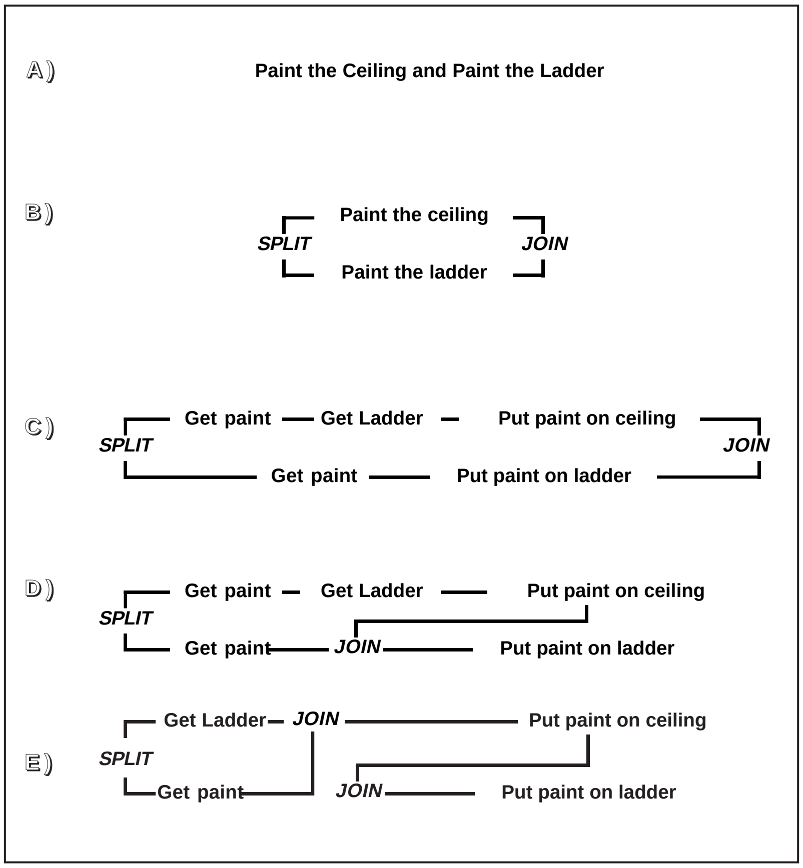

Figure 4. Painting the Ceiling (based on Sacerdoti 1977).

The conjunctive goal (figure 4a) is split into two separate tasks (figure 4b). Plans for each task are separately generated (figure 4c), and ordering links are introduced to avoid interactions (figure 4d). Finally, common subgoals are merged (figure 4e).

mathematical analysis of some properties that allow systems to assume independence of level effects in some cases was performed (Yang 1989).

A second aspect of handling conjunctive goal planning is the treatment of the interactions that arise between the different goals in the planning problem. As an example, consider the situation of trying to paint the ceiling and also painting the ladder used for painting the ceiling (Sacerdoti 1977). If the planner paints the ladder first, the ladder will be wet, and the execution agent will be unable to paint the ceiling. Further, getting the paint for the ladder and getting the paint for the ceiling should be combined in a single task. Thus, to get an optimal solution, the planner must partially merge the two separate plans (figure 4).

Planners can be categorized according to the way in which they manage interactions between goals and the way in which they structure operators in a plan. In terms of goal interactions, there are planners that make the

```
PICKUP-NEAREST(LOC,X)

 FILTER CONDITION:
        For-all (X,Y) DIST(LOC,X) < DIST(LOC,Y)
            &  OBTAINABLE(X)
 PRECONDITION:   ONTABLE(X) ^
                 HANDEMPTY ^
                 CLEAR(X)
 MONITOR:  (NEW-OBJ-MON
      Queue-length 1
        Run-time  75
      Run-every  500
     Reports:
      if DIST(LOC,newobj) < DIST(LOC,X)
          then UPDATE(X = NewObj)
            (PRESERVE-OBJ-MON
      Queue-length 1
      Run-time 25
      Run-every 250
     Reports:
      if NOT(PRESERVE(x)) then FAILURE)
 STEPS:   OPERATOR-MOVE-TO(X)
   LIFT(X)
 FAILURE-DEL:  (KNOWN (LOC X))
 FAILURE-ADD: (Achieve
(PICKUP-NEAREST(CurLoc,X)))
 SUCCESS-ADD:  AT (X)
           HOLDING(X)
 SUCCESS-DEL:  OBTAINABLE(X)
           ONTABLE(X)
           HANDEMPTY
           CLEAR(X)
 PROBABILITY: .72
 RESOURCES: TIME(DIST(LOC,X)  AVG-VEL)
     Consumes(Arm)
```

Figure 5. An Operator for a (Hypothetical) Modern Planning System.

The oprator is annotated to include preconditions, filter conditions, execution-time monitoring (with real-time scheduling constraints), steps to be accomplished, add- and delete-lists for success and failure, resource usage information, and a success probability.

so-called *linearity assumption*, that is, assuming that solving one goal (out of a conjunction) and then following this solution with the solution to the other goals in the conjunction will be successful. This assumption is valid in some domains because the solutions to different goals can often be decoupled. Most current planners do not make the linearity assumption and can consider arbitrary interleavings of all goals and subgoals. These planners are often referred to as *nonlinear planners* because they do not require the linearity assumption to guarantee a correct solution. The term nonlinear is sometimes confusingly used to refer not to this assumption but to the partial ordering of plan steps that some planners use in attempting to handle the problem. We address this issue later.

Early planning systems would sequentially solve the conjunctive goals and then make a simple check to see if the conjunction of goals still held (for example, Strips [Fikes and Nilsson 1971]). Where the linearity assumption failed, a mutual goal or subgoal interference could result that would often require redundant actions being put into the plan. In fact, in the worst case, the planner could get into an endless cycle of reintroducing and trying to satisfy the same goal.

Several systems followed this type that could handle some of the problems introduced when the problems being solved involved interacting goals. These systems either allowed alternate orderings of entire solution paths (Hacker [Sussman 1973]), permitting an interfering action (just being introduced to satisfy some goal) to be placed at different (earlier) points in the current plan until a position was found where there is no interaction (Warplan [Warren 1974]), or goal regression (that is, considering it earlier) when an interaction with some particular solution failed (Waldinger 1975). (The latter had the advantage that redundant actions were not reintroduced if the goal was achievable earlier in the plan).

Interplan (Tate 1975a) introduced a representation called *goal structure* to record the link between an effect of one action that was a precondition (subgoal) of a later one. This representation was additional to the ordering links between the actions themselves (some actions have effects that are used much later in the plan; Sussman [1973] referred to this as the plan's *teleology*.) Interactions were detected as an interference between some new action or goal being introduced and one or more previous goal structure. A minimum set of goal reorderings could be suggested that corrected for the interactions found. This approach was found to be more efficient because it considered fewer reorderings of the given goals and subgoals than the earlier methods.

Noah (Sacerdoti 1975) introduced code called *critics* that was used to search for interactions between parts of the plan, which was represented as a partial ordering. This innovation was important because it allowed the planner to use a least commitment strategy: Separate operators would be considered unordered with respect to each other unless a critic introduced an ordering relation to avoid an interaction (as done in figure 4d). Noah used a data structure, the table of multiple effects, to record the effects of the operators and, thus, aid in discovering interactions.

Noah is often called the first nonlinear planner, referring to the partially ordered plan structures employed by the system. This term is often confused with the fact that Noah was able to solve problems where the linearity assumption didn't hold. To help avoid this confusion, we do not refer to this type of system as nonlinear but rather, because such planners use partially ordered plans, we use the term *partial-order planners*. Thus, Noah was the first partial-order planner.

Nonlin (Tate 1975b, 1977) was a partial-order planner modeled on Noah. As in Noah, the minimum set of orderings necessary to resolve interactions could be suggested by introducing ordering links into a partial-order plan when they became essential. The Nonlin system, however, could also consider alternatives if failures on any chosen search branch occurred. Nonlin introduced the notion of a goal-state table, a data structure that could be used to record dependencies between plan steps facilitating the analysis of interactions. Similar analyses of a representation of the effect and condition structure of a plan to detect and correct for interactions were included in Planx-10 (Sridharan and Bresina 1985), Sipe's plan rationale (Wilkins 1983), and Dean's (1985) work.

A further refinement of the analysis of interactions was added to the Deviser system (Vere 1983). When individual goals or actions in a plan had time constraints on when they could be satisfied or performed, the detection and correction of interactions were sensitive to the temporal displacement introduced. This helped limit the number of legal solutions that could be proposed. Deviser allowed for planning in the presence of deadlines and external events. In addition, the use of objects being manipulated as scarce resources on which usage conflicts occur and need to be avoided was incorporated into the Molgen (Stefik 1981b) and Sipe (Wilkins 1983) planners. Viewing time as such a resource is a current research topic (Dean and Boddy 1987).

Operator Representation

The first systems to handle planning problems simply selected appropriate operators to apply to a problem by considering the differences between the goal state and the initial state and looking for operators that were known to remove such differences. GPS (Newell and Simon 1963), for example, directly associated the operators for the problem with the differences they could reduce. Thus, an operator such as PAINT(Robot,x) would be associated with achieving the fact PAINTED(x).

Strips (Fikes and Nilsson 1971) used this notion of differences, as well as ideas from the situation calculus (McCarthy and Hayes 1969) and Green's (1969) QA3 program, to make the assumption that the initial world model would only be changed by a set of additions to, and deletions from, the statements modeling the world state—everything else remained unchanged. (This assumption is sometimes called the Strips assumption.) Strips then defined an operator as having an add-list, a delete-list, and a preconditions formula (to define the applicability or subgoaling conditions). Operators were selected on the basis of the recognition of goals that matched statements on the add-lists (that is, those statements the operator could add to the current state).

For nonprimitive actions, Sacerdoti's (1975) Noah system used procedurally specified SOUP (semantics of user's problem) code to introduce appropriate methods of achieving goals or orderings to correct for interactions into the network of actions. Later planners introduced declarative representations to handle this with operators that were extensions to the Strips operator type of formalism (for example, Nonlin's task formalism [Tate 1977] and the Sipe notation [Wilkins 1983]). As well as add- and delete-lists and precondition formulas, an expansion of the operator to a lower level of detail could be specified as a partial order on suitable subactions and subgoals.

Recent systems have continued to add information to the operators. One has been adding information about resource usage to the operators so that planners can reason about limited resources. Time constraints have also been encoded on operators. Deviser (Vere 1983), for example, provided a method for specifying a time window on goals and activities, external events and their time of occurrence, and delayed events caused some time after a planned action. Nonlin+ (Tate and Whiter 1984) added the capability of representing multiple limited resources and making selections from appropriate activities on the basis of reducing some overall computed preference between them. The definition of shared objects as resources and the declaration of the use of such resources in operators were also provided in Sipe (Wilkins 1983).

Airplan (Masui, McDermott, and Sobel 1983) maintained information on the operators to be able to reason about time intervals and about how concurrent actions in these intervals could interact. O-Plan (Bell and Tate 1985) uniformly represented time constraints (and resource usage) by a numeric (min, max) pair that bounded the actual values for activity duration, activity start and finish times, and delays between activities. The actual values can be uncertain for various reasons, such as the plan being at a high abstraction level, not having chosen values for objects referred to in the plan, or uncertainty existing in modeling the domain. Constraints can be stated on the time and resource values that can lead to the planner finding that some plans in its search space are invalid. Excalibur (Drabble 1988) allowed for planning in the face of external continuous processes that were qualitatively modeled.

As the domains being considered by planning systems have become more dynamic (see Planning and Execution), information has also been placed on operators to allow for execution-time monitoring (Firby 1989), bounded computation or real-time scheduling needs (Kaelbling 1987; Hendler 1989), and different add- and delete-lists depending on execution-time success or failure and to

This approach toward dealing with failure when it arises has become known as replanning.

predict the probability of an operator's success (Miller, Firby, and Dean 1985) (figure 5).

Planning and Execution

Most of the systems described assume that the planner possesses complete knowledge of the current state of the world and the cause-and-effect relationships that govern changes in this world. Clearly, this approach needs revising in situations where separate execution cannot be guaranteed to succeed. This can occur when agents outside the control of the planner can cause changes in the environment or where the planner might be uncertain of information that can only be ascertained while the plan is being run.

For example, the Strips system (Fikes, Hart, and Nilsson 1972a) was used to plan the motion of a robot called Shakey and to control it as it pushed a set of boxes through a number of interconnecting rooms. A well-known SRI film shows Shakey following a Strips-generated plan using an execution monitor called Planex. Charley Rosen, the founder of the SRI AI Lab, dressed in a sinister cloak, appears and disrupts the position of the boxes during execution. Planex was able to make use of information maintained by Strips to recover.

This approach toward dealing with failure when it arises has become known as *replanning* and is typically assumed to occur when a planner recognizes a mismatch between the expected and actual state of the world. Hayes (1975) proposes that the subgoal trees and decision graphs used in the formation of the plan could be used to guide replanning. Daniel (1983) explores the use of a similar mechanism in a partial-order planner. The Priar system (Kambhampati 1989; Kambhampati and Hendler 1989) used a well-defined subset of this information, the *validation structure*, to provide such guidance. (A formal treatment of replanning can be found in Morgenstern [1987]).

An alternative to replanning is to actually expect potential failures and plan for them. This approach can involve planning for expected possibilities, such as waiting for a light to turn green before crossing a street (Drummond 1985), or it can involve scheduling monitoring task tests to be run at execution time, associated with fixes to be used in the case of failed tests (Doyle, Atkinson, and Doshi 1986). An example of the latter (Kaelbling 1987) would be having a robot check to see if two walls were equidistant at some point during the traversal of a hallway. If not, the robot could achieve this equidistance and continue. Schoppers (1987) proposes taking disjunctive planning to an extreme by generating *universal plans*, plans with conditional tests to deal with all possible execution-time contingencies.

Another approach to handling change in the environment is to provide a different sort of integration between the generation and execution of plans. McDermott's (1978) NASL system, for example, interleaved plan generation and execution by choosing one step at a time and executing it. This approach made the planner more susceptible to errors caused by interactions but less susceptible to errors caused by change in the environment. This works well in environments where a small amount of change can occur but is insufficient in domains in which rapid reaction is needed (for example, the traffic world [Hendler and Sanborn 1987] shown in figure 6). This approach was extended by Drummond (1989): Situated control rules are used to inform an independently competent execution system that the execution system can, if necessary, act without a plan; the plan simply serves to increase the system's goal-achieving abilities.

Much recent work has dealt with designing mechanisms that can handle rapidly changing environments. Most of this work achieves responsiveness by giving up complex planning for shallow planning or by planning using tightly coupled sensing and action (Rosenschein 1982; Chapman and Agre 1987; Sanborn and Hendler 1988). Some of this work, however, has dealt directly with the issues of how to map from planning to reaction. Firby (1989) proposes *reactive action packages* that essentially replace the operators in the system with procedures that include a reactive component. Rosenschein's and Kaelbling's (1988) Gapps system is a compiler that translates constraint expressions into directly executable circuits for use in robotic control systems. Georgeff and Lansky (1987) describe the use of a metareasoning system that can choose from a variety of execution-time options based on the goals being pursued by the system. Ambros-Ingerson and Steel (1983) propose an approach to integrating planning and execution for changing domains using an agenda-driven control structure in which actions that are serially initiated can run concurrently, with information-acquiring actions (for monitoring the environment) included. In addition, an effort is being made to extend temporal representations to handle simultaneously occurring events and event interactions. The use of these extended representations for planning is discussed by Pednault (1987).

In addition, it now appears that an important part of planning in dynamic domains involves making trade-offs—specifically, trading precision in decision making for time in responding to events. In the last few years, a number of researchers have attempted to improve the responsiveness of planning systems operating in dynamic domains by directly reasoning about the utility of planning (Dean 1987; Horvitz 1988; Russell and Wefald 1989). This work involved an examination of reasoning about these trade-offs during plan generation (Kanazawa and Dean 1989; Heckerman, Breese, and Horvitz 1989) or execution (Boddy and Dean 1989; Horvitz, Cooper, and Heckerman 1989).

Learning and Memory

The concentration in most of the work on planning has been on generating plans from scratch, not learning from experience. Thus, much of the classical work in planning has been ahistoric; that is, asked to solve the same problem again, the planner performs no better than it did the first time. Recently, because of the gains being made in machine learning and the new work on case-based reasoning, designing planning systems that learn from experience has become an option.

The earliest approach to learning in plans was the Macrops (for macro-operators) work of Fikes, Hart, and Nilsson (1972b) that extended Strips to include some limited learning from its failures. When a portion of a plan was found to have succeeded, the entire set of operators could be turned into a single operator whose preconditions and effects were identical to the preconditions and effects of the operator sequence. The operators were generalized by using variables to replace the constants found in the specific solution from which the new operator had been derived. Minton's (1985) Prodigy/EBL system used a similar approach but applied an explanation-based learning algorithm to provide accurate generalizations.

Case-based reasoning approaches to planning have also been attempted. In these systems, an old plan is chosen and modified to run in the new situation. Many of these systems concentrate on guiding the search for the old plan (for example, Julia [Kolodner 1987] and Chef [Hammond 1986]) and then using a fairly simple mapping to produce the new plan. Plexus (Alterman 1988) used information about the new context to guide the reuse of an existing plan. The Priar reuse system (Kambhampati 1989) applied a case-based approach in the classical planning framework. Priar, an extension to Nonlin (Tate 1977), allowed the planner to annotate plans being created with information about the dependency structure between operators. This information was then used to guide retrieval, reuse, and replanning.

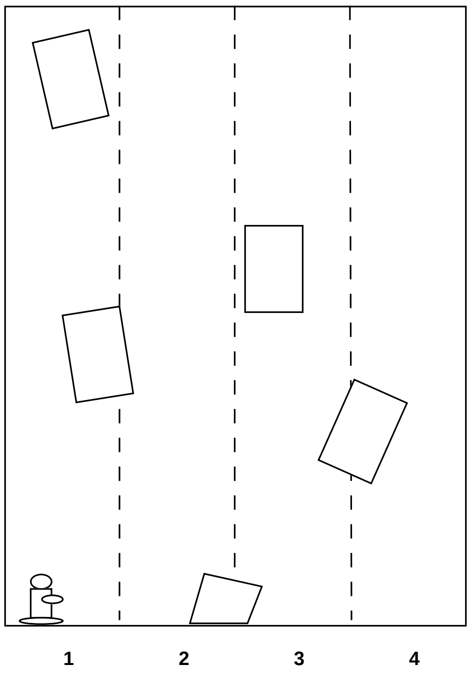

Figure 6. The Traffic World (Hendler and Sanborn 1987).

The traffic world consists of a straight stretch of "four-lane highway," along which "cars" travel at various speeds. There is a much slower–moving "robot" at one end of the highway that is trying to get across to the opposite side. Cars change lanes and speeds as they proceed.

Assorted Shorts

In addition to the issues discussed earlier, many other issues arise in the design of planning systems. Examples include planning with conditionals and iterators, uncertainty in planning, and distributed planning.

Planning with Conditionals and Iterators

Most of the search space control techniques, goal ordering, and interaction correction mechanisms developed in AI planners to date have been oriented toward the generation of plans that are fully or partially ordered sequences of primitive actions. However, there has been some effort on the generation of plans that contain conditionals (if . . . then . . . else . . .) and iterators (repeat . . . until . . .). Conditionals were handled in Warplan-C (Warren 1976) by branching the plan at the conditional and performing separate planning on the two branches using the assumption that the statement in the conditional was true in one branch and false in the other. This analysis led to a case analysis of the separate branches to produce a plan that was tree structured. Drummond (1985) designed an extension to procedural nets allowing for disjunction and iteration that could be used for predicting the behavior of the system where these were present.

Uncertainty in Planning

One source of execution-time problems is uncertainty that exists during plan genera-

One source of execution-time problems is uncertainty that exists during plan generation.

tion. If the planner cannot model the real world with complete information and instead uses some sort of probability-based model, it must deal with low-probability events that might occur during execution. (This problem arises in a significant way for systems that use real sensors to perceive the world). Work using probability estimates during plan generation includes that by Dean, Firby, and Miller (1989) and Kanazawa and Dean (1989). Segre (1988) examines the issue of execution-time failures of plans based on uncertain information.

Distributed Planning

Some systems have been exploring distributing sources of problem-solving expertise or knowledge during planning. They allow fully distributed planning with the subproblems being passed between specialized planning experts. The use of the experts—or the behaviors they might perform—is controlled through a centralized blackboard and executive (with a system similar to priority scheduling of parallel processes) or can be controlled in a more distributed fashion through pairwise negotiation. Examples of relevant work include Smith (1977), Corkill (1979), Kornfeld (1979), Konolige and Nilsson (1980), Georgeff (1982), and Corkill and Lesser (1983).

Recommended Reading

A good and reasonably up-to-date account of AI planning techniques and systems is given in E. Charniak's and D. McDermott's *Introduction to Artificial Intelligence* (Addison-Wesley, 1985). In particular, chapter 9 and sections of chapters 5 and 7 are relevant. Somewhat earlier material is provided in Elaine Rich's *Artificial Intelligence* (McGraw-Hill, 1983). N. Nilsson's *Principles of Artificial Intelligence* (Morgan Kaufmann, 1980) provides a uniform treatment of planning techniques available at the time it was published. There are several useful summaries of early AI planning work in the *Handbook of Artificial Intelligence* (Morgan Kaufmann, 1981) by A. Barr, E. Feigenbaum, and P. Cohen—volume 1, section II.D, and volume 3, sections XI.B, XI.C, and XV. A collection of recent papers concerning planning can be found in M. Georgeff's and A. Lansky's *Reasoning about Actions and Plans: Proceedings of the 1986 Workshop* (Morgan Kaufmann, 1987). A collection of previously published papers on planning and reasoning about actions will soon be available as *Readings in Planning* by J. Allen, J. Hendler, and A.Tate (Morgan Kaufmann, 1990).

Summary

Planning systems have been an active AI research topic for nearly 30 years. A number of techniques have been developed during this period that still form an essential part of many of today's AI planning systems. In this article, we tried to broadly cover the major ideas in the field of AI planning and attempted to show the direction in which current research is going. Such a task is never ending, and thus, any finite document must be incomplete. We provided references to connect each idea to the appropriate literature and allow readers immediate access to the work most relevant to their research or applications.

Acknowledgments

We are grateful for input from research colleagues active in AI planning who helped us structure this review. We hope that we did not misrepresent anyone in showing how the research themes, techniques, and planning systems relate to one another. We also apologize to colleagues for simplifications made in our descriptions of past planning systems. Simplification was a necessary evil for reasons of space; for complete descriptions, the reader should consult the associated references. The work of Austin Tate and the Planning Group at the Artificial Intelligence Applications Institute is supported by the United Kingdom Science and Engineering Research Council (under IED grant 1320, GR/F 36545) and the United States Air Force (under contract EOARD-88-0044). James Hendler is also affiliated with the Systems Research Center and the University of Maryland Institute for Advanced Computer Studies. His work is funded in part by U.S. Office of Naval Research grant N00014-88-K-0560 and National Science Foundation grant IRI-8907890. A portion of Mark Drummond's work at the NASA Ames Research Center is funded by the Air Force Office of Scientific Research, Artificial Intelligence Research Program.

Bibliography

Where names of planning systems and other AI software are described in this article, the appropriate bibliographic entries are annotated with the name of the system in brackets.

Allen, J. F., and Koomen, J. A. 1983. Planning Using a Temporal World Model. In Proceedings of the Eighth International Joint Conference on Artificial

Intelligence, 741–747. Menlo Park, Calif.: International Joint Conferences on Artificial Intelligence. [TIMELOGIC]

Altermann, R. 1988. Adaptive Planning. *Cognitive Science* 12. [PLEXUS]

Ambros-Ingerson, J., and Steel, S. 1983. Integrating Planning, Execution, and Monitoring. In Proceedings of the Eighth National Conference on Artificial Intelligence, 83–88. Menlo Park, Calif.: American Association for Artificial Intelligence.

Appelt, D. E. 1985. Planning English Referring Expressions. *Artificial Intelligence* 26:1–33. [KAMP]

Bell, C. E., and Tate, A. 1985. Using Temporal Constraints to Restrict Search in a Planner. In Proceedings of the Third Workshop of the Alvey IKBS Programme Planning Special Interest Group. London: Institute of Electrical Engineers. [O-PLAN]

Boddy, M., and Dean, T. 1989. Solving Time-Dependent Planning Problems. In Proceedings of the Eleventh International Joint Conference on Artificial Intelligence, 979–984. Menlo Park, Calif.: International Joint Conferences on Artificial Intelligence.

Bresina, J. L. 1981. An Interactive Planner That Creates a Structured Annotated Trace of Its Operation, CBM-TR-123, Computer Science Research Lab., Rutgers Univ. [PLANX-10]

Chapman, D. 1987. Planning for Conjunctive Goals. *Artificial Intelligence* 32:333–377. [TWEAK]

Chapman, D. 1985. Nonlinear Planning: A Rigorous Reconstruction. In Proceedings of the Ninth International Joint Conference on Artificial Intelligence, 1022–1024. Menlo Park, Calif.: International Joint Conferences on Artificial Intelligence. [TWEAK]

Chapman, D., and Agre, P. 1987. Abstract Reasoning as Emergent from Concrete Activity. In *Reasoning about Actions and Plans: Proceedings of the 1986 Workshop*, eds. M. Georgeff and A. Lansky. San Mateo, Calif.: Morgan Kaufmann.

Corkill, D. D. 1979. Hierarchical Planning in a Distributed Environment. In Proceedings of the Sixth International Joint Conference on Artificial Intelligence, 168–175. Menlo Park, Calif.: International Joint Conferences on Artificial Intelligence.

Corkill, D. D., and Lesser, V. R. 1983. The Use of Meta-Level Control for Coordination in a Distributed Problem-Solving Network. In Proceedings of the Eighth International Joint Conference on Artificial Intelligence, 748–756. Menlo Park, Calif.: International Joint Conferences on Artificial Intelligence.

Currie, K., and Tate, A. 1985. O-Plan—Control in the Open Planner Architecture. In *BCS Expert Systems Conference*. Cambridge: Cambridge University Press. [O-PLAN]

Daniel, L. 1983. Planning and Operations Research. In *Artificial Intelligence: Tools, Techniques, and Applications*. New York: Harper and Row. [NONLIN]

Davis, P. R., and Chien, R. T. 1977. Using and Reusing Partial Plans. In Proceedings of the Fifth International Joint Conference on Artificial Intelligence. Menlo Park, Calif.: International Joint Conferences on Artificial Intelligence.

Davis, R., and Smith, R. 1983. Negotiation as a Metaphor for Distributed Problem Solving. *Artificial Intelligence* 20:63–109.

Dean, T. 1987. Intractability and Time-Dependent Planning. In *Reasoning about Actions and Plans: Proceedings of the 1986 Workshop,* eds. M. Georgeff and A. Lansky. San Mateo, Calif.: Morgan Kaufmann.

Dean, T. 1985. Temporal Reasoning Involving Counterfactuals and Disjunctions. In Proceedings of the Ninth International Joint Conference on Artificial Intelligence. Menlo Park, Calif.: International Joint Conferences on Artificial Intelligence. [TNMS]

Dean, T., and Boddy, M. 1987. Reasoning about Partially Ordered Events. *Artificial Intelligence* 36:375–399.

Dean, T.; Firby, J.; and Miller, D. 1989. Hierarchical Planning Involving Deadlines, Travel Time, and Resources. *Computational Intelligence* 3. [FORBIN]

Dershowitz, N. 1985. Synthetic Programming. *Artificial Intelligence* 25:323–373.

Doran, J. E., and Michie, D. 1966. Experiments with the Graph Traverser Program. In Proceedings of the Royal Society, 235–259. [GRAPH TRAVERSER]

Doran, J. E., and Trayner, C. 1985. Distributed Planning and Execution—Teamwork 1, Computer Science Technical Report, Univ. of Essex. [TEAMWORK]

Doyle, J. 1979. A Truth Maintenance System. *Artificial Intelligence* 12:231–272.

Doyle, R. J.; Atkinson, D. J.; and Doshi, R. S. 1986. Generating Perception Requests and Expectations to Verify the Execution of Plans. In Proceedings of the Fifth National Conference on Artificial Intelligence. Menlo Park, Calif.: American Association for Artificial Intelligence.

Drabble, B. 1988. Planning and Reasoning with Processes. In *The Eighth Workshop of the Alvey Planning Special Interest Group*, 25–40. Nottingham, United Kingdom: Institute of Electrical Engineers.

Drummond, M. 1989. Situated Control Rules. In Proceedings of the Conference on Principles of Knowledge Representation and Reasoning.

Drummond, M. 1985. Refining and Extending the Procedural Net. In Proceedings of the Ninth International Joint Conference on Artificial Intelligence, 1010–1012. Menlo Park, Calif.: International Joint Conferences on Artificial Intelligence.

Drummond, M., and Currie, K. W. 1989. Goal Ordering in Partially Ordered Plans. In Proceedings of the Eleventh International Joint Conference on Artificial Intelligence. Menlo Park, Calif.: International Joint Conferences on Artificial Intelligence.

Drummond, M., and Currie, K. W. 1988. Exploiting Temporal Coherence in Non-Linear Plan Construction. *Computational Intelligence* 4(4).

Duffay, P., and Latombe, J-C. 1983. An Approach to Automatic Robot Programming Based on Inductive Learning. IMAG. [TROPIC]

Erman, L. D.; Hayes-Roth, F.; Lesser, V. R.; and

Reddy, D. R. 1980. The HEARSAY-II Speech-Understanding System: Integrating Knowledge to Resolve Uncertainty. *ACM Computing Surveys* 12(2).

Fahlman, S. E. 1974. A Planning System for Robot Construction Tasks. *Artificial Intelligence* 5:1–49.

Faletti, J. 1982. PANDORA—A Program for Doing Commonsense Reasoning Planning in Complex Situations. In Proceedings of the Second National Conference on Artificial Intelligence. Menlo Park, Calif.: American Association for Artificial Intelligence. [PANDORA]

Fikes, R. E. 1982. A Commitment-Based Framework for Describing Informal Cooperative Work. *Cognitive Science* 6:331–347.

Fikes, R. E. 1970. REF-ARF: A System for Solving Problems Stated as Procedures. *Artificial Intelligence* 1:27–120.

Fikes, R. E., and Nilsson, N. J. 1971. Strips: A New Approach to the Application of Theorem Proving to Problem Solving. *Artificial Intelligence* 2:189–208. [STRIPS]

Fikes, R. E.; Hart, P. E.; and Nilsson, N. J. 1972a. Learning and Executing Generalized Robot Plans. *Artificial Intelligence* 3. [STRIPS; PLANEX]

Fikes, R. E; Hart, P. E.; and Nilsson, N. J. 1972b. Some New Directions in Robot Problem Solving. In *Machine Intelligence* 7, eds. B. Meltzer and D. Michie. Edinburgh: Edinburgh University Press. [STRIPS]

Firby, J. 1989. Adaptive Execution in Complex Dynamic Worlds. Ph.D. diss., Dept. of Computer Science, Yale Univ. [RAPS]

Fox, M. S.; Allen, B.; and Strohm, G. 1981. Job Shop Scheduling: An Investigation in Constraint-Based Reasoning. In Proceedings of the Seventh International Joint Conference on Artificial Intelligence. Menlo Park, Calif.: International Joint Conferences on Artificial Intelligence. [ISIS-II]

Georgeff, M. 1982. Communication and Interaction in Multi-Agent Planning Systems. In Proceedings of the Third National Conference on Artificial Intelligence. Menlo Park, Calif.: American Association for Artificial Intelligence.

Georgeff, M., and Lansky, A. 1987. Reactive Reasoning and Planning. In Proceedings of the Sixth National Conference on Artificial Intelligence. Menlo Park, Calif.: American Association for Artificial Intelligence. [PRS]

Georgeff, M., and Lansky, A. 1985. A Procedural Logic. In Proceedings of the Ninth International Joint Conference on Artificial Intelligence. Menlo Park, Calif.: International Joint Conferences on Artificial Intelligence.

Green, C. C. 1969. Theorem Proving by Resolution as a Basis for Question Answering. In *Machine Intelligence* 4, eds. B. Meltzer and D. Michie. Edinburgh: Edinburgh University Press.

Gupta, N., and Nau, D. 1990. Optimal Block's World Solutions are NP-Hard, Technical Report, Computer Science Dept., Univ. of Maryland. Forthcoming.

Hammond, K. 1986. Chef: A Model of Case-Based Planning. In Proceedings of the Fifth National Conference on Artificial Intelligence. Menlo Park, Calif.: American Association for Artificial Intelligence. [CHEF]

Hart, P.; Nilsson, N.; and Raphael, B. 1968. A Formal Basis for the Heuristic Determination of Minimum Cost Paths. *IEEE Transactions on System Science and Cybernetics* SSC-4(2): 100–107. [A*]

Hayes, P. J. 1975. A Representation for Robot Plans. In Advance Papers of the 1975 International Joint Conference on Artificial Intelligence, Tbilisi, USSR.

Hayes-Roth, B. 1983a. A Blackboard Model of Control, HPP-83-38, Heuristic Programming Project, Stanford Univ. [OPM]

Hayes-Roth, B. 1983b. The Blackboard Architecture: A General Framework for Problem Solving? HPP-83-30, Heuristic Programming Project, Stanford Univ.

Hayes-Roth, B., and Hayes-Roth, F. 1979. A Cognitive Model of Planning. *Cognitive Science* 30:275–310. [OPM]

Heckerman, D.; Breese, J.; and Horvitz, E. 1989. The Compilation of Decision Models. In Proceedings of the 1989 Workshop on Uncertainty in Artificial Intelligence, 162–173.

Hendler, J. 1989. Real-Time Planning. Presented at the American Association for Artificial Intelligence Symposium on Planning and Search, Stanford, Calif.

Hendler, J. A. 1987. *Integrating Marker-Passing and Problem Solving: A Spreading Activation Approach to Improved Choice in Planning*. Norwood, N.J.: Lawrence Erlbaum. [SCRAPS]

Hendler, J., and Sanborn, J. 1987. Planning and Reaction in Dynamic Domains. In Proceedings of the Defense Advanced Research Projects Agency Workshop on Planning.

Hendrix, G. 1973. Modelling Simultaneous Actions and Continuous Processes. *Artificial Intelligence* 4:145–180.

Horvitz, E. 1988. Reasoning under Varying and Uncertain Resource Constraints. In Proceedings of the Seventh National Conference on Artificial Intelligence, 111–116. Menlo Park, Calif.: American Association for Artificial Intelligence.

Horvitz, E.; Cooper, G.; and Heckerman, D. 1989. Reflection and Action under Scarce Resources: Theoretical Principles and Empirical Study. In Proceedings of the Eleventh International Joint Conference on Artificial Intelligence, 1121–1127. Menlo Park, Calif.: International Joint Conferences on Artificial Intelligence.

Kaelbling, L. P. 1988. Goals as Parallel Program Specifications. In Proceedings of the Seventh National Conference on Artificial Intelligence, 60–65. Menlo Park, Calif.: American Association for Artificial Intelligence. [GAPPS]

Kaelbling, L. 1987. An Architecture for Intelligent Reactive Systems. In *Reasoning about Actions and Plans: Proceedings of the 1986 Workshop*, eds. M. Georgeff and A. Lansky. San Mateo, Calif.: Morgan Kaufmann.

Kahn, K., and Gorry, G. A. 1977. Mechanizing Temporal Knowledge. *Artificial Intelligence* 9:87–108.

Kambhampati, S. 1989. Flexible Reuse and Modification in Hierarchical Planning: A Validation Structure Based Approach, Ph.D. diss., Dept. of Computer Science, Univ. of Maryland. [PRIAR]

Kambhampati, S., and Hendler, J. 1989. Flexible Reuse of Plans via Annotation and Verification. In Proceedings of the Fifth Institute of Electrical and Electronic Engineers Conference on Applications of Artificial Intelligence. [PRIAR]

Kanazawa, K., and Dean, T. 1989. A Model for Projection and Action. In Proceedings of the Eleventh International Joint Conference on Artificial Intelligence, 985–999. Menlo Park, Calif.: International Joint Conferences on Artificial Intelligence.

Kolodner, J. 1987. Case-Based Problem Solving. Presented at the Fourth International Workshop on Machine Learning, University of California at Irvine. [JULIA]

Konolige, K. 1983. A Deductive Model of Belief. In Proceedings of the Eighth International Joint Conference on Artificial Intelligence, 377–381. Menlo Park, Calif.: International Joint Conferences on Artificial Intelligence.

Konolige, K., and Nilsson, N. J. 1980. Multi-Agent Planning Systems. In Proceedings of the First National Conference on Artificial Intelligence, 138–142. Menlo Park, Calif.: American Association for Artificial Intelligence.

Korf, R. E. 1987. Planning as Search: A Quantitative Approach. *Artificial Intelligence* 33:65–88.

Kornfeld, W. A. 1979. ETHER: A Parallel Problem Solving System. In Proceedings of the Sixth International Joint Conference on Artificial Intelligence, 490–492. Menlo Park, Calif.: International Joint Conferences on Artificial Intelligence.

Lansky, A. L. 1988. Localized Event-Based Reasoning for Multiagent Domains. *Computational Intelligence Journal* (Special Issue on Planning) 4(4). [GEMPLAN]

Lansky, A. L. 1987. A Representation of Parallel Activity Based on Events, Structure, and Causality. In *Reasoning about Actions and Plans: Proceedings of the 1986 Workshop*, eds. M. Georgeff and A. Lansky. San Mateo, Calif.: Morgan Kaufmann. [GEMPLAN]

Latombe, J-C. 1976. Artificial Intelligence in Computer-Aided Design—The TROPIC System, Technical Note 125, SRI AI Center, Menlo Park, Calif.

Lenat, D. B. 1975. BEINGS: Knowledge as Interacting Experts. In Proceedings of the Fourth International Joint Conference on Artificial Intelligence, 126–133. Menlo Park, Calif.: International Joint Conferences on Artificial Intelligence. [PUP]

London, P. 1977. A Dependency-Based Modelling Mechanism for Problem Solving, Memorandum, TR-589, Dept. of Computer Science, Univ. of Maryland.

Luckham, D. C., and Buchanan, J. R. 1974. Automatic Generation of Programs Containing Conditional Statements. In Proceedings of the AISB Summer Conference, 102–126.

McCarthy, J., and Hayes, P. J. 1969. Some Philosophical Problems from the Standpoint of Artificial Intelligence. In *Machine Intelligence* 4, eds. B. Meltzer and D. Michie. Edinburgh: Edinburgh University Press.

McDermott, D. V. 1982. A Temporal Logic for Reasoning about Processes and Plans. *Cognitive Science* 6:101–155.

McDermott, D. V. 1978. Planning and Acting. *Cognitive Science* 2. [NASL]

McDermott, D. V., and Doyle, J. 1979. An Introduction to Nonmonotonic Logic. In Proceedings of the Sixth International Joint Conference on Artificial Intelligence, 562–567. Menlo Park, Calif.: International Joint Conferences on Artificial Intelligence.

Masui, S.; McDermott, J.; and Sobel, A. 1983. Decision-Making in Time Critical Situations. In Proceedings of the Eighth International Joint Conference on Artificial Intelligence, 233–235. Menlo Park, Calif.: International Joint Conferences on Artificial Intelligence. [AIRPLAN]

Mellish, C. S. 1984. Toward Top-Down Generation of Multi-Paragraph Text. In Proceedings of the Sixth European Conference on Artificial Intelligence, 229.

Miller, D.; Firby, J.; and Dean, T. 1985. Deadlines, Travel Time, and Robot Problem Solving. In Proceedings of the Ninth International Joint Conference on Artificial Intelligence, 1052–1054. Menlo Park, Calif.: International Joint Conferences on Artificial Intelligence. [FORBIN]

Minton, S. 1985. Selectively Generalizing Plans for Problem-Solving. In Proceedings of the Ninth International Joint Conference on Artificial Intelligence, 596–599. Menlo Park, Calif.: International Joint Conferences on Artificial Intelligence. [Prodigy/EBL]

Moore, R. 1980. Reasoning about Knowledge and Action, Technical Report 191, SRI AI Center, Menlo Park, Calif.

Morgenstern, L. 1987. Replanning. In Proceedings of the Defense Advanced Research Projects Agency Knowledge-Based Planning Workshop.

Mostow, D. J. 1983. A Problem Solver for Making Advice Operational. In Proceedings of the Third National Conference on Artificial Intelligence, 179–283. Menlo Park, Calif.: American Association for Artificial Intelligence.

Nau, D.; Yang, Q.; and Hendler, J. 1989. Planning for Multiple Goals with Limited Interactions. In Proceedings of the Fifth Institute for Electrical and Electronic Engineers Conference on Applications of Artificial Intelligence.

Newell, A., and Simon, H. A. 1963. GPS: A Program That Simulates Human Thought. In *Computers and Thought*, eds. E. A. Feigenbaum and J. Feldman. New York: McGraw-Hill. [GPS]

Nilsson, N. J. 1971. *Problem Solving Methods in Artificial Intelligence.* New York: McGraw-Hill.

Pednault, E. 1987. Solving Multi-Agent Dynamic World Problems in the Classical Planning Framework. In *Reasoning about Actions and Plans: Proceedings of the 1986 Workshop*, eds. M. Georgeff and A. Lansky. San Mateo, Calif.: Morgan Kaufmann.

Reiger, C., and London, P. 1977. Subgoal Protection and Unravelling during Plan Synthesis. In Proceedings of the Fifth International Joint Conference on Artificial Intelligence. Menlo Park, Calif.: International Joint Conferences on Artificial Intelligence.

Rich, C. 1981. A Formal Representation for Plans in the Programmer's Apprentice. In Proceedings of the Seventh International Joint Conference on Artificial Intelligence, 1044–1052. Menlo Park, Calif.: International Joint Conferences on Artificial Intelligence.

Rich, C.; Shrobe, H. E.; and Waters, R. C. 1979. Overview of the Programmer's Apprentice. In Proceedings of the Sixth International Joint Conference on Artificial Intelligence, 827–828. Menlo Park, Calif.: International Joint Conferences on Artificial Intelligence.

Rosenschein, S. 1982. Synchronization of Multi-Agent Plans. In Proceedings of the Second National Conference on Artificial Intelligence. Menlo Park, Calif.: American Association for Artificial Intelligence.

Rosenschein, S. J. 1981. Plan Synthesis: A Logical Perspective. In Proceedings of the Seventh International Joint Conference on Artificial Intelligence. Menlo Park, Calif.: International Joint Conferences on Artificial Intelligence.

Rosenschein, S. J. 1980. Synchronization of Multi-Agent Plans. In Proceedings of the Second National Conference on Artificial Intelligence. Menlo Park, Calif.: American Association for Artificial Intelligence.

Russell, S., and Wefald, E. 1989. Principles of Metareasoning. In *Proceedings of the First International Conference on Principles of Knowledge Representation and Reasoning.* San Mateo, Calif.: Morgan Kaufmann.

Sacerdoti, E. D. 1979. Problem Solving Tactics. In Proceedings of the Sixth International Joint Conference on Artificial Intelligence. Menlo Park, Calif.: International Joint Conferences on Artificial Intelligence.

Sacerdoti, E. D. 1977. *A Structure for Plans and Behaviour.* Amsterdam: Elsevier–North Holland. [NOAH]

Sacerdoti, E. D. 1975. The Non-Linear Nature of Plans. In Advance Papers of the Fourth International Joint Conference on Artificial Intelligence. [NOAH]

Sacerdoti, E. D. 1973. Planning in a Hierarchy of Abstraction Spaces. In Advance Papers of the Third International Joint Conference on Artificial Intelligence. [ABSTRIPS]

Sanborn, J., and Hendler, J. 1988. Monitoring and Reacting: Planning in Dynamic Domains. *International Journal of AI and Engineering* 3(2).

Sathi, A.; Fox, M. S.; and Greenberg, M. 1985. Representation of Activity Knowledge for Project Management. *IEEE Special Issue of Transactions on Pattern Analysis and Machine Intelligence.* [CALLISTO]

Schank, R. C., and Abelson, R. P. 1977. *Scripts, Plans, Goals, and Understanding.* Hillsdale, N.J.: Lawrence Erlbaum.

Schoppers, M. 1987. Universal Plans for Reactive Robots in Unpredictable Domains. In Proceedings of the Tenth International Joint Conference on Artificial Intelligence. Menlo Park, Calif.: International Joint Conferences on Artificial Intelligence.

Segre, A. 1988. *Machine Learning od Robot Assembly Plans.* Norwell, Mass.: Kluwer Academic.

Siklossy, L., and Dreussi, J. 1975. An Efficient Robot Planner That Generates Its Own Procedures. In Proceedings of the Third International Joint Conference on Artificial Intelligence. Menlo Park, Calif.: International Joint Conferences on Artificial Intelligence. [LAWALY]

Siklossy, L., and Roach, J. 1973. Proving the Impossible Is Impossible Is Possible: Disproofs Based on Hereditary Partitions. In Proceedings of the Third International Joint Conference on Artificial Intelligence. Menlo Park, Calif.: International Joint Conferences on Artificial Intelligence. [DISPROVER/LAWALY]

Simmons, R., and Davis, R. 1987. Generate, Test, and Debug: Combining Associational Rules and Causal Models. In Proceedings of the Tenth International Joint Conference on Artificial Intelligence, 1071–1078. Menlo Park, Calif.: International Joint Conferences on Artificial Intelligence. [G-T-D]

Smith, R. G. 1979. A Framework for Distributed Problem Solving. In Proceedings of the Sixth International Joint Conference on Artificial Intelligence. Menlo Park, Calif.: International Joint Conferences on Artificial Intelligence.

Smith, R. G. 1977. The Contract Net: A Formalism for the Control of Distributed Problem Solving. In Proceedings of the Fifth International Joint Conference on Artificial Intelligence, 472. Menlo Park, Calif.: International Joint Conferences on Artificial Intelligence.

Sridharan, A., and Bresina, J. L. 1985. Knowledge Structures for Planning in Realistic Domains. *Computers and Mathematics with Applications* (Special Issue on Knowledge Representation) 11(5): 457–480. [PLANX-10]

Stallman, R. M., and Sussman, G. J. 1977. Forward Reasoning and Dependency Directed Backtracking. *Artificial Intelligence* 9:135–196.

Steele, G. L., and Sussman, G. J. 1978. Constraints, MIT AI Lab Memo, 502, AI Lab., Massachusetts Institute of Technology.

Stefik, M. J. 1981a. Planning and Meta-Planning. *Artificial Intelligence* 16:141–169. [MOLGEN]

Stefik, M. J. 1981b. Planning with Constraints. *Artificial Intelligence* 16:111–140. [MOLGEN]

Sussman, G. A. 1973. A Computational Model of Skill Acquisition, MIT AI Lab Memo, AI-TR-297, AI Lab., Massachusetts Institute of Technology. [HACKER]

Sussman, G. A., and McDermott, D. V. 1972. Why Conniving Is Better than Planning, MIT AI Lab Memo, 255A, AI Lab., Massachusetts Institute of Technology. [CONNIVER]

Tate, A. 1984a. Goal Structure: Capturing the Intent of Plans. Presented at the European Conference on Artificial Intelligence, Pisa, Italy. [NONLIN]

Tate, A. 1984b. Planning and Condition Monitoring in a FMS. Presented at the International Conference on Flexible Automation Systems, Institute of Electrical Engineers, London, July 1984. [NONLIN]

Tate, A. 1977. Generating Project Networks. In Proceedings of the Fifth International Joint Conference on Artificial Intelligence. Menlo Park, Calif.: International Joint Conferences on Artificial Intelligence. [NONLIN]

Tate, A. 1975a. Interacting Goals and Their Use. In Proceedings of the Fourth International Joint Conference on Artificial Intelligence, 215–218. Menlo Park, Calif.: International Joint Conferences on Artificial Intelligence. [INTERPLAN]

Tate, A. 1975b. Project Planning Using a Hierarchical Non-Linear Planner, Report 25, Dept. of Artificial Intelligence, Edinburgh Univ. [NONLIN]

Tate, A.,and Whiter, A. M. 1984. Planning with Multiple Resource Constraints and an Application to a Naval Planning Problem. *First Conference on the Applications of Artificial Intelligence, Denver, Colorado, USA*. San Mateo, Calif.: Morgan Kaufmann. [NONLIN+]

Vere, S. 1983. Planning in Time: Windows and Durations for Activities and Goals. *IEEE Transactions on Pattern Analysis and Machine Intelligence* PAMI-5(3): 246–267. [DEVISER]

Vilain, M. B. 1980. A System for Reasoning about Time. In Proceedings of the Second National Conference on Artificial Intelligence. Menlo Park, Calif.: American Association for Artificial Intelligence.

Waldinger, R. 1975. Achieving Several Goals Simultaneously, Technical Note 107, SRI AI Center, Menlo Park, Calif.

Warren, D. H. D. 1976. Generating Conditional Plans and Programs. In Proceedings of the AISB Summer Conference, 344–354. [WARPLAN-C]

Warren, D. H. D. 1974. WARPLAN: A System for Generating Plans, Memorandum, 76, Dept. of Computational Logic, Edinburgh Univ. [WARPLAN]

Wilensky, R. 1983. *Planning and Understanding*. Reading, Mass.: Addison-Wesley.

Wilensky, R. 1981a. A Model for Planning in Complex Situations, Memorandum, UCB/ERL M81/49, Electronics Research Lab., Univ. of California at Berkeley.

Wilensky, R. 1981b. Meta-Planning: Representing and Using Knowledge about Planning in Problem Solving and Natural Language Understanding. *Cognitive Science* 5:197–233.

Wilensky, R. 1978. Understanding Goal-Based Stories, Research Report, 140, Dept. of Computer Science, Yale Univ.

Wilkins, D. E. 1988. *Practical Planning—Extending the Classical AI Planning Paradigm*. San Mateo, Calif.: Morgan Kaufmann.

Wilkins, D. E. 1983. Representation in a Domain-Independent Planner. In Proceedings of the Eighth International Joint Conference on Artificial Intelligence, 733–740. Menlo Park, Calif.: International Joint Conferences on Artificial Intelligence.[SIPE]

Wilkins, D. E., and Robinson, A. E. 1981. An Interactive Planning System, Technical Note 245, SRI, Menlo Park, Calif. [SIPE]

Yang, Q. 1989. Improving the Efficiency of Planning, Ph.D. diss., Dept. of Computer Science, Univ. of Maryland.

James Hendler is an assistant professor with a joint appointment from the Computer Science Department and the Systems Research Center at the University of Maryland. He received a Ph.D. from Brown University in the area of AI planning systems. He authored *Integrating Marker-Passing and Problem Solving* (Lawrence Erlbaum, 1987) and edited *Expert Systems: The User Interface* (Ablex, 1988) and *Readings in Planning* (with J. Allen and A. Tate) (Morgan Kaufmann, 1990). He is also an associate editor for the international journal *Connection Science*. His current work focuses primarily on planning in rapidly changing, dynamic domains.

Austin Tate is the director of the Artificial Intelligence Applications Institute at the University of Edinburgh and is a university professorial fellow. In the mid-1970s, he developed the Nonlin planner and its associated task formalism. Tate's current research focuses on the development of O-Plan (open planning architecture), which is a flexible workbench for the construction of computer systems to aid in the command, planning, and control of tasks such as factory management, project management, and spacecraft operations.

Mark Drummond is currently employed by Sterling Software and works at the NASA Ames Research Center. He received a Ph.D. in AI from the University of Edinburgh in 1987, where he also worked as a consultant on the design and construction of knowledge-based planning systems for the Artificial Intelligence Applications Institute. His general research interests include temporal and spatial reasoning, real-time systems, scheduling, and decision support systems. His current work focuses on the problem of plan synthesis in the context of situated reaction.

Notes

1. An early version of this article appeared in *Knowledge Engineering Review* in June 1985 (volume 1, number 2, pages 4–17). Part of that review, along with a tutorial introduction to the field, appeared in the planning chapter of *Knowledge Engineering,* volume 1, edited by H. Adeli (McGraw-Hill, 1990). A version of this article, stressing technical definitions, appears in *Readings in Planning,* edited by J. Allen, J. Hendler, and A. Tate.