A REVIEW OF KNOWLEDGE-BASED PLANNING TECHNIQUES

Austin Tate

Knowledge-based Planning Group
Artificial Intelligence Applications Institute
University of Edinburgh

©A Tate

SUMMARY

Planning systems have been an active research topic within Artificial Intelligence for over two decades. There have been a number of techniques developed during that period which still form an essential part of many of today's planners. This paper introduces the techniques, attempts to classify some of the important research themes in AI planning and describes their historical development.

There has been a recent surge of interest in planning systems at both the research and applications levels. This paper should act as a brief introduction to the large volume of literature on the subject. An extensive bibliography is provided and cross-referenced throughout the text.

1. INTRODUCTION

General purpose planning systems which can automatically produce plans of action for execution by robots have been a long standing theme of AI research. A large number of techniques have been introduced in progressively more ambitious systems over a long period. There have been several attempts to combine the planning techniques available at a certain time into prototypes able to cope with realistic application domains.

Planning research has often built on earlier work. There have usually been full written accounts of the techniques employed in a form that could be abstracted out from the application domain and re-implemented or incorporated in other systems. A "tradition" of using a simple block stacking domain for examples has aided comparison and improved understanding (though today's more sophisticated planners barely show their true worth on these simple problems). Several researchers have used a set of problems to facilitate efficiency comparisons between the various systems, so introducing a (mildly) competitive spirit into the production of realistic prototype planners.

Planning research involves many of the areas of principal concern to AI:

- Search and choice making
- Knowledge Representation
- Learning

In this respect planning is a useful forcing ground for general techniques developed in other sub-fields of AI as well as being a generator of techniques itself. Planning also introduces its own problems:

- Representing and reasoning about time, causality and intentions
- Uncertainty in the execution of plans and dealing with the "real world"
- Sensation and perception of the "real world" and beliefs about it
- Multiple agents who may cooperate or interfere
- Physical or other constraints on suitable solutions

Planning also has some features in common with many other Expert Systems applications (as well as the general areas of search control and knowledge representation):

- Need to justify solutions
- Knowledge elicitation

Planners have been applied to a variety of application domains (as well as the simple exploratory domains such as block stacking). A selection is shown in Table 1

Table 1: Applications tackled by Knowledge-based Planning Systems

Domain	Planner
Robot Control	STRIPS [22]
Simple Program Generation	HACKER [71]
Mechanical Engineers Apprentice Supervision	NOAH [59]
House Building and Elect. Turbine Overhaul	NONLIN [74]
Experiment Planning in Molecular Genetics	MOLGEN [69]
Journey Planning	OPM [30]
Job Shop Scheduling for Turbine Production	ISIS-II [25]
Aircraft Carrier Mission Planing	SIPE, AIRPLAN [88][43]
Naval Logistics	NONLIN [77]
Voyager Spacecraft Mission Sequencing	DEVISER [78]
Planning conversations	KAMP [2]
etc.	

Plans have also been used as a basic data structure in other areas of AI such as:

- — Story understanding [62] [83]
- — Speech and text generation [48]
- — Office systems modelling [21]

Figure 1 shows many of the major planning systems developed in AI and how they relate to one another. No attempt has been made to add planners developed in the last year or two to the diagram since they often draw on techniques from many of the systems shown. It is convenient to survey the field on the basis of areas of interest rather than chronologically, since many themes recur. A bibliography is attached to this paper to act as an entry point to the literature.

2. SEARCH SPACE CONTROL

A planner is organised so that it defines a search space and then seeks a point in that search space which is defined as a solution. Planners differ as to 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 "state search space" can be traversed by applying any applicable operator to any state leading to a different point in the search space. A problem solution can be defined as the sequence of operators that can traverse the search space from some initial state to some state defined as meeting the goal criteria.

Other (most of the post-1975) planners define points in the search space as partially elaborated plans. One point in this space is transformed into another by applying any applicable planning operation 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, etc.. Given some initial (skeleton) plan defining a point in this "partial plan search space", a set of decisions has to be taken which lead to a fully detailed plan that meets the goal criteria and various constraints.

Many search control methods are aimed at reducing the initial search space defined by the planner. Other techniques are then geared towards choosing a path through the search space that leads as quickly as possible to a solution.

State Space Search

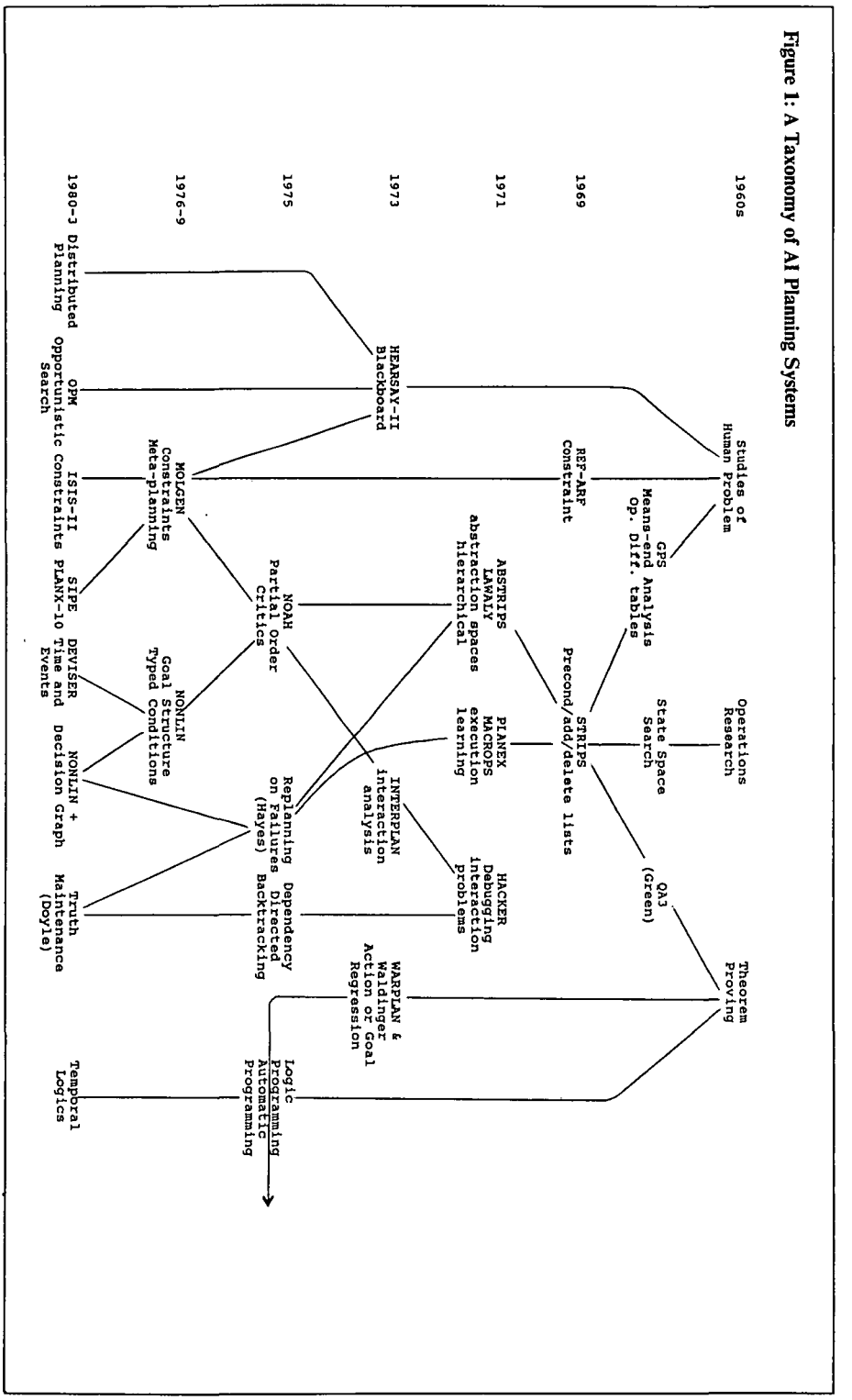
Early approaches to planning sought to apply legal "moves" to some initial state to search for a state that satisfied the given goals. There was a great deal of overlap with game playing work. Heuristic evaluation functions were employed to rate the various intermediate states produced to estimate their "closeness" to a goal state (e.g., in the Graph Traverser [12]).

State space search is often used to control the selection of alternatives in other search strategies, but is not often used for search in the actual problem space in general planners today.

Means-ends Analysis

To reduce the number of intermediate states considered, only those operators or activities that can satisfy some outstanding goal may be considered. These in turn can introduce new (hopefully simpler) sub-goals. This was introduced in GPS [51] and used in many later systems.

Figure 1: A Taxonomy of AI Planning Systems



Search Reduction through Least Commitment/No Alternatives Considered

Some systems, especially ones which introduced important new techiques, did not search through the possible alternatives at all (e.g., NOAH [(Sacerdoti, 1977]). Selections were made on the basis of the information available locally and then a commitment to that solution path was made. Of course, this often means that some problems could not be solved. However, since these systems were often demonstrated on particular applications for which a technique was most appropriate, the search was often successful at arriving at a solution directly.

Very often, techniques demonstrated in isolation in such a way were incorporated in later planners that could search alternatives and use the method alongside others depending on its relevance.

Depth-first backtracking

A simple method of considering alternative solution paths (especially when there are only a few to choose from due to the use of means-ends analysis, etc.) is to save the state of the solution at each point at which there are alternative ways forward and to keep a record of the alternative choices. The first is chosen and search continues. If there is any failure, the last choice point saved state is restored and the next alternative taken (if there are none, "backtracking" continues up one more level). Simple stack based implementation techniques can be used for this process (e.g., in STRIPS [22]).

Beam Search

A search method which considers all possible solutions so that they can be compared. Such a method is normally used when tight search space constraints are known and hence the solution space is expected to be small. It is also normally employed along with other heuristic search methods so that a heuristic choice can be of the "best" solution found amongst the solutions to some sub-problem which have been proposed by a beam search (e.g., in ISIS-II [25]).

One-then-Best Backtracking

Since there is often good local information available to indicate the preferred solution path, it is often appropriate to try the best choice indicated by local heuristic information before considering the many alternatives that may be available should the local choice prove faulty. Taken to the extreme, depth-first search gives something of the flavour 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 one of the local choices which seems most promising. This continues while all is going well (perhaps with some cut-off points to take a long, hard look at how well things are going). However, if a failure occurs, the *whole* set of alternatives which has been generated (and ranked by a heuristic evaluator) is reconsidered to select the re-focussing point for the search (e.g., in NONLIN [74]).

Dependency-Directed Search

It is well known that any backtracking system based on saved states and resumption points (whether depth-first or heuristically controlled) can waste much valuable search effort. There may be several unrelated parts to a solution. If it so happens that backtracking on one part has to go back beyond points at which work was done on an unrelated part, all effort on the unrelated part will be lost. Examples of work that has explored this important area includes Stallman and Sussman [67], NONLIN +Decision Graph [7], and MOLGEN [69] to some extent.

Some 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 of undoing any failure by propagating the undoing of all dependent parts of the solution. This leaves unrelated parts intact irrespective of whether they were worked on after some undone part of the solution.

Opportunistic Search

Some systems do not take a fixed (goal-driven or data-driven) directional 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 may be suggested by comparison of the current goals with the initial world model state, by consideration of the number of likely outcomes of making a selection, by the degree to which goals are instantiated, etc.. Any problem solving component may summarise its requirements for solution as constraints on possible solutions or restrictions of the values of variables representing objects being manipulated. They can then suspend their operation until further information becomes available on which a more definite choice can be made. (e.g., MOLGEN [68])

Many such systems operate with a "blackboard" through which the various components can

communicate via constraint information (e.g., HEARSAY-II [17] and OPM [30]). The scheduling of the various tasks associated with arriving at a solution may also be dealt with through (an area of) the blackboard.

Meta-level Planning

There are a number of planning systems which have an operator-like representation of the different types of operations that a planner can perform. A separate search is made to decide which of these is best applied at any point before decisions are taken about the details of the particular application plan being produced (e.g., MOLGEN [69] and Wilensky [84]). This technique is often used in opportunistic planners.

Distributed Planning

Some systems have gone further in distributing the sources of problem solving expertise or knowledge. They allow fully distributed planning with the sub-problems being passed between specialised planning experts. The use of the experts or the types of things they can do may be controlled through centralised blackboard and executive (with a system rather like priority scheduling of parallel processes) or may be controlled in a more distributed fashion via pairwise negotiation. Examples of relevant work include Smith's [65] Contract Net, Corkill [5], Kornfeld [37], Konolige and Nilsson [36], Georgeff [26], Corkill and Lesser [6].

Work is underway to interleave the time spent planning, sensing the state of the world to gather information, and execution (e.g., Doran and Trayner [13], Drummond [15]).

Pruning of the Search Space

Besides the basic methods of reducing the search space by selection of relevant operators in means-ends analysis, many other methods of reducing the search space size have been employed in planners. Some are identified here:-

- by considering "higher priority" goals first in hierarchical planners (e.g., ABSTRIPS [57] and LAWALY [64], See below.
- by detecting and correcting for interactions in an intelligent fashion (e.g., Waldinger [88] with Goal Regression and in INTERPLAN [72], NOAH [59] and NONLIN [74]). See section 4 below.
- by using applicability conditions and other information restricting ways that goals can be satisfied (e.g., "Typed Preconditions" in NONLIN [73], DEVISER [78]).
- by rejection of states or plans that are known to be impossible or in violation of some rule about the state or plan (e.g., WARPLAN [81] "IMPOSS" statements and Allen and Koomen's [1] "Domain Constraints").
- by checks on resource usage levels, time constraints on actions, etc. (e.g., NONLIN [77], DEVISER [78] and SIPE [88]).

3. HIERARCHY/ABSTRACTION LEVELS

The order in which conjunctive goals are tackled can have a marked effect on the efficiency of the search process for a solution. In some linear planners, it can make the differences between finding a solution and looping round on the same goals repreatedly or getting solutions with redundant steps. Some approaches to ordering the various goals involve separating the goals into levels of importance or priority (the more abstract and general goals being worked on first and the more concrete or detailed levels being filled in later).

Strict Search by Levels

The early hierarchical systems (e.g., ABSTRIPS [57], LAWALY [64], NOAH [59]) formed a solution at the most abstract level and then made a commitment to this solution. The lower levels were then planned using the pre-set skeleton plan formed at the upper levels. No backtracking to the higher levels was possible.

Non-strict by Levels

Later systems (e.g., NONLIN [74]) treat the abstraction levels as a guide to a skeleton solution, but are able to re-plan or consider alternatives at any level if a solution cannot be found, or if problems with part of a solution indicate that a higher level choice was faulty.

Opportunistic by Levels

Other hierarchical systems use the abstraction levels as one guide to the ordering of goals, but have other mechanisms that can be considered alongside this. 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 (e.g., MOLGEN [70]).

4. GOAL ORDERING AND INTERACTION DETECTION AND CORRECTION

Planners can be split into two basic types with respect to the way that they tackle conjunctive goals. The "linear" planners make the "linearity assumption" — that solving one goal and then following it by the solution to a subsequent goal is often successful because the solutions to different goals are often decoupled. The "non-linear" planners take a "least-commitment" approach by representing a plan as a partially ordered network (a graph) of actions and goals and only introducing ordering links between actions or goals when the solutions demand this.

In both types of systems, the solutions to one goal may interact with the solutions to the others. Planners can be categorised by their ability to detect and in some cases correct for the interactions.

Linear — Interactions Ignored

If the separate sub-goals are solved sequentially and then a simple check made to see if the conjunction of goals then holds (e.g., in STRIPS [22]), this can lead to mutual goal or subgoal interference which at best can lead to redundant actions in the plan and at worst may get the planner into an endless cycle of re-introducing and trying to satisfy the same goal over and over again.

Linear — Interactions Detected

In early systems that recognised the importance of detecting "bugs" with the "linearity assumption" (e.g., HACKER [71]), corrections were attempted by backtracking to points where alternative goal orderings could be tried (as part of the overall search space). This could solve some types of problems.

Linear — Interactions Detected — Corrected by Action Regression

One method of correcting for detected interactions is to allow the interfering action (just being introduced to satisfy some goal) to be placed at different (earlier) points in the partial linear plan until a position is found where there is no interaction (e.g., WARPLAN [81]). Unfortunately this can lead to the inclusion of redundant actions since the action may not be necessary (or may not be the best choice) when moved earlier in the plan.

Linear — Interactions Detected — Corrected by Goal Regression

The problem of regressing the actual action chosen to satisfy a goal back through the plan to correct for an interaction can be solved by regressing the goal itself when an interaction with some particular solution fails (e.g., Waldinger [80]). This has the advantage that redundant actions are not included if the goal is already solved at some earlier point in the plan.

Linear — Interactions Detected — Corrected by Analysis

A technique called "Goal Structure" was introduced in INTERPLAN [72] to record the link between an effect of one action that was a precondition (subgoal) of a later one. This representation is additional to the ordering links between the actions themselves (as some actions have effects that are used much later in the plan). Sussman [71] referred to this as the "teleology" of the plan.

Interactions are detected as an interference between some new action or goal being introduced and one or more Goal Structure ranges (originally called "Holding Periods" in INTERPLAN). A table called a "ticklist" was built to represent the relationships between actions and the goals they achieved for later points in the plan. A minimum set of goal reorderings (called the "Approach") could be suggested that corrected for the interactions found. This considered fewer reorderings of the given goals and subgoals than the regression methods.

Non-linear — Interactions Detected — Limited Correction

The first non-linear planner, NOAH [59], incorporated procedures called "critics" which were used to search for interactions between parts of the plan. It used a Table of Multiple Effects (TOME) to aid in discovering the interactions. The TOME was modelled on the concept of Goal Structure as it appeared in INTERPLAN. Once detected, NOAH could correct for interactions in a limited way suited to the applications it was employed on. This was primarily due to the limitation in NOAH that no alternatives could be considered — the best choice was committed to at each stage.

Non-linear — Interactions Detected — Corrected by Analysis

The detection of interactions by analysis of the underlying "Goal Structure" was added to a non-linear planner, called NONLIN, modelled on NOAH. The combination in NONLIN [74] was more effective than its earlier use in the linear INTERPLAN system. In NONLIN, the minimum set of orderings necessary to resolve any interaction could be suggested by the introduction of ordering links into a partially ordered plan only when this became essential. The NONLIN system could consider alternatives if failures on any chosen search branch occurred.

Similar analyses of a representation of the effect and condition structure of a plan to detect and correct for interactions have been included in PLANX-10 [4], SIPE's "Plan Rationals" [88] and Dean's TNMS [10].

Non-linear — Interactions Detected and Corrected — Time Limits Handled

When individual goals or actions in a plan have time constraints on when they can be satisfied or performed, the detection and correction of interactions must be sensitive to the temporal displacement introduced (e.g., in DEVISER [78]). This normally further limits the number of legal solutions that can be proposed. There may also be a need to link the plan to externally events (beyond the control of the planner and possibly occurring at pre-specified times). The DEVISER system added such capabilities to a NONLIN-like planner.

Non-linear — Interactions Detected and Corrected — Resoures Handled

The treatment of objects being manipulated as scarce resources on which usage conflicts occur and need to be avoided was incorporated in the MOLGEN [69] and SIPE [88] planners.

5. 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 orientated towards the generation of plans which are fully or partially ordered sequences of primitive actions. However, there has been some effort on the generation of plans which contain conditionals ("if ... then ... else ...") and iterators ("repeat ... until ...").

Black Box Approach to Conditionals and Iterators

A conditional or iterator which can be modelled as a single entity and for which the differences between the multiple paths or the individual loops are not important can be handled by many of the planning systems and techniques already described. NOAH [59] explicitly dealt with such packaged conditionals and iterators. SIPE [88] was able to deal with iteration which repeated a planned action sequence on each member of a set of parameters (which could be objects or points on an aircraft flight trajectory).

Conditionals: Branch and Case Analysis

Conditionals were handled in WARPLAN-C (Warren [82]) 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 lead to a case analysis of the separate branches to produce a plan that was tree-structured.

Conditionals: Branch and Re-Join

There has been relatively little work in the mainstream AI planning literature which has attempted a full treatment of conditionals which can branch and re-join later in the plan and for which the black box approach was not sufficient. However, work on automatic programming and theorem proving has comsidered this area. See, for example, Luckham and Buchanan [42] and Dershowitz [11]. The latter reference has an extensive bibliography of papers on automatic programming and shown how AI planning research overlaps with this work.

Formal Models of Plans and Actions

Plan representations and formalisms which are more powerful than the orderings-on-actions basis of many AI planners are now being explored. For example, petri-nets (Drummond [15]), formal models of processes, actions and states (see section 7 of this paper), etc.. Some of these allow for the representation of and reasoning about iterative and conditional statements in plans.

6. TIME AND RESOURCE HANDLING

It is becoming increasingly important in planning systems to perform in realistic domains where resources of various types are limited. Also, planners are being used in domains where time considerations and matters beyond the control of the planning system itself must be accounted for. Several systems have explored this area.

Compute Time and Resource Usage

In a project planning domain, NONLIN [73] maintained information about the durations of the various activities and used this to compute earliest and latest start times for each action. A critical path of actions could be found. A "cost" measure could also be kept with each activity. This information was used in an extension to the NONLIN planner that could selectively remove costly activities or lengthy activities from a plan and replace them with others that either reduced the cost or the duration depending on the limitation that was exceeded [7]

More recent work on NONLIN (Tate and Whiter [77]) has added the capability of representing multiple limited resources and making selection from appropriate activities on the basis of reducing some overall computed "preference" between them.

Objects as Resources

SIPE [88] is able to reason about competing demands for the use of scarce objects (such as a ladder or a key). An analysis of the objects used and returned by actions, or objects "consumed" by them, can be used to linearize the plan to prevent usage conflicts.

Event and Time Specifications for Actions

DEVISER [78] allows a "time window" to be specified for any goal or action. External events can be described as having some effect at a particular time. The planner propagates the temporal links between these time windows, progressively narrowing them as they become constrained by other actions. It can detect when some plan step prevents a goal from being achieved or an activity from being executed at the required time. In such a case, backtracking will consider alternative solutions.

AIRPLAN (Masui, McDermott and Sobel [43]) was able to reason about time intervals and how actions suggested in them could interfere, etc..

O-Plan (Open Planner Architecture) (Bell and Tate [31]) the successor to NONLIN, uniformly represents time constraints (and resource usage) by a numeric (min, max) pair which bound the actual values for activity duration, activity start and finish times, and delays between activities. The actual values may 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, uncertainty in modelling the domain, etc.. Constraints can be stated on the time (and resource) values which may lead to the planner finding that some plans are invalid.

Flexible Time Handling

Very much more flexible handling of the propagation of temporal constaints between the steps of a plan is being considered in the widespread research effort on temporal logic (e.g., McDermott [46], Allen and Koomen [1], etc.).

Soft Constraints

ISIS-II [25] allows a wide variety of constraints on the problem to be specified. Some of these can be in the form of preferences or "soft" constraints which are used to guide the search for acceptable solutions.

7. DOMAIN REPRESENTATION

Many AI systems have introduced formalisms or constructs for capturing the information about the application domain and for making knowledge available to a planning system. Once again, later systems have often built on the best aspects of earlier efforts.

Differences

The early means-ends analysis systems selected appropriate operators to apply to a problem by considering the "differences" between the goal state and the initial state. The General Problem solver (GPS [51]) associated the operators for the problem with the "differences" they could reduce.

Add/Delete/Precondition Lists

Using ideas from problem solving in a "situational calculus" (McCarthy and Hayes [44], Green's QA3 [28]) and the notion of differences from GPS, STRIPS [22] took the assumption that the initial world model was only changed by a set of additions to and deletions from the statements modelling the world state (everything else remaining unchanged). This is called the "STRIPS assumption". STRIPS then defined an operator as having an add-list, a delete-list and a preconditions-list (to state the applicability or sub-goaling conditions). Operators could be selected on the basis of goals which occur on their add-lists (statements added to the world model).

Abstraction Levels on Goals

The hierarchical systems introduced the ability to specify the abstraction or criticality levels of the various statements that could be made about a world state (e.g., ABSTRIPS [57], LAWALY [64]).

Partial-orders on plan parts in hierarchical planners

The first non-linear planner, NOAH [59], allowed plan steps to be kept in parallel until some cause was found to linearise them. NOAH had 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.

Declarative partial-order representation

Later planners introduced declarative representations for operators in extensions of the STRIPS operator type of formalism (e.g., NONLIN's Task Formalism [74] and SIPE Notation [88]). As well as add, delete and precondition lists, an "expansion" of the operator to a lower level of detail could be specified as a partial order on suitable sub-actions and sub-goals.

Intent of plan steps

Some systems distinguish between ordering relationships on actions and the purpose of each action or goal with respect to the other parts of the plan at which they are required. Such information was first made available through the STRIPS "triangle tables" [23] to aid in re-use of parts of plans in new situations. In HACKER [71], the information is kept as protection intervals (Sussman used the term "teleology") and was used to detect "protection violations" which led HACKER to consider alternative ways to perform a task being tried. In INTERPLAN [72] and NONLIN [74] [76] such information (called "Goal Structure") was used to be more precise about interaction detection and aided in suggesting the corrections (for this, "typed conditions" could be stated in an operator description). PLANX-10 [4] has internal structures similar to Goal Structure. SIPE's "Plan Rationale" [88] is a more limited version that reflects only the main outcomes of given actions.

Typed Preconditions

There is often a distinction between applicability conditions for an operator and sub-goals to be introduced to enable the operator to achieve its purpose. AI languages used for planning have recognised this distinction by providing two types of goal statement (e.g., in POPLER [94]). NONLIN [74] and DEVISER [78] allow this same distinction between goals which are simply to be checked if they already hold and goals that can cause further introduction of operators and sub-goals.

NONLIN introduced other condition types, as well as those mentioned above, to reflect internally managed goals and actions within the control of a particular operator's "manager" ("supervised") and those beyond the manager's control ("unsupervised"). These different types of condition are useful in differentiating between the various methods by which goals can be satisfied and enabling the system to recover from planning failures in more appropriate ways.

Resources

The definition of shared objects as resources and the declaration of the use of such resources in operators was provided in SIPE. The declaration of multiple consumable resoures, their limitations and preferences as to which should be minimised in solutions has been added to NONLIN. DEVISER can also handle consumable resources (such as fuel).

Time Windows and Events (e.g., DEVISER [78])

DEVISER has provided a method for specifying a time window on goals and activities. External events and their time of occurrence can also be given. Delayed events (caused some time after a planned action) can be specified.

Property inheritance

Some planning systems make use of extensive knowledge representation schemes that can express the type of objects and the properties that instances of the types can inherit (e.g., SIPE [88] and PLANX-10 [4]). This can significantly increase the size of problems that can be tackled.

Action Logics and Formal Models

Formal models of actions, processes and states have been a recurring theme of AI work and this has impacted on AI planning work. Some of the work has its roots in process models and Finite State Automata. The majority of the AI planning techniques reviewed in this paper are most suited to planning for discrete activities. Some of the work on formal models and action logics admit reasoning about continuous processes. Such topics require a survey in their own right and are only briefly touched upon here. Example references include Moore [49], Lansky [38]. Recent research in AI planning is taking more account of such work.

8. SUPPORT LANGUAGES AND DATA BASE SYSTEMS

Some early planning systems used specially tailored languages designed to allow search and pattern directed inference (e.g., PLANNER [100], POPLER [94], CONNIVER [112] and their derivatives). The depth-first search strategy of the logic programming language PROLOG [93] was used in WARPLAN [81], probably the first planner to be implemented in that language. In other cases a base AI Programming language was augmented by suitable language extensions (e.g., QA4 [108], QLISP [109]).

However most systems have been implemented in a combination of an AI language with a knowledge representation language such as UNITS [111] in LISP (for MOLGEN [69], HBASE [89] in POP-2/POPLOG [110]) (for INTERPLAN [72] and NONLIN [74]), etc. Separate research and development on knowledge representation languages (e.g., KRL [91], LOOPS, parallel reduction machines and large content addressable memories (e.g., NETL [97], FACT [106]) is an important source of support for new planning work.

9. PLANNING BIBLIOGRAPHY

Where names of planning systems and other AI software are described in the paper, the bibliography is annotated with the name of the system in brackets.

[1] Allen, J.F. and Koomen, J.A. (1983) "Planning Using a Temporal World Model", IJCAI-83, pp 741-747, Karlsruhe, West Germany. (TIMELOGIC)

[2] Appelt, D.E. (1985) "Planning English Referring Expressions", Artificial Intelligence, 26, pp 1-33. (KAMP)

[3] Bell, C.E. and Tate, A. (1985) "Using Temporal Constraints to Restrict Search in a Planner" Proceedings of the Third Workshop of the Alvey IKBS Programme Planning Special Interest Group, Sunningdale, Oxfordshire, UK, April 1985. Available through the Institute of Electrical Engineers, London, UK (O-PLAN)

[4] Bresina, J.L. (1981) "An Interactive Planner that Creates a Structured Annotated Trace of its Operation", Rutgers University, Computer Science Research Laboratory, Report CBM-TR-123. (PLANX-10)

[5] Corkill, D.D. (1979) "Hierarchical Planning in a Distributed Environment", IJCAI-79, pp 168-756, Tokyo, Japan.

[6] Corkill, D.D. and Lesser, V.R. (1983) "The Use of Meta-level Control for Coordination in a Distributed Problem Solving Network", IJCAI-83, pp 748-756, Karlsruhe, West Germany.

[7] Daniel, L. (1983). "Planning and Operations Research", in "Artificial Intelligence: Tools, Techniques and Applications", Harper and Row, New York. (NONLIN)

[8] Davis, R. and Smith, R. (1983) "Negotiation as a Metaphor for distributed problem solving", Artificial Intelligence, 20, pp 63-109.

[9] Davis, P.R. and Chien, R.T. (1977) "Using and Re-using Partial Plans", IJCAI-77, Cambridge, Mass., USA.

[10] Dean, T. (1985) "Temporal Reasoning Involving Counterfactuals and Disjunctions", IJCAI-85, Los Angeles, Calif. (TNMS)

[11] Dershowitz, N. "Synthetic programming", Artificial Intelligence, 25, pp 323-373.

[12] Doran, J.E. and Michie, D. (1966) "Experiments with the Graph Traverser Program" Proceedings of the Royal Society, A, pp 235-259. (GRAPH TRAVERSER)

[13] Doran, J.E. and Trayner, C. (1985) "Distributed planning and Execution — Teamwork 1", Computer Science Technical Report, University of Essex, UK (TEAMWORK)

[14] Doyle, J. (1979) "A Truth maintenance System", Artificial Intelligence, 12, pp 231-272.

[15] Drummond, M.E. (1985) "Refining and Extending the Procedural Net" IJCAI-85, Los Angeles, Calif.

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

[17] 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, No. 2.

[18] Fahlman, S.E. (1974) "A Planning System for Robot Construction Tasks" Artificial Intelligence, 5, pp 1-49.

[19] Faletti, J. (1982) "PANDORA — A Program for Doing Commonsense Reasoning Planning in Complex Situations", AAAI-82, Pittsburgh, Pa., USA, Aug, 1982. (PANDORA)

[20] Fikes, R.E. (1970) "REF-ARF: A System for solving problems stated as Procedures", Artificial Intelligence", 1, pp 27-120.

[21] Fikes, R.E. (1982) "A Commitment-based Framework for Describing Informal Cooperative Work", Cognitive Science, 6, pp 331-347.

[22] Fikes, R.E. and Nilsson, N.J. (1971) "STRIPS: a New Approach to the Application of Theorem Proving to Problem Solving", Artificial Intelligence, 2, pp 189-208. (STRIPS)

[23] Fikes, R.E., Hart, P.E. and Nilsson, N.J. (1972a) "Learning and Executing Generalised Robot Plans", Artificial Intelligence, 3. (STRIPS/PLANEX)

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

[25] Fox, M.S., Allen, B. and Strohm, G. (1981) "Job Shop Scheduling: an Investigation in Constraint-based Reasoning", IJCAI-81, Vancouver, British Columbia, Canada, August 1981. (ISIS.II)

[26] Georgeff, M. (1982) "Communication and Interaction in Multiagent Planning Systems", AAAI-3.
[27] Georgeff, M. and Lansky, A. (1985) "A Procedural Logic", IJCAI-85, Los Angeles, Calif., Aug 1985.
[28] Green, C.C. (1969) "Theorem Proving by Resolution as basis for Question Answering" in Machine Intelligence, 4, eds. Meltzer, B. and Michie, D., Edinburgh University Press.
[29] Hayes, P.J. (1975) "A Representation for Robot Plans", Advance papers of IJCAI-75, Tbilisi, USSR.
[30] Hayes-Roth, B. and Hayes-Roth, F. (1979) "A Cognitive Model of Planning", Cognitive Science, pp 275-310. (OPM)
[31] Hayes-Roth, B. (1983a) "The Blackboard Architecture: A General Framework for Problem Solving?", Heuristic Programming Project Report No. HPP-83-30. Stanford University. May 1983.
[32] Hayes-Roth, B. (1983b) "A Blackboard Model of Control", Heuristic Programming Project Report No. HPP-83-38. Stanford University. June 1983. (OPM)
[33] Hendrix, G. (1973) "Modelling Simultaneous Actions and Continuous Processes", Artificial Intelligence, 4, pp 145-180.
[34] Kahn, K. and Gorry, G.A. (1977) "Mechanizing Temporal Knowledge" Artificial Intelligence, 9, pp 87-108.
[35] Konolige, K. (1983) "A Deductive Model of belief", IJCAI-83, pp 377-381, Karlsruhe, West Germany, Aug 1983.
[36] Konolige, K. and Nilsson, N.J. (1980) "Multi-agent Planning Systems", AAAI-1, pp 38-142, Stanford, Ca., USA.
[37] Kornfeld, W.A. (1979) "ETHER: a Parallel Problem Solving System" IJCAI-79 pp 490-492, Tokyo, Japan.
[38] Lansky, A. (1985) "Behavioral Planning for Multi-Agent Plans", SRI AI Center Report.
[39] Latombe, J-C. (1976) "Artificial Intelligence in Computer-aided Design — The TROPIC System", Stanford Research Institute AI Center Technical Note 125, Menlo Park, Ca., USA.
[40] Lermat, D.B. (1975) "BEINGS: Knowledge as Interacting Experts", IJCAI-75, pp 126-133, Tbilisi, USSR.
[41] London, P. (1977) "A Dependency-based Modelling Mechanism for problem solving", Dept of Computer Science, University of Maryland, Memo. TR-589.
[42] Luckham, D.C. and Buchanan, J.R. (1974) "Automatic Generation of Programs Containing Conditional Statements", AISB Summer Conference, University of Sussex, UK, pp 102-126, July 1974.
[43] Masui, S., McDermott, J. and Sobel, A. (1983) "Decision-making in Time Critical Situations", IJCAI-83, pp 233-235, Karlsruhe, West Germany, Aug, 1983. (AIRPLAN)
[44] McCarthy, J. and Hayes, P.J. (1969) "Some Philosophical Problems from the standpoint of Artificial Intelligence", in Machine Intelligence 4, ed. Meltzer, B. and Michie, D., Edinburgh University Press.
[45] McDermott, D.V. (1978) "Planning and Acting", Cognitive Science, 2.
[46] McDermott, D.V. (1982) "A Temporal Logic for Reasoning about Processes and plans", Cognitive Science, 6, pp 101-155.
[47] McDermott, D.V. and Doyle, J. (1979) "An Introduction to Nonmonotonic Logic", IJCAI-79 pp 562-567, Tokyo, Japan.
[48] Mellish, C.S. (1984) "Towards Top-down Generation of Multi-paragraph Text", Proceedings of the Sixth European Conference on Artificial Intelligence, pp 229, Pisa, Italy, September 1984.
[49] Moore, R. (1980) "Reasoning about Knowledge and Action", SRI AI Center Report No. 191.
[50] Mostow, D.J. (1983) "A Problem Solver for Making Advice Operational", Proc. AAAI 3, pp 179-283
[51] Newell, A. and Simon, H.A. (1963) "GPS: a Program that Simulates Human Thought", in Feigenbaum, E.A. and Feldman, J. eds Comuters and Thought (McGraw-Hill, New York, 1963). (GPS)
[52] Reiger, C. and London, P. (1977) "Subgoal Protection and Unravelling During Plan Synthesis", IJCAI-77, Cambridge, Mass., USA.
[53] Rich, C. (1981) "A Formal Representation for Plans in the Programmer's Apprentice", IJCAI-81 pp 1044-1052, Vancouver, British Columbia, Canada.
[54] Rich, C. Shrobe, H.E. and Waters, R.C. (1979) "Overview of the Programmer's Apprentice", IJCAI-79, pp 827-828, Tokyo, Japan.
[55] Rosenschein, S.J. (1980) "Synchronisation of Multi-agent Plans", AAAI-2.
[56] Rosenschein, S.J. (1981) "Plan Synthesis; A Logical Perspective", IJCAI-81, Vancouver, British Columbia, Canada.
[57] Sacerdoti, E.D. (1973) "Planning in a Hierarchy of Abstraction Spaces", Advance papers of IJCAI-73, Palo Alto, Ca., USA. (ABSTRIPS)
[58] Sacerdoti, E.D. (1975) "The Non-linear Nature of Plans", Advance papers of IJCAI-75, Tbilisi, USSR. (NOAH)

[59] Sacerdoti, E.D. (1977) "A Structure for plans and Behaviour", Elsevier-North Holland. (NOAH)
[60] Sacerdoti, E.D. (1979) "Problem solving Tactics", IJCAI-79, Tokyo, Japan.
[61] 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, July, 1985. (CALLISTO)
[62] Schank, R.C. and Abelson, R.P. (1977) "Scripts, Plans, Goals and Understanding", Lawrence Erlbaum Press, Hillsdale, New Jersey, USA.
[63] Siklossy, L. and Roach, J. (1973) "Proving the Impossible is Impossible is Possible: Disproofs based on Hereditary Partitions", IJCAI-73, Palo Alto, Calif. (DISPROVER/LAWALY)
[64] Siklossy, L. and Dreussi, J. (1975) "An Efficient Robot Planner that Generates its own Procedures", IJCAI-73 Palo Alto, Ca., USA. (LAWALY)
[65] Smith, R.G. (1977) "The Contract Net: a Formalism for the Control of Distributed Problem Solving", IJCAI-77 pp 472 Cambridge, Mass, USA.
[66] Smith, R.G. (1979) "A Framework for Distributed Problem Solving", IJCAI-79, Tokyo, Japan.
[67] Stallman, R.M. and Sussman, G.J. (1977) "Forward Reasoning and Dependency Directed Backtracking", Artificial Intelligence, 9, pp 135-196.
[68] Steele, G.L. and Sussman, G.J. (1978) "Constraints", MIT AI Lab Memo 502.
[69] Stefik, M.J. (1981a) "Planning with Constraints", Artificial Intelligence, 16, pp 111-140. (MOLGEN)
[70] Stefik, M.J. (1981b) "Planning and Meta-planning", Artificial Intelligence, 16, pp 141-169. (MOLGEN)
[71] Sussman, G.A. (1973) "A Computational Model of Skill Acquisition", M.I.T. AI Lab. memo no. AI-TR-297. (HACKER)
[72] Tate, A. (1975) "Interacting Goals and Their Use", IJCAI-75, pp 215-218, Tbilisi, USSR. (INTERPLAN)
[73] Tate, A. (1976) "Project Planning Using a Hierarchical Non-linear Planner", Dept. of Artificial Intelligence Report 25, Edinburgh University. (NONLIN)
[74] Tate, A. (1977) "Generating Project Networks", IJCAI-77, Boston, Ma., USA. (NONLIN)
[75] Tate, A. (1984a) "Planning and Condition Monitoring in a FMS", International Conference on Flexible Automation Systems", Institute of Electrical Engineers, London, UK July 1984. (NONLIN)
[76] Tate, A. (1984b) "Goal Stucture: Capturing the Intent of Plans", European Conference on Artificial Intelligence, Pisa, Italy, September 1984. (NONLIN)
[77] 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. December 1984. (NONLIN)
[78] Vere, S. (1983) "Planning in Time: Windows and Durations for Activities and Goals", IEEE Trans. on Pattern Analysis and Machine Intelligence, PAMI-5, No. 3, pp. 246-267, May 1983. (DEVISER)
[79] Vilain, M.b. (1980) "A System for Reasoning about Time", AAAI-2.
[80] Waldinger, R. (1975) "Achieving Several Goals Simultaneously", SRI AI Center Technical Note 107, SRI, Menlo Park, Ca., USA.
[81] Warren, D.H.D. (1974) "WARPLAN: a System for Generating Plans", Dept. of Computational Logic Memo 76. Artificial Intelligence, Edinburgh University. (WARPLAN)
[82] Warren, D.H.D. (1976) "Generating conditional plans and programs", Proceedings of the AISB Summer Conference, pp 344-354, University of Edinburgh, UK, July 1976. (WARPLAN-C)
[83] Wilensky, R. (1978) "Understanding Goal-based Stories", Dept. of Computer Science, Yale Unviersity, Research Report No. 140.
[84] Wilensky, R. (1981a) "Meta-planning: Representing and Using Knowledge about planning in Problem Solving and Natural Language understanding", Cognitive Science, 5, pp 197-233.
[85] Wilensky, R. (1981b) "A Model for Planning in Complex Situations", Electronics Research Lab. Memo. No. UCB/ERL M81/49, University of California, Berkeley, Ca., USA.
[86] Wilensky, R. (1983) "Planning and Understanding", Addison-Wesley, Reading, Mass.
[87] Wilkins, D.E. and Robinson, A.E. (1981) "An Interactive Planning System", SRI Technical Note 245. (SIPE)
[88] Wilkins, D.E (1983) "Representation in a Domain-Independent Planner", IJCAI-83, pp 733-740, Karlsruhe, West Germany. (SIPE)

10. BIBLIOGRAPHY ON SUPPORT LANGUAGE, DATABASES AND BACKGROUND

[89] Barrow, H.G. (1975) "HBASE: a Fast Clean Efficient Data Dase System", D.A.I. POP-2 library documentatin. Edinburgh University., (HBASE)
[90] Bobrow, D.G. and Stefik, M.J. (1983) "The LOOPS Reference Manual", Xerox Palo Alto Research Center, Ca. USA. (LOOPS)
[91] Bobrow, D.G. and Winograd, T. (1976) "An Overview of KRL — a Knowledge Representation

Language", Xerox PARC Repport CSL-76-4. Xerox Palo Alto Research Center, Ca. USA. (KEL)
[92] Carnegie Group Inc. "Knowledge Craft", Commerce Court at station square, Pittsburgh, PA 15219, USA. (Knowledge Craft, SRL+)
[93] Clocksin, W. and Mellish, C. (1981) "Programming in PROLOG", Springer-Verlag. (PROLOG)
[94] Davies, D.J.M. (1973) "POPLER 1.5 Reference Manual", D.A.I. Theoretical Psychology Unit Report no.1, Edinburgh university. (POPLER)
[95] Deering, M., Faletti, J. and Wilensky, R. (1981) "PEARL: An Efficient Language for Artificial Intelligence Programming", IJCAI-81, Vancouver, British columbia, Canada, Aug, 1981. (PEARL)
[96] Elcock, E.W., Foster, J.M., Gray, P.M.D., McGregor, J.J, and Murrry A.M. (1971) "ABSET, a programming Language based on sets: Motivation and Examples", in Meltzer, B. and Michie, D. Machine Intelligence, 6, (Edinburgh University Press). (ABSET)
[97] Fahlman, S.E. (1979) "NETL: a System for Representing Real World Knowledge", MIT Press, Cambridge, Mass. USA. (NETL)
[98] Fox, M. (1983) "SRL user's Manual", Technical Report, Robotics Institute, Carnegie-Mellon University, Pittsburgh, Pa., USA. (SRL)
[99] Hendrix, G. (1975) "Expanding the Utility of Semantic Networks through Partitioning", IJCAI-75, Tbilisi, USSR.
[100] Hewitt, C. (1972) "Description and Theoretical Analysis (using Schemata) of PLANNER", MIT AI Lab. Memo no. MAC-TR-256. (PLANNER)
[101] Hillis (1981) "The Connection Machine" MIT AI Lab Memo 640.
[102] Inference Corporation (1985), "ART Manual", 5300 West Century Blvd, 5th Floor, Los Angeles, CA 90045. (ART)
[103] Intellicorp (1985), "KEE System Manual" 707 Laurel Street, Menlo Park, CA 94025-3445. (KEE)
[104] Jiang, Y.J. and Lavington, S.H. (1985) "The Qualified Binary Relationship Model of Information", University of Manchester, Department of Computer Science, Internal Report IFS/2/85.
[105] McDermott, D.V. and Sussman, G.J. (1972) "The CONNIVER Reference Manual", M.I.T. AI Lab. Memo no. 259. (CONNIVER)
[106] McGregor, D.R. and Malone, J.R. (1981). "The FACT Database: A System using Generic Associative Networks", Research Report No. 2/80, Department of Computer Science, Univ. of Strathclyde, UK. (FACT)
[107] Nii, H.P. and Aiello, N. (1979) "AGE (Attempt to Generalize): A Knowledge-based Program for Building Knowledge-based programs", IJCAI-79, Tokyo, Japan. (AGE)
[108] Rulifson, J.F., Derkson, J.A. and Waldinger, R.J. (1972) "QA4: A Procedural Calculus for Intuitive Reasoing", Technical Note 73, SRI International Menlo Part, Ca., USA, (QA4)
[109] Sacerdoti, E.D., Fikes, R.E., Reboh, R., Saglowicz, D. and Waldinger, R.J. (1976) "QLISP: a Language for the Interactive Development of Complex Systems", SRI International, AI Center, (QLISP) Stanford, Ca. USA.
[110] Sloman, A. (1983) (POPLOG — a Multi-purpose, Multi-language Program Development Environment" Cognitive Studies Programme, University of Sussex, UK. (POPLOG)
[111] Stefik, M.J. (1979) "an Examination of a Frame-structured Representation system", IJCAI-79, pp 845-852 Tokyo, Japan, (UNITS)
[112] Sussman, G.A. and McDermott, D.V. (1972) "Why conniving is Better than Planning", MIT AI Lab. Memo 255A. (CONNIVER)

11. RECOMMENDED READING

A good and reasonably up-to-date account of AI planning techniques and systems is given in Charniak and McDermott's Introduction to Artificial Intelligence textbook (114). In particular, chapter 9 and sections of chapters 5 and 7 are relevant.

Somewhat earlier material is provided in Elaine Rich's Artificial Intelligence textbook (116). Nilsson's book on the Principles of Artificial Intelligence (115) provides a uniform treatment of planning techniques available up to the time it was published. There are several useful summaries of early AI planning work in the Handbook of Artificial Intelligence (113) Volume I section II.D and Volume III sections XI.B, XI.Cf and XV.

[113] Barr, A. and Feigenbaum E.A. (1981) "The Handbook of Artificial Intelligence" William Kaufmann, Los Angeles, Calif.
[114] Charniak, E. and McDermott, D.V. (1985) "Introduction to Artificial Intelligence", Addison-Wesley.
[115] Nisson, N.J. (1980) "Principles of Artificial Intelligence", Tioga Press, Palo Alto, Calif.
[116] Rich, E. (1983) "Artificial Intelligence", McGraw-Hill, New York.

ACKNOWLEDGEMENTS

Thanks go to my colleagues in the Artificial Intelligence Applications Institute and the Department of Artificial Intelligence at Edinburgh University for their comments on drafts of this paper. Mark Drummond was especially helpful in updating an earlier draft to the present paper.

The support of my colleagues at Systems Designers Scientific for the work of the AIAI Knowledge-Based Planning Group is gratefully acknowledged.

This article is derived from an earlier invited paper for the U.K. Alvey IKBS Expert Systems Research Theme Workshop at Abingdon, Oxfordshire, U.K. in April 1984.