NONLINEAR PLANNING: A RIGOROUS RECONSTRUCTION

David Chapman

MIT Artificial Intelligence Laboratory
545 Technology Square Room 709
Cambridge. MA 02139
Net ZVONAfuMIT-MC.ARPA

The problem of achieving several goals simultaneously lias been central to domain-independent planning research, the non-linear constraint-posting approach has been most successful Previous planners of this type ". 4, 5. f> have been complicated, heuristic, and ill-defined. 1 have combined and distilled the state of the art into a simple, precise, implemented algorithm (TWEAK) which I have proved correct and complete The simplicity and rigor of this algorithm illuminate the workings of previous planners, the range of applicability of current planning technology, and suggest future directions for research. This paper presents the mathematical foundations for non-linear planning; due to space limitations, I have omitted proofs, some detail, and much discussion These appear in 1

This paper begins by presenting a series of necessarily dry and obvious definitions, leading up to that of a plan that solves a problem 1 present a "truth criterion which provides an efficient means ol analyzing a plan to determine when a proposition will be true in the world as the plan is executed. The truth criterion also provides a wa\ of making a plan achieve a goal, and this is the basis of the TWEAK algorithm Finally 1 state a completeness correctness theorem and present conclusions.

TWEAK is a constraint posting planner Constraint posting is the definition of an object, a plan in this case, by successively specifying more and more partial descriptions it must fit Alternatively, constraint posting can be viewed as a search strategy in which rather than generating and testing specific alternatives, chunks of the search space are progressively removed from consideration by const raints that rule them out, until finally every remaining alternative is satisfactory. The advantage of the const ramt posting approach is that properties of the object being searcher! for do not have to be chosen until a reasoned decision can be made This reduction of arbitrary choice oft.en reduces the amount of backtracking necessarv.

This report describes research done at the Artificial Intelligence Laboratory of the Massachusetts Institute of Technology. Support for the laboratory's artificial intelligence research has been provided in part by the Advanced Hesearch Projects Agency of the Department of Defense under Office of Naval Hesearch contract N()0()14-80-C-05()5. in part by National Science Foundation grants MCS-7912179 and MCS-811 76.'O and in part by the IBM Corporation. The views and conclusions contained in this document are those of the author, and should not be interpreted as representing the policies, either expressed or implied, of the Department of Defense, of the National Science Foundation, or of the IBM Corporation

As TWEAK works on a problem, it has at all times an incomplete plan, which is a partial specification of a plan that may solve the problem This incomplete plan could be completed m many different ways, depending on what constraints are added to it. thus it represents a class of complete plans The incomplete plan supplies partial knowledge of the complete plan that will eventually be chosen; ideally all possible completions of the current plan should solve the given problem 1 will say "necessarily foo" if foo is true of all possible completions of the current plan, and "possibly foo" if foo is true of some completion of the current plan.

A complete plan is a total "'time" order on a set of steps, which represent actions The plan is executed by performing the actions corresponding to the steps in the order given A step has a set of preconditions, which are[1] things that must be true about the world for it to be possible to execute the action A step also has postconditions, which are things that will be true about the world after the corresponding action has been executed. Pre- and postconditions are both expressed as propositions Propositions can be positive or negative, and have a content, which is a tuple of elements Elements can be variables or constants Functions, proposit lonal operators and quantification are not allowed.

Plans in TWEAK can be incomplete m two ways; the time order may be incompletely specified, using temporal constraints, and steps may be incompletely specified, using codesignation constraints A temporal constraint is a requirement that one step be before another; thus a set of temporal constraints is simply a partial order on steps A completion of a set of temporal constraints *C* is any total order *O* on the same set of steps such that *aCt* implies *sOt.*

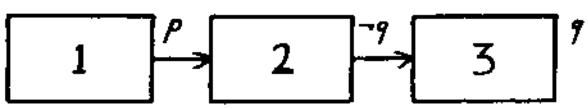
Codesignation is an equivalence relation on variables and constants In a complete plan, each variable that appears in a pre- or postcondition must be constrained codesignat ing with a specific constant. In execution, that constant will be substituted for the variable when the action is performed. Distinct constants may not codesignate Two propositions codesignate if both are positive or both are negative and if their contents are of the same length and if corresponding elements in the contents codesignate

TWEAK represents the state of the world as a set of propositions. This set changes as steps are executed A plan has an initial situation, which is a set of propositions describing the world at the time that the plan is to be executed Associated with each step in a plan is its input situation, which is the set of propositions that are true in the world just before it, is executed, and its output situation, which is the set of propositions that are true in the world just after it is executed. In a complete plan, the input, situation of each step) is required to be the same set as the output situation of the previous step. The final situation of a plan has the same set of propositions in it as the output situation of the last step. The time order extends to situations: the initial and final situations are before and after every other situation respectively. The input situation of a step is before the step and after every other situation that is before the step; the output situation of a step is after the step and before any other situation that is after the step.

Say that a proposition is true in a situation if it codesignates with a proposition that is a member of the situation Say that, a step asserts a proposition in its output situation if the proposition codesignates with a postcondition of the step Say that a proposition is asserted in the initial situation if it true in that situation A proposition is denied in a situation if another proposition with codesignating content but opposite truth value is asserted there, It's illegal for a proposition to be both asserted and denied in a situation

A step can be executed only if all its preconditions are in true in its input situation. In this case, the output situation is just the input situation minus any propositions denied the step, plus any propositions asserted by the step. This model of execution does not allow for indirect or implied effects of actions, any changes in the world must be explicitly mentioned as postconditions.

J will now sketch the derivation of a criterion for when a proposition is necessarily true in a situation Of course a proposition is necessarily true in situation if it is necessarily asserted in it, Once a proposition has been asserted, it remains true until denied Thus a proposition *p* is necessarily true in a situation if there is some previous situation in which it is necessarily true, and there is no possibly intervening step that possibly denies it for if there is a step that is even possibly inbetween that even possibly denies *p.* there is a completion in which the step actually is inbetween and actually denies *p* (A step possibly denies *p* by denying a proposition *q* which possibly codesignates with *p)* The converse of this criterion is not true; this plan illustrates an exception:



If *p* and *q* are possibly ('©designating, this plan has two classes of completions: one in which *p* and *q* actually codesignate, in which case *p* is asserted by step 3; and one in which *p* and *q* are noncodesignatmg, so that *p* is asserted by step 1. and is never denied In either case, p is true in the final situation, even though no step necessarily asserts *p* without an intervening step possibly denying it The complete criterion is as follows

Truth criterion: A proposition *p* is necessarily true in a situation *s* iff two conditions hold: there is a situation *t* necessarily equal or previous to *s* in which *p* is necessarily asserted; and for every step *C* possibly before *s* and every proposition *q* possibly codesignating with *p* which *C* possibly denies, there is a step *W* necessarily between *C* and *s* which asserts r, a proposition such that *r* and *p* codesignate whenever *p* and *q* codesignate The criterion for possible truth is exactly analogous, with all the modalities switched (read "necessary" for "possible" and vice versa)

This criterion can be computed in polynomial time, though it does exponentially much "work" by describing properties of the exponentially large set of completions of a plan. The remainder of TWEAK depends heavily on this theorem; its proof is surprisingly complex. Jt can be usefully viewed as a completeness/soundness theorem for a version of the situation calculus

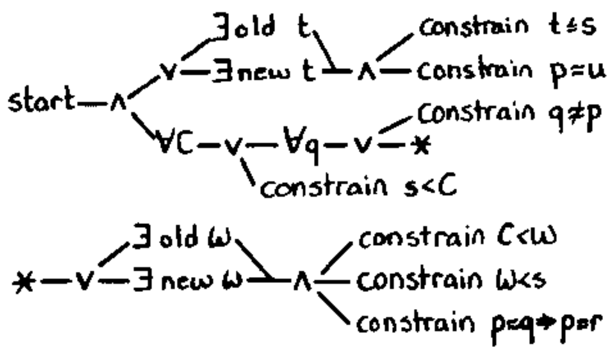
Now I will define problems and their solutions. A problem is an initial situation and a final situation, which are two sets of propositions A plan for a problem is a plan every proposition of whose initial situation is true in the initial situation of the problem A goal is a proposition which must be achieved (true) in a certain situation The goals of a plan for a problem are defined to be the propositions in the final situation of the problem which must be true in the final situation of the plan, and the preconditions of steps in the plan, which must be true in the corresponding input situations A complete plan for a problem solves the problem if all its goals are achieved Thus, a complete plan solves a problem if it can be executed in the initial situation of the problem and if the final situation of the problem is a correct partial description of the world after execution The aim of TWEAK is to produce a plan that, necessarily solves the problem it is given This plan may be incomplete in this case any of its completions may be chosen for execution.

TWEAK's contract is to produce a plan for a specific problem it is given TWEAK has at all times an incomplete plan, initially null, which is an approximation to a plan that solves the problem. The top-level loop of the planner nondeterministically chooses a goal that is not already achieved and uses a procedure which J will now describe to make the plan achieve that goal

The goal-achievement procedure is derived by interpreting the truth criterion as a nondetermmistic procedure Universal quantification over a set becomes iteration over that set. existential quantification a nondetermmistic choice from a set; disjunction a simple nondetermmistic choice; and conjunction several things that must all be done Also, an existentially quantified situation can be satisfied b\ nondeterministically choosing either an existing situation in the plan or a situation belonging to a newly added step The newly added step must represent an action that is possible to execute in the domain in which the problem is specified, the choice is among those that are allowed in the domain and that possibly assert, the desired goal

To make a situation be before another or to make two propositions codesignating or not codesignating. the procedure just adds constraints. These constraints may be incompatible with existing constraints; for example, you can't constrain *a* before / if you have already constrained / before ,s. The constraint maintenance mechanism signals failure in such cases, and the top-level control structure backtracks Since the set of things of things possibly asserted in a situation can not be changed, to make a proposition necessarily asserted there, the procedure constrains codesignation of the given proposition with one of those asserted

The following diagram defines the nondeterministic procedure



Because the truth criterion is sufficient as well as necessary, this achievement procedure encompasses *all* the ways to make a plan achieve a goal In this respect TWEAK can not. be improved upon.

Step addition adds new preconditions to the plan that need to be achieved, and the added step may also deny, and so undo, previously achieved goals Therefore. TWEAK tries to avoid step addition. This is not always successful There are three possible outcomes to planning: success, in which a plan is found; failure, when the planner has exhaustively searched the space of sequences of plan modification operations, and every branch fails; and nontermination, when the plan grows larger and larger and more and more operations are applied to it, but it never converges to solve the problem

The central theorem of this paper is the following

Correctness/completeness theorem: If TWEAK, given a problem, terminates claiming a solution, the plan it produces does in fact solve the problem. If TWEAK returns signalling failure or does not halt, there is no solution to the problem

The theorem follows easily from the truth criterion

The rigor of TWEAK's formulation has several uses Previous planning research becomes substantially clearer when analyzed with the tools built in constructing TWEAK In [1,] present a detailed history of planning, showing that the classic planners are more similar than has previously been realized

Another use of this rigor is that the range of applicability of state-of-the-art planning techniques becomes clear The correctness of TWEAK and similar planners depends crucially on details of the representation of actions. Useful extensions to this representation, such as range restrictions for variables, non-atomic propositions, derived effects of actions, actions whose effects depend on the situation m which the are applied, and changes in the world due to agencies other than execution of the constructed plan, invalidate the truth criterion, and so the correctness of TWEAK. There seems to be no simple way to extend the criterion to accommodate these effects The difficulty is just that of the McCarthy frame problem 2 ; my thesis suggests an approach to its solution

ACKNOWLEGEMENTS

This paper incorporates suggestions Phil Agre, Steve Ragley, Alan Bawden, Mike Brady, Gary Drescher. Margaret Fleck. Walter Hamscher, Scott Layson, David Mc.Allester, Kent Pitman. Charles Rich, Mark Shirley. Reid Simmons, and Dan Weld

REFERENCES

E Chapman, David, *Planning for Conjunctive Goals* MIT Technical Report 801'. May, 1985

:2 McCarthy, John, and Hayes. Patrick, "Some Philosophical Problems from the Standpoint of AI " *Machine Intelligence 4.*B Meltzer and D. Mitchie, eds Edinburgh: Edinburgh University Press, 1969

;l3| Sacerdoti, Earl D., *A Structure for Flans and Behavior* New York Elsevier North Holland, 1977

I4I Stefik, Mark Jeffrey, *Planning with Constraints* Stanford Heuristic Programming Project Memo 80-2

i5j Tate, Austin, "Project Planning Using a Hierarchic Non-linear Planner." Department of Artificial Intelligence Research Report No 25, University of Edinburgh, Edinburgh, August 1976

|6I Wilkins, David E., "Domain-Independent Planning Representation and Plan Generation." *Artificial Intelligence 2'2'.'*> (1984) pp. 269 301