Planning for Conjunctive Goals:

The State of the Art and Ahead

This paper describes the work of David Chapman, MIT 1987

Steve Woods, University of Waterloo, November 1989

1. Introduction

This paper constitutes a discussion of the work of David Chapman in the area of planning for conjunctive goals. Specifically, TWEAK, a domain independent nonlinear planner designed and implemented by Chapman will be discussed in light of planning in general . The major contributions of his work will be identified, and highlights of other areas of planning will be brought into focus with some of the results of Chapman's work. Specific sections of this paper will discuss the problem Chapman faced in tackling the planning problem, the solution he arrived at, the major contributions of the work he did to the AI field in general, some of the areas in which his work falls short, and finally, how Chapman's work can be built upon or extended. I think it is important to briefly describe the type of planning which Chapman concerned himself with. Domain independent conjunctive planning basically attempts to achieve several goals simultaneously (conjunctive) in a generally useful or reusable manner (domain independent). The problem of interacting subgoals arises in this type of planning, a classic example of which is the Sussman Anomaly, where the 'work' done by a planner in achieving one goal is undone in the achievement of another goal. A class of planning known as 'nonlinear planning' can solve this problem, and Chapman's solution is based on this concept due to Sacerdoti [2]. I will expand on nonlinear planning later in my discussion of TWEAK.

2. The Problem

David Chapman first became interested in planning in an unorthodox manner. He wanted a planner to be a part of a integrated problem solver he was working on, and had heard that Sacerdoti's NOAH [3] was the state of the art in planning. After several failed attempts to implement NOAH from it's description in the aformentioned reference, he succeeded in that he had a working planner. Chapman points out in his paper that to use some routine as a part of a system (in this case, the NOAH based planner in his integrated problem solver), it is important to know for certain that it works. In the course of showing that the planner was reliable, Chapman needed to simplify the algorithm of the planner, and evaluate it with some rigor. Of concern to

Chapman was both the correctness of the planner algorithm he was to use, and its completeness. That is to say, it was important to him that the algorithm was provably correct in that if the algorithm found a solution, it was a correct solution, and that the algorithm was complete in that if a solution to the problem existed, the algorithm would find it. The actual implementation of a planner (named TWEAK by Chapman) is a "rigorous mathematical construction", encompassing most of the previous domain independent planning techniques known, and improving much of what Chapman terms ’scruffy’ research by making it ’neat’. What has been stressed throughout Chapman’s work is rigor in all aspects of the implementation and discussion about his planner. Sacerdoti claims in [3] that "... it should be possible to define an algebra of plan transormations...a body of formal theory about the ways in which interacting subgoals can be dealt with." Chapman has accepted this challenge and his landmark paper is the result.

3. Chapman’s Solution

Chapman’s solution, TWEAK, can be viewed as three layers or parts: plan representation, the way in which a plan achieves a goal, and the control structure of the planner. I will look at each in turn.

Solution part 1 : Plan Representation

Chapman describes at some length terms and defintions of all of the many parts of his planner. I will not repeat these at the length he does in his paper, however, I will attempt to capture the essence of the plan representation he describes. TWEAK makes plans by incrementally specifying partial descriptions or constraints that the plan must fit. This method is known as ’constraint posting’. Constraint posting equates to a search methodology which progressively removes portions of the search space through constraints which rule them out. The advantage of this ’constraint’ approach is that a reduction in the amount of strictly arbitrary choice is enjoyed, and a corresponding reduction in the amount of backtracking required in planning is realized. When TWEAK is working on a problem, it maintains a specification of the planning it has done

to a given point (the incomplete plan which is a partial specification of the eventual complete plan), and it keeps on adding (constraints) to this plan until all of the possible completions of the plan solve the problem. We can say that a proposition which is satisfied in all of the possible completions of a plan is 'necessarily true', and that a proposition which is satisfied in some possible completion of a plan is 'possibly true'. Necessary and possible truth are concepts that will be required to describe TWEAK more fully later. It should be noted that the addition of a constraint to an incomplete plan could conceivably make the plan inconsistent (ie consider constraining some variable *x* to bind to a constant *a*, while the variable is already constrained to bind to a different constant *b*). In this case backtracking would be invoked and some other plan completion path not constraining *x* to bind to *a* would be pursued. The number of plan completions is exponential in size, so computation of the necessary or possible truth of propositions in a plan is prohibitively expensive. However, TWEAK does utilize a polynomial time algorithm to compute possible and necessary 'properties' of the incomplete plan. This algorithm will be examined later in this paper.

As in other planners, a complete plan in TWEAK is a total order on a finite set of steps or actions. A step in TWEAK has a set of preconditions which must be true in order for an action to occur, and a set of postconditions which are guaranteed to be true after the action has occurred. In each case, set elements are expressed as propostions which are function-free atomic (ie $p(x)$, $\neg p(x)$, $p(x,y)$, etc). Unlike most other planners, TWEAK utilizes the notion of a partial order on it's steps while a plan is incomplete. This concept makes TWEAK nonlinear in that the order of steps need not be specified entirely until completion of the plan. This nonlinearity will be discussed at more length later in this paper. TWEAK's plans are deemed to be incomplete in two cases only. In the first case, a plan is incomplete while the time order of the plan's steps is not completely specified through 'temporal' constraints which require that certain steps precede others. For instance, if a plan had three steps *A*, *B*, and *C*, and at some point in planning, *A* was constrained before *B*, and *C* before *B* then a partial order is imposed: $A < B$; $C < B$. In order for the time order of this incomplete plan to be completely specified, the relationship between *A* and

C needs to be clarified. Additionally constraining A before C gives the total order $A < C < B$. In the second case, plans are incomplete when steps in the plan are not completely specified through 'codesignation' or unification contraints which determine variable to constant 'assignments' or 'codesignations' (ie all variables must be 'bound' or 'codesignated' to a specific constant in order for a plan to be complete). Simply put, codesignation constraints can enforce either codesignation of elements (ie $x \approx y$ meaning variable x is constrained to codesignate to variable y), or non-codesignation (ie $x \not\approx y$ meaning variable x is constrained not to codesignate with variable y). Rules for codesignation parallel those of unification, and so I will not expound on them further. States of the world in TWEAK are described through 'situations', which are sets of propositions. A plan has an initial situation (before plan execution), a final situation (after plan execution), and associated with each step are input (before), and output (after) situations. Quite simply, a proposition is true in a situation if it codesignates with a proposition that is a member of the situation. A step can only be executed if all of its preconditions are true in its input situation. The step's output situation is simply the input situation with any propostions denied by the step removed, and any propositions asserted by the step added. Consider a State $S1$ consisting of propositions $p(a)$ and $\neg p(b)$, and an action A with preconditions $\{ p(a), \neg p(b) \}$, and postconditions $\{p(b), p(a)\}$. If $S2$ is the state after action A is performed, it will consist of:

$$S2 = S1 - (p(b) : \text{denied by } A) + (p(b) : \text{asserted by } A) = \{p(a), p(b)\}$$

It should be noted that the addition of asserted propostions cannot be performed before the subtraction of denied propositions since the situation could arise, as it would here, where the intermediate result of state S2 would allow inconsistencies such as $\{ p(a), \neg p(b), p(b) \}$, and clearly $p(b)$ and $\neg p(b)$ cannot be true simultaneously in some state of the world. All changes in the world of TWEAK must be mentioned explicitly in this fashion, neither uncertainty of execution of an action, nor indirect or implied effects are allowed. TWEAK is intended to solve problems, where a problem is composed of the initial situation propositions or starting point of the plan,

and the final situation propositions, which must be achieved in order for the plan to be complete. A complete plan solves a problem if the plan can be executed in the initial situation or world state of the problem, and the final situation of the problem is a correct partial description of the world state after execution. TWEAK's aim is to produce a plan that necessarily solves the problem it was given.

It is important to know during the planning process whether a proposition is necessarily or possibly true in order to determine whether a goal is or could be statisfied, and thus direct our building of a plan. Chapman defines his "Modal Truth Criterion (MTC)" which will allow us to procedurally determine the truth of a proposition. Chapman deals with the 'Frame Problem' in TWEAK by assuming that a proposition, once asserted by some step, remains true in later steps unless denied. This assumption may appear simplistic at first, but when viewed in the framework of his truth criterion, it's power becomes evident. Chapman's Modal Truth Criterion states: A propositon *p* is necessarily true in a situation *S* if and only if, for some step *T* (known as the establisher), necessarily equal or previous to *S*, *p* is necessarily asserted by *T*, and for all steps *C* (known as a potential Clobberer), possibly previous to *S*, and all propositions *q* in *C* possibly codesignating with *p*, if *C* denies *q*, then there must exist some step *W* (know as a White Knight), necessarily between *C* and *S* such that *r* is a proposition asserted by *W*, and if *p* codesignates with *q*, then *r* codesignates with *p*.

Similarly, possible truth can be determined by substituting "possible" for "necessary" and vice versa in the above description. In the discussion to follow regarding TWEAK's method for making a plan achieve a goal, I will show a procedural interpretation of this criterion. Of importance, as alluded to earlier in this paper, is the fact that this criterion can be computed in polynomial time.

Solution part 2 : Plan Achievement

The top level of TWEAK chooses goals repeatedly and attempts to make the plan achieve these goals. By treating the necessary truth criterion explained in the previous section as a non-deterministic algorithm, we obtain a goal achievement procedure. The criterion Chapman defines describes all of the ways in which a proposition can be made to hold or be necessarily true in some situation. The criterion defines explicitly the constraints that need to be added to the plan definition to make the goal true. The nondeterministic portion of this procedure is simply the choice of which constraint ’type’ or ’plan modification operator’ to use in goal achievement. If the constraints required by the selected ’operator’ are inconsistent with the existing constraints in the current incomplete plan (ie attempting to constraint some step T before some step *S* when step *S* has already been constrained before T), then a ’failure’ would be signalled from the constraint maintenance module of TWEAK, and the control structure would backtrack, and select a different ’plan modification operator’.

A description of the 'Modal Truth Criterion (MTC)' in terms of it's value as a plan modification algorithm would be useful in clearing up any confusion at this point. I will first show the parse tree of Chapman's MTC, and then the general algorithm which would serve a the plan modifier. Notice that square bracketed titles indicate the specific plan modification operators, and also that "(?)" indicates a choice not specified in its nature by Chapman, but I assume it to be nondeterministic. Also, the algorithm shown is strictly my interpretation of his description, and is not included in his paper as such.

MTC Plan Modification Algorithm

[Want to achieve some goal proposition *p* in step *S*]

[First need to establish *p*]

1. Either (choose nondeterministically):

A. [Simple Establishment]

Find(?) some step *T* existing, which both:

a) Is possibly before or equal to *S*

b) Could assert a propostion *u*, which could possibly codesignate with *p*

Constrain *T* to be necessarily before or equal to *S*

Constrain *u* to necessarily codesignate with *p*

or:

B. [Step Addition]

Add(?) some step *T*, which both:

a) Could be possibly before or equal to *S*

b) Could assert a propostion *u*, which could possibly codesignate with *p*

Constrain *T* to be necessarily before or equal to *S*

Constrain *u* to necessarily codesignate with *p*

[Insure no Declobbering occurs]

2. Loop through all steps *C*:

3. Either (choose nondeterministically):

A. [Promotion]

Constrain *S* to be necessarily before *C*

or:

B. [Loop through all propositions *q* in step *C*:]

Either (choose nondeterministically):

i) [Separation]

Constrain *q* to noncodesignate with *p*

or:

ii) [White Knight]

Either (choose nondeterministically):

a) Find(?) some step *W* which satisfies:

(1) *C* possibly before *W*

(2) *W* possibly before *S*

(3) *W* asserts a proposition *r* such that if

Constrain *C* before *W*

Constrain *W* before *S*

Constrain *r* in *W* such that if

p codesignates with *q*, *p* codesignates with *r*

or:

b) Add(?) some step *W* which satisfies:

(1) *C* possibly before *W*

(2) *W* possibly before *S*

(3) *W* asserts a proposition *r* such that if

p codesignates with *q*, *p* codesignates with *r*

Constrain *C* before *W*

Constrain *W* before *S*

Constrain *r* in *W* such that if

p codesignates with *q*, *p* codesignates with *r*

Chapman proves in the appendix of his paper that the MTC is correct in that the goals (and hence problems) solved by the goal achievement procedure are solved correctly. I will not cover the details of his proof, I will only mention that he builds on the work of Warren (1974), Waldinger (1975), Rosenchein (1981), Kautz (1982), and Pednault (1985) who together prove the correctness of linear planners, and Sussmann (1973) who first put forward a 'loose' correctness argument.

As an extension of his proof of MTC's correctness, Chapman also proves that TWEAK is both correct and complete, that is, if TWEAK, given a problem, terminates giving a solution, the produced plan solves the problem, and if a solution exists, TWEAK will find it.

I will mention each specific plan modification operation listed above briefly and outline a simple example of each for explanation purposes. Apart from these Chapman claims that no other possible way to accomplish a goal exists, except for the removal of a clobbering step *C*. He observes that since each step is added to accomplish some goal, removal would result in negative progress, and claims that the search control structure would guarantee that the same plan without the clobbering step would be found eventually anyway, and that it is never the case that the only

way to achieve a goal would be to remove a step. Chapman does not prove any of his claims about step removal explicitly in this paper, but since his MTC is proved correct and complete, clearly step removal is not strictly necessary.

Consider the following blocks world scenario for the operator examples.

Initial situation (I) : $ontable(c)$ $on(a,b)$ $on(b,c)$ $clear(c)$

Step Template:

Action: $Puton(x,y)$

Preconditions: { $on(x,z)$, $clear(x)$, $clear(y)$ }

Postconditions: { $on(x,y)$, $\neg on(x,z)$, $\neg clear(y)$, $clear(z)$ }

PLAN MODIFICATION OPERATIONS

A. ESTABLISHMENT Operations

1. Simple Establishment

A goal of a step S is p, and a step T exists in the current plan asserting a proposition q. Simple Establishment of p entails constraining q to codesignate with p, and constraining T to necessarily precede S.

Example: A goal chosen as part of the preconditions for an existing $pickup(x)$ action/step S is $clear(x)$. This goal is satisfied through simple establishment by using the initial state I as the establisher step for x. Step I is constrained to precede S, and x is constrained to codesignate with c.

2. Step Addition

A goal of a step S is p, and a step T is added to the current plan such that T asserts a proposition q. Step Addition establishment of p entails constraining q to codesignate with p, and constraining T to necessarily precede S.

Example: A goal chosen as part of the preconditions for a step S which performs a $pickup(x)$ action is $clear(x)$. This goal is established through step addition of step T, a $Puton(x,y)$ action to the incomplete plan. Step T is constrained to precede S, and x is

constrained to codeisgnate with *z* in the postcondition $clear(z)$ of the *Puton* action.

B. DECLOBBERING Operations

3. Promotion

For all steps *C* possibly before *S* where *C* denies a proposition *q* that possibly codesignates with a desired goal proposition *p* in *S*, 'promote' step *C* by constraining *C* to be strictly after *S* (*S* before *C*).

Example: Consider the incomplete plan with unordered steps Step1 & Step2 & Step3:

Step1 = ... postconditions = $\{\neg\, clear(a), clear(b)$ ... $\}$

Step2 = ... postconditions = $\{clear(a), clear(b)$ ... $\}$

A goal is chosen in Step3, $\neg\, clear(a)$ Clearly, since Step1 and Step2 are unordered, Step2 which negates $\neg\, clear(a)$ possibly precedes Step3. Promotion of Clobberer Step2 results in the addition of a constraint Step2 > Step3, Step2 is necessarily after Step3.

4. Separation

For all steps *C* possibly before *S* where *C* denies any proposition *q* that possibly codesignates with a desired goal proposition *p* in *S*, 'separate' *p* and *q* by constraining *p* to not codesignate with *q*.

Example: Consider the incomplete plan with steps Step1 and Step2:

Step1 = ... postconditions = $\{\neg\, clear(x)$ ... $\}$

Step2 = preconditions = $\{\ clear(a)$ ... $\}$ ...

and Step1 is already constrained to precede Step2 (Step1 < Step2). Since $\neg\, clear(x)$ in Step1 possibly codesignates with $clear(a)$ in Step2 necessarily after Step1, we 'separate' *x* and *a* by constraining *x* and *a* to noncodesignate ($x \not\approx a$).

5. White Knight - old step

A goal of a step *S* is *p*, and a step *T* (necessarily before *S*) exists in the current incomplete plan asserting *p*. Some step *C* also exists such that *C* possibly lies between *T* and *S*, and some proposition *q* is denied in *C*, and possibly codesignates with *p* in *S*. Some step *W*

exists in the current plan such that *W* asserts a proposition *r* that also possibly codesignates with *p*. Performing 'White Knight - old step' entails constraining step *W* to necessarily follow *C* and also to necessarily precede *S*. As well, the proposition *r* in *W* is constrained to codesignate with *p* if propostion *q* in *C* codesignates with *p*.

Example: A goal *clear*(*a*) is chosen as one of the preconditions of a step Step4. The current incomplete plan looks like:

Step0 ... postconditions {*clear*(*y*) ... }

Step1 ... postconditions {*clear*(*a*) ... } Step1 < Step4 (establisher)

Step2 ... postcondtions {¬ *clear*(*x*) ... } (clobberer)

Step4 preconditions {*clear*(*a*) ... } ... (goal)

Clearly Step2 can possibly lie between Step1 and Step4 (since Step2 is currently unordered), and since *x* can possibly codesignate with *a*, Step2 can possibly deny *clear*(*a*). Application of 'White Knight - old step' would entail that a constraint would be added to the incomplete plan constraining Step0 to necessarily follow Step2 (Step0 > Step2), and to necessarily precede Step4 (Step0 < Step4). As well, the variable *y* in Step0 would be constrained to codesignate with a if the variable *x* in Step2 codesignated with *a*.

6. White Knight - new step

   This operation is identical to the above operation except that Step0 would be added to the plan from a nondeterministically chosen step/action template, rather than selected from existing steps in the plan.

A couple of points on these operations are worth observing here. First, step addition clearly adds new preconditions to the incomplete plan which must be achieved at some point. As well, the added step could deny previously achieved goals. For these reasons, TWEAK 'prefers' (in some manner not explicitly described by Chapman) constraint posting to step addition. Second, Chapman points out that since the MTC is sufficient as well as necessary, these operations encompass ALL the operations needed to make an incomplete plan achieve a goal. As a result, TWEAK cannot be improved upon in this regard. This result must be seen as a large

accomplishment in view of earlier planners which made little or no attempt to fully describe or justify their operators.

Solution part 3 : Control Structure

As mentioned earlier in this paper, TWEAK begins work on a problem with an initially empty incomplete plan, and gradually adds steps and constraints as it attempts to solve goals. To do this, TWEAK must select unachieved goals (nondeterministically) and apply to goal achievement routine we have seen. Since it is not always possible to select the best goal first all of the time, the top level control of TWEAK can be seen as a search through the space of alternate paths throught the goal achievement procedure. Chapman selected dependency-directed breadth first search for TWEAK. Chapman observes that the use of step addition can allow the plan to grow arbitrarily, and the search may never converge on a solution. More strictly, Chapman proved that there are three possible outcomes in TWEAK. Either TWEAK succeeds in finding a solution plan, fails when it has completely searched the possible search space, or TWEAK never terminates.

4. Contributions

Throughout his paper, Chapman is quick to point out the contributions his paper has added to the area of domain independent nonlinear planning, and planning in general. To his credit, he often points out areas in planning that would (or, as in many instances, can) benefit at all from future work. I will comment on the contributions he describes, and offer other contributions that I see in this work.

Chapman points out in several parts of his paper that research in general, and research in planning and AI in particular tend to work in cycles of what he terms ’scruffy’ research and ’neat’ research. He typifies ’scruffy’ as new work in a field that is necessarily difficult to duplicate (like planning work done previous to Chapman’s paper), and ’neat’ as simply the more formal and rigorous interpretation of the ’scruffy’ work. I will be involved in the implementation

of TWEAK here at the Unverisity of Waterloo as part of further research into Chapman's concepts and possible extensions, so I will soon have a good idea of whether Chapman's paper fulfills his promise of 'neatness' and is relatively straightforward to duplicate and implement. Since he is explicit in his accounts of methodology and leaves little fuzzyness in terms of the MTC heart of TWEAK, I believe his contribution to future planning research is great.

Chapman states that the introduction of a provably correct 'Modal Truth Criterion' and its use in his proof of TWEAK's correctness and completeness is quite likely the largest single contribution of his paper. When the implications of this theorem are examined at some length, clearly there is justification for this observation. As a direct result of MTC and the subsequent proof, an algebra of plan tranformations which is sufficient and necessary to non-linear planning emerges where there were only poorly specified operations in other previous planners. Chapman's explanation of the price to be paid of extending TWEAK's very limited action representation outlines some of the importance of this improvement to the global knowledge of planning. The two major restrictions on TWEAK's action representation are that it cannot allow the effects of actions to depend on the situation in which they are applied, and that all changes made by an action must be explicitly represented in the postconditions of the action. In the first case TWEAK cannot allow for situations such as (example due to Chapman) in the blocks world only zero, one, or two blocks may be on a given block. A *space*(*block*) function tells how much room is left on a block. A precondition of using a *Puton*(*onblock*,*baseblock*) action would then be that *space*(*baseblock*) be non-zero. Hence a situation is achieved in which the representation has postconditions (results) that are functionally dependent on the input situation. Chapman demonstrates that if TWEAK were extended in its representation just enough to allow for this, the Modal Truth Criterion would fail as a result of two steps acting together to deny some proposition *p* in some step *S*. (Neither of the two individual steps (say *puton* where space is decremented by 1 as a result) would deny *p*, and so MTC would not require that they not precede *S*. Clearly if not constrained, it is possible that both steps are before *S*, and therefore *p* (say space non-zero) is denied.) The second major restriction on TWEAK, (example due to Chapman, and,

as a point of interest, was suggested on the midterm of cs686 this semester) where actions have side-effects not explicitly specified in the postconditions, can be illustrated by considering the case where block *a* is on block *b*, and block *b* is moved somewhere. Explicitly, block *b* is moved, but the sideeffect is that block *a* is moved also. If TWEAK is allowed to include deduction within situations, this sideffect action can be represented. Once again, with this addition, MTC fails through two (or more) steps working together to deny some propostion where none of the offendiing steps themselves does the dirty work. The major contribution of this demonstration of MTC's failure in cases of extension of TWEAK's action representation is in Chapman's explanation of the 'costs' that are necessarily incurred by modifying the MTC. Chapman points out that the extension of action representation over that described in TWEAK runs into difficulties with the Frame Problem, which is equated in TWEAK to finding an efficiently implementable truth criterion. Since Chapman proved that determining necessary truth in the case where action representations are extended to represent conditional actions, dependency of effects on input situations, or derived side effects is NP-hard (SAT - satisfiability reduces to planning, and a nondeterministic polynomial algorithm exists for planning), the pursuit, in general, of domain independent planning in these cases will not lead to the long sought polynomial solution (Unless of course it turns out that P = NP and the joke is on everyone).

Clearly planning is still a valuable area (despite it's nature in general), and Chapman suggests that we can either ignore the NP-hard nature of planning, and (to steal a term from Shoham) use the 'Ostrich Principle' and assume it will be alright for our application, relax our correctness requirement thus producing a planner that sometimes doesn't quite work, or relax our generality requirement and head off into domain dependent planning. I will discuss these options as possible improvements to the state of the art later in this paper, including Chapman's ideas and my own.

5. Shortcomings

As with any work, Chapman's paper does have some shortcomings. He himself mentions many aspects of future work using the ideas he has created as a basis, however, I will discuss these in the final section of my paper dealing with possible future work in planning. What I will discuss in this portion of this paper are possibilities Chapman may have missed, areas in which he has not been explicit enough in his coverage of planning problems, and proofs that Chapman has omitted that seem to be required for a true 'neatifying' of planning to be complete.

In Chapman's explanation of plan modification operators, he noticed that one way in which a goal can be achieved is to remove a clobbering step in the plan. At this point he makes the observation that this action would make negative progress in every case. As well, Chapman claims that the search control of TWEAK would find the same solution eventually that would have resulted from the use of step removal. I feel that the use of the term eventually in this type of statement should be supported by either a description of why the difference in time in insignificant, or at least some justification of the claim. It is not obvious to me that TWEAK could not benefit in some class of problems by removing a step to solve some goal, and then asserting some other step that could perhaps solve (say) several other goals. All I claim is that in general, it is not obvious that no gain can be achieved through step removal.

Earlier in the paper the two plan modification operators dubbed 'White Knight' in Chapman's truth criterion were discussed at some length. In fact, Chapman does not use this operation in the case where the white knight step is distinct from the establishing case in his implementation of TWEAK. Chapman claims that (in this case), white knight is actually useless as an independent operation since either the white knight will turn out to assert the goal (at we might as well consider the output situation of the 'white knight' step as the actual establisher, or it will not, and we might as well have used the modification operator 'separation' to defeat the clobberer. For an example of this, see the earlier discussion of plan operations, case numbers 5 and 6. The point is that either the white knight step W will end up asserting r codesignating with p in S (essentially asserting p in the ground instance) in the case where C denied q and $q \approx p$,

since $q \approx p \Rightarrow p \approx r$ by MTC and this ends up being equivalent to the case where *W* acts simply as an establisher asserting *p*, or the white knight step *W* will end up not asserting *r* (and thus we know by modus tollens that the clobbering of *p* through *C*'s denial of *q* codesignating with *p* did not happen, and thus *q* did not codesignate with *p* ($q \approx p \Rightarrow p \approx r :: \neg (p \approx r) \Rightarrow \neg (q \approx p)$), and this is equivalent to the situation where *C* was 'defeated' by separating *q* and *p* by constraining *q* and *p* to not codesignate. I simply point out that a strict proof is not given at this point, and that despite the fact that I cannot construct a counter example (yet), the question of the general case white knight's usefulness is not closed. Although the discussion above is strictly my own, the question of the validity of Chapman's claim about white knight was suggested to me by Qiang Yang in a discussion.

I have said a great deal about the value of Chapman's rigor in that his proofs about planning give some very strong ideas as to where work could be valuably focused in the future, and which areas are strictly undecidable in the general case. Chapman proves in his 'First Undecidability Theorem' that planning is undecidable provided that problems have a potentially infinite initial state, and that the shortest plan to solve a problem is not arbitrarily large. One area Chapman points out that is open to discussion is whether or not planning is undecidable in the case where problems do not have a potentially infinite initial state. Clearly in any real world example, the initial state (specified at least) must be finite. We know that extending TWEAK's representation (even for finite states) proves to be undecidable, but there still is an open question as to whether TWEAK itself is decidable in its described form for finite states.

In Chapman's discussions about modifying his Modal Truth Criterion to attempt to allow extended action representations, he alludes to the fact that the result of such modifications is sometimes the same as reverting to linear planning, and hence is not efficient. Specifically, the "synergistic" collusion of two or more steps to deny some proposition can be dealt with by either considering all of the possible completions of a plan, thus defeating synergy since the state of the world is always known in linear planning, or by considering (all) the sets of steps in trying to find establishers and clobberers. The first solution is clearly a reversion to linear planning, and

the second solution also gives an exponential search space since there are an exponential number of subsets of a set of steps. However, it is an open question as to the degree nonlinear planning is in fact superior to linear planning. Chapman believes that nonlinear planning is exponentially worse than linear planning, and points out that empirical evidence supports this view, however the proof of this seemingly innocuous point has never been shown.

6. Possible Future Work

Many areas in planning are left to be explored. It is a relatively young field within AI itself, a very young science. Of particular interest are applications of work done in other areas of AI to applications such as planning which can benefit from the knowledge gained elsewhere. In Chapman's paper he mentions in passing a few areas which could benefit from future work and improvements, I will discuss both these and some of my own ideas in the following paragraphs.

Search is clearly a critical topic in any real application of planning. Heuristics that are domain specific can gain much by understanding in advance what the search space could possibly look like, and can easily make the difference between a viable implementation that is decidable in the instances that are of concern in some particular case, and an implementation that is of no use. Chapman uses dependency-directed breadth first search in TWEAK, primarily because of it's ability to perform nicely in certain cases where search spaces decompose nicely into independent subparts. Chapman also points out later in his paper that search control is the aspect of domain-independent planning understood least. The selection of plan modification operators is a portion of the planning concept that certainly deserves more investigation. The concept of solving multiple goals through the selection of only one operator can be viewed as effectively narrowing the search space more quickly than solving them one at a time, but Chapman is quick to discredit this approach since there are exponential subset of the set of goals which might fit into this category. The reason he does this is because he is concerned primarily with general cases and proving his results in the general case. I think that some interesting heuristic approaches might arise from studying further these types of 'beneficial' goal interactions in

some instances. Certainly it is easy to contrive an example where the intelligent operator selection can eliminate unnecessary search (goals $goal(a)$ and $goal(b)$ exist in the incomplete plan, and some step G asserts $goal(a)$ and $goal(b)$ in its postconditions, yet some other step T asserts only $goal(a)$ in its postconditions, and denies some $goal(c)$. So now, say $goal(c)$ has precluded a simple solution to $goal(b)$ in the incomplete plan through another step ....). The point I make is that in certain cases such as these it would be beneficial to recognize that step G is the one to choose. A point arises here that I'm sure Chapman would agree with, that selection of these types of heuristics may be most beneficial in a domain specific implementation where more is known about specific goal interactions ahead of time.

A very interesting area of planning that Chapman touches only briefly on is the use of planning itself in the selection of the plan modification operator. This area of planning is known as 'metaplanning' and there has been some work done on it. Chapman notes that TWEAK could not be used as a metaplanner since it has an action representation that is too weak. Wilkins makes a brief statement about metaplanning in [7]. He says that "interesting (read valuable) metaplanning appears very difficult in a system that makes the Strips Assumption". Clearly this is an area that deserves some attention, although I have given it none in the course of this project.

Just as the top level search of planning stands to gain much through domain dependent heuristics and knowledge, so too do other areas of planning. TWEAK, as explained at some length uses a minimal action representation. Certainly there will be many domains that may take advantage of stronger action representations and manage to 'live' somehow with the consequences. I mentioned previously that we must either relax the generality of a planner or the correctness if we wish to gain stronger action representations. Clearly, relaxing the generality equates to making a planner domain specific, and Chapman comments on some of these possibilities. Chapman claims to have attempted the creation of extended representations and corresponding truth criterion for those representations with little difficulty, but found that the criterion were very different. Armed with this knowledge, Chapman ponders whether or not

domain independent planning is of any real value, or whether the domain a planner is to be used for necessarily should supply its own truth criterion. This area seems to show a great deal, if not all, of the promise for planning in the near future. Conditional actions, actions which have varying effects based on input situations, and derived side effects constitute some of the 'purchase' of loss of generality in planning.

In the real world, Murphy would claim that actions rarely have the desired or predicted outcome. The outcome of actions can often depend on factors that are totally outside the 'sphere of understanding' of a particular acting agent, or in fact, on totally random factors. Chapman views these types of actions as 'nondeterministic'. I would claim that in many real situations (and that would seem to be the direction planning is taking), the possibilities are in fact quite obvious. For example, when a block is placed on another block, we would expect (most likely) that the two blocks end up upon one another. However, what if they don't? Any of a number of factors could cause the top block to fall off, get picked up by some other agent, or perhaps get stuck in the hand of the stacker. In fact, in some situations in the real world, it can be very important that recovery of the planning agent in these abnormal types of situations is not delayed or, even worse, irrevocably interrupted. In situations such as these that are likely identifiable in advance within the domain, consideration must be given for developing sub-plans to recover from predictable errors to the state where the plan can continue as originally intended, or perhaps an entirely new plan could take effect at this time. The important considerations here are what actions would be 'critical' enough to justify spending this excess effort on, and what are the 'most likely' possible failures that we will need to plan in anticipation of? There is a definite relationship here to another field of AI work currently being pursued, that of knowledge representation. In knowledge representation, there has been much use of the term 'typically' with respect to the characteristics various objects possess. Exceptions to the 'typical' or 'norm' seem to be presenting some difficulty to easy construction of reliable and 'intuitively' correct knowledge bases. In view of planning, it is easy to see that a 'typical' situation is simply the result of an action you expect 'most', and the exceptions are the dreaded 'tigers' that seem to

creep into the real world and give our actions their nondeterministic quality.

7. Addendum

In the appendix of this paper, I have sketched out a proof verifying Chapman's observation that White Knight did not seem to be useful in general since it was handled by either Separation or Establishment in his implementation of TWEAK. The ideas represented here originated through discussions with Qiang Yang, and the proof was outlined by him to me. As well, a PhD student, Jean-Francois Puget, indicated to me that he had proven the issue in his forthcoming PhD thesis. Chapman's intuition seems to have been sound in that his implementation did not use the White Knight concept.

REFERENCES

1. David **Chapman**, Planning for conjunctive goals, *Artificial Intelligence*, vol 32, p.333, 1987.
2. E.D. **Sacerdoti**, The nonlinear nature of plans, Advance Papers, IJCAI 1975, Tbilisi, USSR, vol. 1, pp. 206-214.
3. E.D. **Sacerdoti**, *A Structure for Plans and Behavior*, American Elsevier, New York, 1977.
4. David E. **Smith**, A decision-theoretic approach to the control of planning search
5. Austin **Tate**, Interacting goals and their use Advance Papers, IJCAI 1975, Tbilisi, USSR vol. 1.
6. Robert **Wilensky**, *Planning and Understanding: A Computational Approach to Human Reasoning*, Addison-Wesley, 1983.
7. D.E. **Wilkins**, Domain independent planning: representation and plan generation, *Artificial Intelligence*, vol. 22, no. 3, April, 1984.

APPENDIX

The White Knight operation outlined by Chapman is of the following form:

We wish to achieve some goal proposition p in an arbitrary step S in our incomplete plan. Some step T necessarily before S has asserted p, however the problem which exists is insuring that no step between T and S denies p. So, for each step C possibly denying some proposition q which possibly codesignates with p (thus possibly denying p) between T and S, White Knight ensures that there exists another step W, asserting some propostion r, necessarily between C and S, such that if C denies q and q codesignates with p, then r in W codesignates with p as well. This 'insurance' logically reduces to "(W between C and S, and if C denies q codesignating with p in S, then r asserted by W codesignates with p in S", or more formally,

$(C < W < S) \wedge [\ (p \approx q) \Rightarrow (p \approx r)\ ]$

$(C < W < S) \wedge [\ \neg (p \approx q) \vee (p \approx r)\ ]$

$(C < W < S) \wedge [\ (p \not\approx q) \vee (p \approx r)\ ]$

$[(C < W < S) \wedge (p \not\approx q)] \vee [(C < W < S) \wedge (p \approx r)]$

The first part of this conjunction can be read as "Constrain W after C and before S, and constrain p in S to not codesignate with q in C". Clearly W is extraneous, and the result is "Separation" as defined by Chapman. The second part of the disjunction can be read as "Constrain W after C and before S, and constrain p in S to codesignate with r in W". Here W is acting as the "Establisher" as defined by Chapman, and C is extraneous. Thus we have reduced Chapman's White Knight operation to be either a case of "Separation" or of "Establishment", and nothing else, and a simpler new model emerges, just as sufficient and complete as Chapman's MTC.