Laforia Technical Report 90/29, December 1990.

From TWEAK to PWEAK:

Reducing the Non Linear PLanning Framework

Éric Jacopin
LAFORIA,
4, Place Jussieu, Boîte 169,
75252 Paris Cedex 05, France
jacopin@laforia.ibp.fr

Jean-François Puget
ILOG,
2 Avenue Galliéni, BP 85,
94253 Gentilly Cedex, France
puget@ilog.fr

ABSTRACT

The TWEAK system (Chapman 87) is a very nice synthesis of a long trend in planning and Artificial Intelligence, including STRIPS (Fikes & Nilsson 71). We derive a correctness criterion which is simpler that the one used for TWEAK since it does not involve modal operators and does not introduce neither white knights nor separation. This criterion is compared to the one of TWEAK and other related works. We then derive a correct and complete planning procedure from this criterion, as it was done for TWEAK. The resulting planner is proved correct (it finds good plans), complete (it can find all possible plans with appropriate control) and shall always stop with a solution if there exists one. Moreover, this planner avoids some looping problems of TWEAK. This planner is implemented in Prolog and Lisp.

1 Introduction

This paper is concerned with the framework of non linear planning using the action representation of STRIPS [3]. There has been little work in this area concerned with formalization. One of the main exceptions is David Chapman's TWEAK [2] which was developed in the idea of gathering and rationalizing the techniques used in the field. Although his work allowed better understanding of planning techniques, it is not as simple as it could be. We show in this paper how the techniques used in TWEAK can be simplified, and why they have to be.

The second section presents how the framework of non linear planning evolved from Chapman's work. Then, the third section presents PWEAK, a logically minimal construction of the framework based on TWEAK, which does not contain a white knight operator nor a separation technique. Finally, we discuss how the minimality of the framework of PWEAK can avoid some looping problems encountered with TWEAK.

2 The evolution of the non linear planning framework

In this section, we review systems that contributed to the formalization of the framework. Although, it is clear that Chapman's TWEAK is the very first non linear planner that has been proved to be correct and complete, Tate [10] claims that his planner, NONLIN, is complete. NONLIN is a NOAH [9] based planner. Tate argues that since NOAH did not keep any backtrack choice points, it was not complete. To prove that NONLIN is complete, Tate proposes two things: (i) to keep alternative choice points and (ii) to introduce two alternative orderings between instantiated operators in a plan [10, page 889]. But Tate did not introduce any formal proof of his arguments.

With TWEAK Chapman introduces a modal truth criterion (MTC). He proves that a plan is correct if and only if it satisfies his MTC. Figure 1 presents a graphic parsing of this MTC. This criterion can be interpreted as a planning algorithm. This algorithm has two tasks: (i) finding a situation where a proposition is asserted; this can be done either by finding a situation already in the plan or by inserting a new situation through the instantiation of an operator (i.e. the upper branch of the graphic of Figure 1), and (ii) ensuring that clobbering situations do not prevent the asserting process; this is done by inserting some ordering constraints between clobbering situations and establishing situations (i.e. the lower branch of the graphic of Figure 1). In Figure 1, T is the establishing situation for proposition p, s is the situation where p should be asserted, C is the clobbering situation and W the white knight situation. In his paper, Chapman proposes three ways for declobbering:

- Promotion. This process positions the clobberer after the situation where the proposition must be asserted.

- Demotion with a White Knight. This “tricky” process tries to find another establishing situation after the clobberer and before the situation where the proposition has to be asserted. Chapman names this another situation a white knight.
- Separation. This technique tries to avoid (by constraint posting) the codesignation between the clobberer and the establisher.

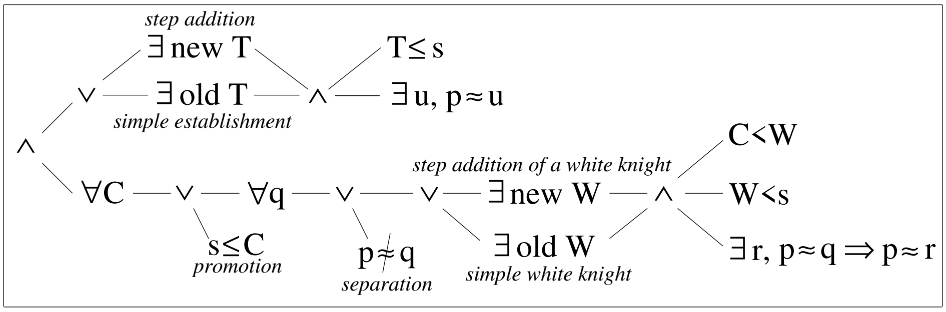

Figure 1: A graphic parsing of Chapman's planning mechanism.

About the white knight strategy, it is important to note that according to Tate's paper, NONLIN was only using promotion and demotion (with no white knight, though Chapman describes NONLIN as a white knight user). Moreover, a white knight is necessary to prove that TWEAK is correct and complete, but Chapman mentions [2, page 361] that his implementation of TWEAK does not use it and that finally the white knight is useless since it is nothing else than a well chosen establisher. The next section presents a correct and complete criterion with no white knight and no separation.

Like NONLIN, IPEM [1] (a post-TWEAK non linear planner developed at the University of Colchester), does not use a white knight; but their authors (Ambros-Ingerson and Steel) mention that a good establisher can be chosen in order to avoid the white knight strategy. However they do not present any formal proof of their argument.

Finally, McCluskey [4,5] presented an MTC that was not containing any white knight. McCluskey obtains his MTC by reducing Chapman's; he rewrites Chapman's truth criterion with the white knight as an ordinary establisher (i.e. W=T). McCluskey's MTC is important in two ways: (i) it is not containing the white knight strategy and (ii) it involves operators instead of situations that codesignate. But his criterion still involves separation (a situation based technique) and so is not minimal. Figure 2 presents McCluskey's planning mechanism where C,T and S denote operators; C is the clobberer, T the establisher and S is the operator the MTC tries to assert.

3 PWEAK: a reconstruction of TWEAK

This section presents PWEAK, a non linear planner implemented in Prolog and Lisp, that is correct, complete and uses only promotion and demotion as declobbering techniques. We first give definitions and then present PWEAK's truth criterion.

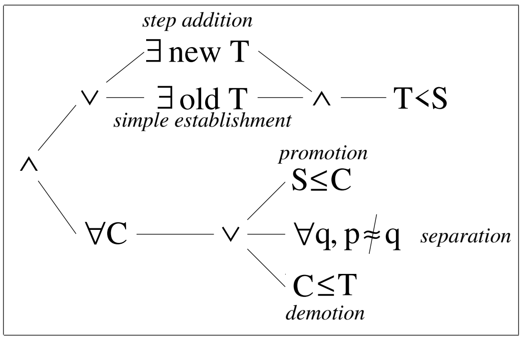

Figure 2: A graphic parsing of McCluskey's planning mechanism.

3.1 Definitions

The language used in non-linear planning is a subset of the predicate logic (i.e. predicate with constants or arguments).

A state is a conjunction of ground atoms.

An operator O is a couple (*pre*(O), *pos*(O)) such that *pre*(O) and *pos*(O) are set of states. *pre*(O) is called the preconditions set of O and *pos*(O) the postconditions set of O. A usual STRIPS operator is a triple (preconditions, add, delete) of sets of states. Our notion of operator is constructed on the following equivalences:

pre(O) = preconditions ∪ delete

pos(O) = (preconditions – delete) ∪ add

Therefore, the delete list is: delete = *pre*(O) - *pos*(O). An operator is totally instantiated; it contains no variable. We will remove this limitation later.

A problem is a tuple (σ_i, σ_f, Ω) where σ_i is called the initial state of the problem, σ_f the final state of the problem, and Ω the set of available operators for the problem.

From σ_i and σ_f, it is easy to construct the initial operator O_i for the problem as the couple ($\emptyset$, σ_i), and the final operator O_f for the problem as the couple (σ_f, $\emptyset$). Since the initial situation and the final situation are integrated into the description of an operator, planning only deals with operators, whereas Chapman uses both situations and operators.

A non linear plan Π for a problem is a couple (π, $<_p$) where π is a set of operators containing O_i and O_f; $<_p$ is a partial order on π, such that the initial operator is before every other operator and the final operator is after every other operator; i.e.

$$\forall\ O \in \pi,\ (O \neq O_i \Rightarrow O_i <_p O) \wedge (O \neq O_f \Rightarrow O <_p O_f)$$

Note that $O_i <_p O_j$ means that operator O_i is necessarily before operator O_j, and that $\neg(O_i <_p O_j)$ is not equivalent to $(O_j <_p O_i)$; i.e. O_i and O_j may not be comparable through $<_p$.

A non linear plan can be executed sequentially. We define a completion of a non linear plan $(\pi, <_p)$, to be a possible execution of the plan; i.e. a total ordering of the operators of $(\pi, <_p)$ that is compatible with the partial order $<_p$. Since there is always a total order $<$ compatible with $<_p$, there always exists a *completion* of a non linear plan, defining $(\pi, <)$ as a linear plan.

3.2 PWEAK's truth criterion

We here present PWEAK's truth criterion. All the proofs can be found in [6,7].

Intuitively, the last definition (stating a completion of a non linear plan) brings up that planning is highly concerned with orders between operators: some may not be applicable and/or others may clobber what has been asserted. We first define asserters and then un-asserters:

An operator T establishes a proposition p iff $p \in pos(T)$.
An operator C clobbers a proposition p iff $p \in \{pre(C) - pos(C)\}$.
One should note that $pos(O) \cap (pre(O) - pos(O)) = \emptyset$, which means that an operator cannot both clobber and establish the same proposition (i.e. add $\cap$ delete is empty). A clobberer of p is thus an operator that removes p from any states where it is applicable and an establisher of p is an operator that adds p to any state where it is applicable.

Hence, to build a **correct non linear** plan, the condition is that for every operator A_i of Π and for every precondition of A_i, there exists an establisher T of p which is before A_i, and there exists no clobberer C of p after T and before A_i. This must be true in every completion of the plan. We proved [6,7] that this amounts to the following criterion:
Theorem 1 (truth criterion): Let $(\pi, <_p)$ be a non linear plan for a problem. $(\pi, <_p)$ is a correct non linear plan iff:

$$\forall A \in \pi, \forall p \in pre(A), \exists T \in \pi, p \in pos(T) \wedge T <_p A$$

$$\wedge (\forall C \in \pi, C \neq A \wedge p \in (pre(C) - pos(C)) \Rightarrow (C <_p T \vee A <_p C))$$

This truth criterion presents some advantages over the ones presented previously. Although it does not contain the white knight declobbering technique, it does not contain separation either. Whereas McCluskey consider S=C and T=C as declobbering techniques, PWEAK's MTC does not. The case T=C is avoided by considering that an operator cannot both establish and clobber a proposition ($pos(O) \cap (pre(O) - pos(O)) = \emptyset$); S=C is not really a cure to the clobbering problem. Instead, it is just remarking that it is not a problem if one clobberer is the operator we try to assert: this case has then been integrated in the universal clause concerning clobberers. Figure 3 presents a graphic version of this MTC where A denotes McCluskey's S.

3.3 Procedural interpretation of the truth criterion

After theorem 1, in order to have a correct non linear plan (π, $<_p$) one must, for every operator A_i and for each $p \in pre(A_i)$:

1. Find an establisher of p before A_i, noted $T(p,A_i)$

2. Make sure that every clobberer of p are either before $T(p,A_i)$ or after A_i.

If all preconditions of all the operators fulfill conditions 1 and 2 above, the plan is correct. If not, two cases can happen:

First case: There exists no establisher T of p before A_i. To correct the plan, all we need is to find an operator T such that $p \in pos(T) \wedge T <_p A_i$; there are two possible choices for T:

• Either take an operator existing in the plan, such that $p \in pos(T)$ and such that we may have T before A_i (i.e. such that $A_i <_p T$ is not true). We then add the constraint $T <_p A_i$ if this is not already the true. We call this choice *establishment by an old operator*.

• Or find a new T such that $p \in pos(T)$ and add the constraint $T <_p A_i$. We call this choice *establishment by a new operator*. In this step, we can use operators templates with variables. In that case, the template is instantiated before use: we here removed the previous limitation.

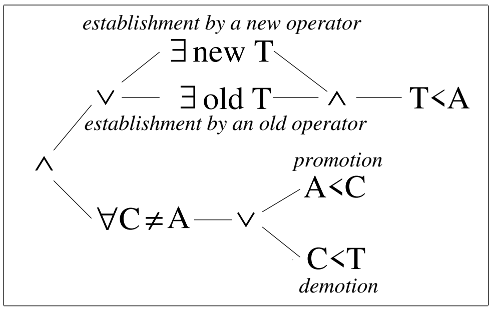

Figure 3: A graphic parsing of PWEAK's planning mechanism.

Second case: There exists an establisher T of p before A_i. However there exists a clobberer C of p which can be after T and before A_i (i.e. both $\neg(C <_p T)$ and $\neg(A_i <_p C)$ are true). To repair the plan one must do one of the following:

• Add the constraint $A_i <_p C$ if possible, i.e. if $C <_p A_i$ is not already true. Thus C is promoted after A_i. We call this *promotion*.

• Add the constraint $C <_p T$ if possible, i.e. if $T <_p C$ is not already true. This is the opposite of promotion, and it is called *demotion*.

• Change the choice of T, i.e. backtrack on the choice made for the establishment of p.

To use the four plan transformation operators above ((1) establishment by an old operator, (2) establishment by a new operator, (3) promotion and (4) demotion), PWEAK starts from the initial plan containing the initial operator O_i and the final operator O_f. Then, the plan is checked against the correctness criterion. If the criterion is satisfied, PWEAK stops with a correct plan. If not, the corresponding plan transformation operator is applied if possible and the process iterates. If an impossibility occurs, PWEAK backtracks. We have proved [6,7] that these plan modification operators are sufficient to find all minimal plans; i.e. plans without unnecessary operators. We first define minimal plans and then give the completeness theorem.

A non-linear plan PL is minimal if for every non linear plan Θ, Θ is included in Π implies $\Theta = \Pi$. $\Theta = (Q, <_Q)$ is included in $\Pi = (P, <_P)$ iff $\forall X,Y \in P, (X<_P Y \Rightarrow X<_Q Y)$.

Theorem 2 (completeness): Let σ_i be the initial state of a problem and σ_f its final state. If $(P, <_p)$ is a minimal non linear correct plan for the problem, then there exists a sequence of applications of the planning operators that achieve $(P, <_p)$, starting from the initial plan $((O_i, O_f), <_p)$.

Moreover, PWEAK always stops if there is a solution.

Corollary 1: PWEAK is complete and planning is semi-decidable.

Corollary 2: Planning is decidable if there is a finite set of constants describing the planning problem.

We end this section with a description of the algorithm of PWEAK. We give here two algorithms. The first algorithm is a mere translation of the truth criterion and the second one avoids scanning again all preconditions for each modification of the plan.

First algorithm:

0. Construct the initial plan $P=\{O_i, O_f\}$, with $O_i<_p O_f$.

1. Repeat for every $p \in pre(A)$ such that $A \in P$, as long as P is modified:

   a - Choose T in P, or instantiate an operator schema T to P such that $p \in pos(T)$ and $\neg(A<_p T)$, then add $T<_p A$ if this is not already true (establishment).
   If this is impossible then backtrack.
   b - For every $C \in P$ such that C clobbers p and $\neg(C<_p T)$ and $\neg(A<_p C)$, add $A<_p C$ (promotion) or $C<_p T$ (demotion).
   If this is impossible then backtrack.

In the second algorithm, ET is the set of already processed preconditions and EN is the set of preconditions to be processed. T(p,A) is the establisher chosen for precondition p of operator A.

Second algorithm:

0. Construct the initial plan $P=\{O_i, O_f\}$ with $O_i <_p O_f$;

ET := ∅; EN := σ_f.

1. Repeat a, b, c and d while EN≠∅:

a - Select p∈EN such that p∈*pre*(A) and A∈P
ET := ET ∪ {p}; EN := EN – {p};
b - Choose T(p,A) in P or instantiate an operator schema T(p,A) to P such that p∈*pos*(T(p,A)) and $\neg(A <_p T(p,A))$; then add $T(p,A) <_p A$ if not already true (establishment).
c - If T(p,A) is a new action (result of an instantiation) then add *pre*(T(p,A)) to EN and
for every q∈ET such that D∈P and q∈*pre*(D) and T(p,A) clobbers q, then add $T(p,A) <_p T(q,D)$ if it is possible (demotion of T(p,A)) or add $D <_p T(p,A)$ if it is possible (promotion of T(p,A)).
If it is impossible then backtrack.
d - For all C∈P such that C clobbers and C≠A and $\neg(C <_p T)$ and $\neg(A <_p C)$ then add $A <_p C$ (promotion of C) or $C <_p T$ (demotion of C).
If it is impossible then backtrack.

4 Discussion

What interestingness is there to reduce the non linear planning framework to the MTC presented in the previous section? First, it is always important to manipulate concepts that are minimal: it is then easier to build new ones above. Also, it ensures that derived concepts are correct. It also clearly shows that the white knight is pretty useless; this section provides other arguments about the misconception of white knight strategy .

4.1 Why a White Knight?

We try here to explain how the white knight strategy has been introduced.

Planning is usually performed in a backward chaining manner. Then building a plan amounts to asserting some goals backwardly until there is an error in the construction. There, backtracking is used to try other constructions. Of course, it is always possible to add redundant steps to a plan, but it is generally admitted that a solution to a given problem should not be too long. In some cases, a plan resulting of a planning algorithm had to be executed once, to ensure it was correcting leading to the final goal [8]. For example, there was some algorithm (e.g. [11]) to prove that a resulting plan was correct or not. Such algorithms execute forwardly plans that were built backwardly: they were used to determine the correctness of a plan. They consider that once a proposition (a goal) has been asserted (by an establisher), it must not be un-asserted (by a clobberer), unless there might be some new asserter (a white knight) after the clobberer. It is reasonable to construct an MTC on such a technique, but one must be aware it is forward based. From a backward point of view, we just need to backtrack until *the closest* establisher is found; this is what [1] and [2, page 361], [12, page 73] (in some way) are proposing.

So why a white knight? Chapman recognizes that [2, page 361] it falls out from the MTC and that “The only possible advantage is slightly less commitment, and so possibly less backtracking.” Although this may intuitively be true, the most important problem might be to calculate a white knight. Wilkins explains that [12, page 72]: “If there are *n* possible clobberers followed by *n* possible white knights, then there are n^2 ways to match them one-to-one (...). In fact there is a combinatorial explosion — one white knight might take care of a whole set of codesignating clobberers”.

Moreover, next section shows that the strategy is a source for looping.

4.2 Avoiding looping.

Chapman argues about TWEAK that (outcomes lemma [2, page 370]) success, failure or looping are the three possible outcomes of TWEAK's algorithm. Success or failure are unavoidable and desirable for decidability of the algorithm. But looping is certainly undesirable. Where could looping come from in a non linear planning algorithm like TWEAK's? Roughly, looping comes from the fact that neither success nor failure are detectable in problems where the truth criterion is always true although it should be false at some moment. This means that the planner keeps on constructing a plan for the problem although it is impossible. So the question becomes: “What is always true in the truth criterion?” The white knight is. It is always possible to insert a white knight after the clobberer and before the operator to be

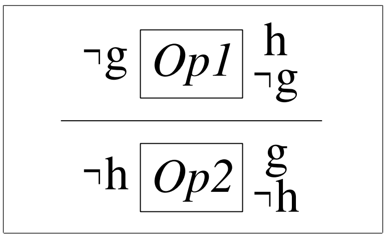

Figure 4: The operators for TWEAK's looping problem.

asserted. One just have to choose the new W (see Figure 1) as the previous clobbered establisher for the proposition. Therefore the white knight strategy is a source for looping. Chapman says [2, page 344] that a way to improve TWEAK is perhaps loop-detection. He then suggests to add heuristics or techniques for proving that no solutions exist. One sound technique is to get rid of the white knight technique.

To prove TWEAK's outcomes lemma, an example of a looping problem is given [2, page 370]. This problem involves two mutually exclusive operators (see Figure 4) and a goal that is reachable if and only if the two operators are

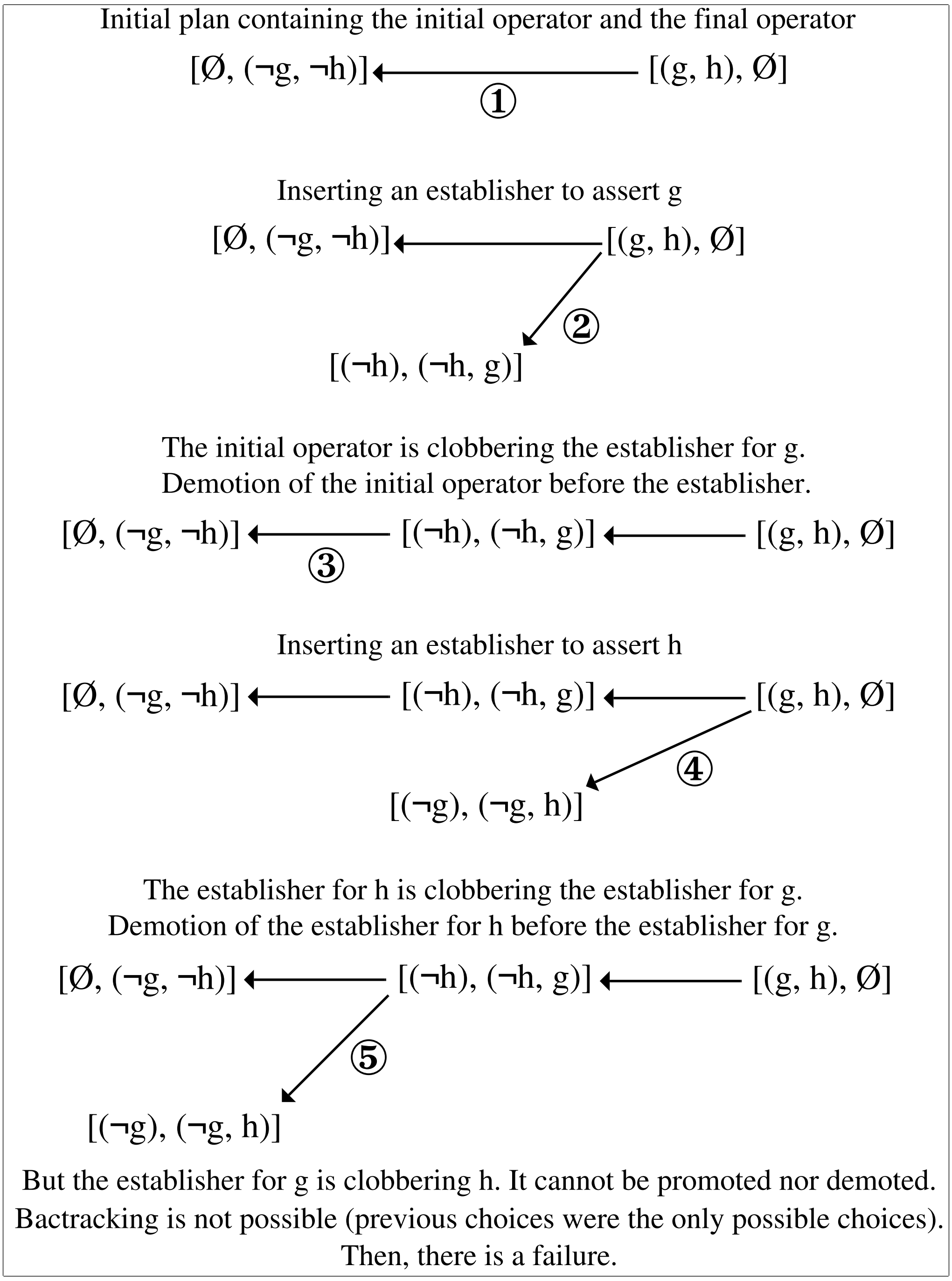

Figure 5: How PWEAK stops on TWEAK's looping.

applied. But this cannot be, since the operators are designed to clobber themselves mutually. And it is always possible to add again one of the two mutually exclusive operators as a new establisher (the white knight strategy) just before the operator whose preconditions we have to assert. This new instantiation also introduces a new clobberer, which creates looping. Figure 5 illustrates how PWEAK's algorithm stops on this problem since it possess no white knight. Figure 4 and 5 represent the looping problem as in Chapman's paper.

Chapman shows that planning with a STRIPS-like action representation is undecidable if and only if there is an infinite set of constants generating an infinity of initial situations, but it is clear that planning is decidable with a finite set of constants and a finite set of operators (since there is then a finite number of states). This is obviously the case for the looping problem stated in Figure 5. Therefore, it is possible to build a planning mechanism which stops on every finite problem. As Chapman claims TWEAK does not report failure but loops, it is not clear how TWEAK is complete on this problem.

5 Concluding remarks

In this section, we briefly discuss the usefulness of planning from PWEAK's minimal formalization. Does PWEAK bring something else than some looping avoidance to non linear planning?

Minimality is an important concept. It can now be easier to build some neat concept over this framework. For instance, it could be easier to develop an ABPWEAK or a Meta-PWEAK. ABPWEAK is surely possible since ABTWEAK has been nicely formalised [13]. Meta-PWEAK means to use a copy of PWEAK to control (plan) the nondeterministic goal-achievement procedure of PWEAK. Unfortunately, this is not possible since the STRIPS-based operator representation used in PWEAK is not able to represent a goal generated by PWEAK's MTC (the operator representation should need to represent universal quantifier and disjunction).

Acknowledgements

The authors thank Claude Le Pape and Pierre Regnier for their remarks on earlier drafts of this paper, Jim Doran for his help on IPEM, Sam Steel for his help on McCluskey's work, Austin Tate for his extended version of [11]. Thanks to Yves Kodratoff and his group at LRI Orsay, Jacques Ferber and his group at Laforia Paris 6 and Patrick Albert for their support.

References

[1] Jose Ambros-Ingerson and Sam Steels, *Integrating Planning, Execution and Monitoring*, Proceedings of AAAI-88 (1988), pp. 83-88.

[2] David Chapman, *Planning for conjunctive goals*, Artificial Intelligence **32** (1987), pp. 333-377.

[3] Richard Fikes and Nils Nilsson, *STRIPS: A new approach to the application of theorem proving to problem solving*, Artificial Intelligence **2** (1971), pp. 189-208.

[4] Thomas Leo McCluskey, *Experience-Driven Heuristic Acquisition in General Problem Solvers,* Ph.D. Thesis, The City University, December 1988.

[5] Thomas Leo McCluskey, *Deriving a Correct Logic Program from the Formal Specification of a Non-Linear Planner*, Proceedings of the Third International Symposium on Intelligence Systems, North-Holland, October 1988. Also Appendix A5 in [4].

[6] Jean-François Puget, *Apprentissage par explications d'échecs ; théories et applications*, Ph.D. Thesis, LRI, January 90. (In French).

[7] Jean-François Puget, *PWEAK: A Rational Reconstruction of TWEAK*, November 1990. Submitted to the European Workshop on Planning.

[8] Stanley Rosenschein, *PLAN SYNTHESIS: A Logical Perspective*, Proceedings of IJCAI-81 (1981), pp. 331-337.

[9] Earl Sacerdoti, *The Non Linear Nature of Plans*, Proceedings of IJCAI-75 (1975), pp. 206-214.

[10] Austin Tate, *Generating Project Networks*, Proceedings of IJCAI-77, (1977) pp. 888-893.

[11] Austin Tate, *Goal Structure — Capturing the intent of plans*, Proceedings of ECAI-84 (1984), pp. 273-276. Also (extended version) AIAI-TR-3, University of Edinburgh, Artificial Intelligence Applications Institute.

[12] David Wilkins, *Practical Planning — Extending the Classical AI Paradigm*, Morgan Kaufmann (1988), 205 pages.

[13] Qiang Yang and Josh D. Tenenberg, *ABTWEAK: Abstracting a Nonlinear, Least Commitment Planner*, Proceedings of AAAI-90 (1990), pp.204-209.

Appendix

This appendix contains some comments about an introduction of [2]; this introduction is on page 521 of the book "Readings in Planning", by James Allen, James Hendler and Austin Tate. These comments have been sent to the authors of "Reading in Planning".