Enabling-Condition Interactions and Finding Good Plans*

Dana S. Nau†
University of Maryland
College Park, Maryland 20742
nau@cs.umd.edu

Overview

In AI planning research, the best-known goal interaction is the deleted-condition interaction, in which the side-effect of achieving one condition is to delete some other condition that will be needed later. Enabling-condition interactions—in which the side-effect of achieving one condition is to make it easier to achieve some other condition—are not as well known. In this paper, I argue that enabling-condition interactions merit more attention than they have heretofore received.

This paper is organized as follows:

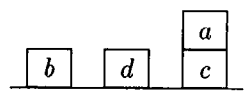
1. A definition of enabling-condition interactions.
2. The importance of finding "good" plans (rather than simply being satisfied with any plan we find), with examples from several planning domains.
3. How a number of planning strategies take advantage of enabling-condition interactions to produce better plans.
4. Some of the effects of enabling-condition interactions on the complexity of planning.
5. Concluding remarks.

Definition

An *enabling-condition interaction* is a situation in which some action invoked to achieve one goal G_1 also makes it easier to achieve another goal G_2. For example, in Figure 1, the action move(a, c, b) achieves the goal on(a, b), but it also has the side-effect of clearing c, making it easier to achieve the goal on(c, d). As another example, consider the following situation (based on (Wilensky, 1983)):

> John lives two miles from a bakery and two miles from a dairy. The two stores are one mile apart. John has two goals: to buy bread and to buy milk.

Initial state: $I =$
$\{\text{clear}(b), \text{on}(b, \text{TABLE}),$
$\text{clear}(d), \text{on}(d, \text{TABLE}),$
$\text{clear}(a), \text{on}(a, c),$
$\text{on}(c, \text{TABLE})\}$



Goal formula: $G =$
$\{\text{on}(a, b), \text{on}(c, d)\}$

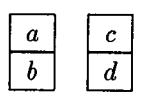


Figure 1: In this problem, moving a to b enables us to move c to d.

If John goes to the bakery to buy bread, then this puts him closer to the dairy, making it easier for him to buy milk.

Finding "Good" Plans

Perhaps the most obvious case in which enabling-condition interactions affect planning is if we want to find a "good" plan (i.e., one whose length or cost is small, or whose efficiency is high) rather than being satisfied with any plan (no matter how inefficient) that achieves the goal.[1] Until recently, the objective of finding "good" or "optimal" plans does not appear to have received much explicit discussion in the AI planning literature. However, it appears to have been an underlying motivation behind a number of existing planning strategies.

For example, Fig. 2 shows the well known "Sussman anomaly" of blocks-world planning (Waldinger, 1990, p. 127) (Sussman, 1975). The primary reason why this problem generated so much interest was that the best plan that previously existing planning procedures could generate for this problem was

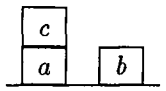
> move(b, TABLE, c), move(b, c, TABLE),
> move(c, a, TABLE), move(b, TABLE, c),
> move(a, TABLE, b).

*This work was supported in part by NSF Grants IRI-8907890 and NSFD CDR-88003012.

†Department of Computer Science, Systems Research Center, and Institute for Advanced Computer Studies.

[1] Actually, enabling-condition interactions affect planning even if we are not concerned with the goodness of a plan. I hope to discuss this more fully in a subsequent paper.

Initial state: $I =$
$\{\text{clear}(c), \text{clear}(b),$
$\text{on}(a, \text{TABLE}), \text{on}(c, a),$
$\text{on}(b, \text{TABLE})\}$



Goal formula: $G =$
$\{\text{on}(a, b), \text{on}(b, c)\}$

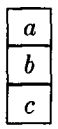


Figure 2: The Sussman anomaly.

This plan does achieve the desired goal, but it does so inefficiently. If we did not care about the goodness of the plan, then the Sussman anomaly would be of no concern.

In addition, there are a number of practical planning situations in which it is important to find good plans. Below are several examples.

Machining-Operation Sequencing

Consider the problem of using machining operations to make holes in a metal block. Several different kinds of hole-creation operations are available (twist-drilling, spade-drilling, gun-drilling, etc.), as well as several different kinds of hole-improvement operations (reaming, boring, grinding, etc.). Each time one switches to a different kind of operation or to a hole of a different diameter, one must mount a different cutting tool on the machine tool. If the same machining operation is to be performed on two different holes of the same diameter, then these two operations can be merged by omitting the task of changing the cutting tool. Thus, by performing the two operations at the same time, both are made easier. This and several other similar manufacturing problems are of practical significance (Chang and Wysk, 1985; Hayes, 1987; Nau, 1987; Nau *et al.*, 1988; Mantyla and Opas, 1988).

Logistics Planning

Different plans containing deliveries to the same place may be combined into a single trip, saving considerable expense. In addition, an inexpensive delivery technique needed for one delivery can be subsumed by one with a greater cost in a second plan, while still allowing for an overall savings. For example, if a cargo of food could be delivered to a location via a parachute drop, but some other piece of equipment going to the same site requires a landing, then both items can be delivered on the same plane (the one that lands) and the extra steps required for the parachute drop can be deleted.

Scheduling

Consider the problem of finding a schedule for satisfying some set of orders for products that can be produced in a machine shop. For each order, there may be a set of alternative schedules for producing it, and each such schedule consists of a set of operations to be performed on various machines. An operation in a schedule is usually associated with a machine for carrying it out. If two or more operations require the same type of set-up, then doing them on the same machine may reduce the total time required—and thus reduce the total time required to complete all the schedules.

Multiple Database-Query Optimization

Let $Q = \{Q_1, Q_2, \ldots, Q_n\}$ be a set of database queries. Associated with each query Q_i is a set of alternate access plans $\{P_{i1}, P_{i2}, \ldots, P_{ik_i}\}$. Each plan is a set of partially ordered tasks that produces the answer to Q. For example, one task might be to find all employees in some department whose ages are less than 30, and whose salaries are over $50,000. Each task has a cost, and the cost of a plan is the sum of the costs of its tasks. Enabling-condition interactions occur if plans for two different tasks contain the same query, or if the result of evaluating one task reduces the cost of evaluating the other. A better overall plan can be produced by taking advantage of these interactions, and several research papers have been written on this topic (Sellis, 1988; Shim *et al.*, 1991).

Planning Strategies

A number of planning strategies have been formulated to exploit enabling-condition interactions in order to produce better plans. Below are two examples.

Studies of human planning behavior show that people look for enabling-condition interactions when they are formulating plans, in order to make the plans more efficient. For example, consider the following excerpt from Hayes-Roth and Hayes-Roth's transcript of someone "thinking aloud" while planning a hypothetical day's errands (Hayes-Roth and Hayes-Roth, 1990, p. 254):

> In section 6, the subject asks, "What is going to be the closest one?" This question indicates a strategic decision to plan to perform the closest errand next in the procedural sequence ...

This planning strategy is one that Pollack (Pollack, 1992) has referred to as *overloading*: if you notice that an action you have selected in order to accomplish one goal also makes it easier to accomplish another goal, then use that action for both of these purposes.

The same basic idea has been incorporated into nonlinear planning systems, where it is sometimes called *phantomization*: SIPE (Wilkins, 1988), Nonlin (Tate, 1977), and Kambhampati's plan reuse framework (Kambhampati, 1990) are capable of recognizing an operator in the current plan satisfies another goal in addition to the one it was originally intended to achieve, and imposing constraints on the plan so that this operator will be used to achieve both goals.

a is in d's way and d is in a's way, so $\{a, d\}$ is deadlocked.

a is in its own way, so $\{a\}$ is deadlocked.

Initial state: $I = \{\text{clear}(a), \text{on}(a, b), \text{on}(b, c), \text{on}(c, \text{TABLE}), \text{clear}(d), \text{on}(d, e), \text{on}(e, \text{TABLE})\}$

Goal formula: $G = \{\text{on}(a, e), \text{on}(e, b), \text{on}(d, c)\}$

Figure 3: A problem in which two sets of blocks are deadlocked: $\{a, d\}$ and $\{a\}$.

A different but related strategy is *action merging* (Yang *et al.*, 1990; Foulser *et al.*, 1992; Yang *et al.*, 1993). In this case, rather than simply recognizing that an existing action accomplishes an additional goal, the planner replaces an action that doesn't accomplish an additional goal with one that does (provided that this produces a lower total cost than would be incurred by using a separate action to achieve the additional goal). NOAH's "eliminate redundant precondition" critic (Sacerdoti, 1977) is a special case of this strategy.

Effects on Complexity

If a problem contains more than one enabling-condition interaction, then it can be difficult to determine which set of actions will produce the best plan. For example, if actions A and B both achieve goal G_1, and A also aids in achieving goal G_2, then we might prefer action A to action B—but if B also aids in achieving goal G_3, then it may no longer be clear which of A and B we should prefer. The difficulty of resolving such tradeoffs is illustrated in some of the repeated revisions that Hayes-Roth and Hayes-Roth's subject makes to his plan (Hayes-Roth and Hayes-Roth, 1990, p. 246–247). Below, I discuss several other cases where enabling-condition interactions increase the difficulty of planning.

Blocks World (The Usual Formulation)

The effect of enabling-condition interactions on blocks-world planning was analyzed formally in (Gupta and Nau, 1992). This paper showed that in the blocks world, finding an optimal plan is NP-hard—and that the NP-hardness is not due to deleted-condition interactions such as the Sussman anomaly, but instead due to a particular kind of enabling-condition interaction called a deadlock. For problems that do not contain deadlocks, there is a simple hill-climbing strategy that can easily find an optimal plan, regardless of whether or not the problem contains any deleted-condition interactions.

To see that deadlocks are a special case of enabling-condition interactions, consider the definition of a deadlock given in (Gupta and Nau, 1992). The set of blocks $\{b_1, b_2, \ldots, b_p\}$ is *deadlocked* in the state S if there is a set of blocks $\{d_1, d_2, \ldots, d_p\}$ such that the following three conditions hold:

1. In S, b_i is above d_i for $i = 1, 2, \ldots, p$.
2. In G, b_i is above d_{i+1} for $i = 1, 2, \ldots, p-1$, and b_p is above d_1.
3. In S, none of $b_1, b_2, \ldots, b_p$ are in their final positions (if $p > 1$, then the other two conditions entail this condition).

For example, in Figure 3, in the initial state I there are two deadlocked sets of blocks:

1. In I, a is above c and d is above e. In G, a is above e and d is above c. Thus $\{a, d\}$ is deadlocked.
2. a is above b in both I and G, and a is not in its final position in I. Thus $\{a\}$ is deadlocked.

Suppose some set of blocks $D_1 = \{a, b\}$ is deadlocked. Then before we can move a and b to their final positions, we must resolve the deadlock by moving either a or b out of the way. Now, suppose that a is also in some other deadlocked set D_2, and b is also in some other deadlocked set D_3. Then there are two enabling-condition interactions: moving a out of the way will also resolve the deadlock D_2, and moving b out of the way will also resolve deadlock D_3. Since these two possible moves will have side-effects of making different sets of goals easier to achieve later on, it is unclear which of of the two moves we should prefer. As shown in (Gupta and Nau, 1992), in the general case of this problem it is NP-hard to find an optimal plan.

Table 1: Complexity of domain-independent planning. These results are independent of whether or not conditional operators are allowed.

Lang. restrictions	How the operators are given	Allow delete lists?	Allow negated preconditions?	Telling whether a plan exists	Telling whether there is a plan of length $\leq k$	Line number
no function symbols; finitely many constant symbols	given in the input	yes	yes/no	EXPSPACE-comp.	NEXPTIME-comp.	(1)
		no	yes	NEXPTIME-comp.	NEXPTIME-comp.	(2)
			no	EXPTIME-comp.	NEXPTIME-comp.	(3)
			no$^{\alpha}$	PSPACE-complete	PSPACE-comp.	(4)
	fixed in advance	yes	yes/no	PSPACE $^{\gamma}$	PSPACE $^{\gamma}$	(5)
		no	yes	NP $^{\gamma}$	NP $^{\gamma}$	(6)
			no	P	NP $^{\gamma}$	(7)
			no$^{\alpha}$	NLOGSPACE	NP	(8)
all predicates are 0-ary (propositions)	given in the input	yes	yes/no	PSPACE-comp.$^{\delta}$	PSPACE-comp.	(9)
		no	yes	NP-complete$^{\delta}$	NP-complete	(10)
			no	P $^{\delta}$	NP-complete	(11)
			no$^{\alpha}$/no$^{\beta}$	NLOGSPACE-comp.	NP-complete	(12)
	fixed	yes/no	yes/no	constant time	constant time	(13)

$^{\alpha}$No operator has more than one precondition.
$^{\beta}$Every operator with > 1 precondition is the composition of other operators.
$^{\gamma}$With PSPACE- or NP-completeness for some sets of operators.
$^{\delta}$Results from (Bylander, 1991).

Blocks World (Another Formulation)

(Chenoweth, 1991) describes a more complicated version of blocks world planning, in which more than one block can have the same name. It is an open question how difficult deleted-condition interactions are in Chenoweth's version of the blocks world—but Chenoweth's NP-hardness proof for his domain again shows that enabling-condition interactions make the problem NP-hard.

Chenoweth's proof of NP-hardness is by reduction from 3SAT. Given a 3SAT problem with m clauses and n variables, he generates an MPBW problem in which $L = 3n + 5m + 1$. For each i ($i = 1, \ldots, n$), there are two blocks named u_i, at the tops of two large stacks. For each i, one of the two u_i's must be moved to the top of a block named v_i, and the question is which u_i to move. If we move the wrong one, then later in the plan, we will have to move one or more blocks temporarily to the table rather than moving them directly to their final positions, whence the resulting plan will be longer than L.

The above problem is similar to the problem of resolving multiple deadlocks in the more usual formulation of the blocks world. In both problems, if we make the wrong choice, then too many blocks must be moved temporarily to the table rather than directly to their final positions. However, the two problems are not identical. If no two blocks have the same name, then for a wrong choice to force us to move extra blocks to the table, we must have blocks which mutually block each others' progress—and this led to our definition of deadlock. But if more than one block can have the same name, then one can find other ways for a wrong choice to force us to move extra blocks to the table—and that is what Chenoweth did.

Domain-Independent Planning

Enabling-condition interactions are important not only in the blocks world, but in domain-independent planning in general. As an example, consider Table 1, which was taken from (Erol *et al.*, 1991; Erol *et al.*, 1992). This table shows how the complexity of domain-independent planning with STRIPS-style operators depends on the nature of those operators.

In lines (11) and (12) of Table 1, plan existence can be determined in polynomial time, but the problem of finding an optimal plan is NP-complete. The reason for this is as follows. In these cases, if we are simply interested in finding a plan but do not care how good a plan it is, the restrictions allow us to plan for each subgoal separately, using backwards chaining. But if we are interested in a short plan (i.e., one of length $\leq k$ for some k) rather than just any plan, then we need to pay attention to enabling-condition interactions, because they make it possible to produce a shorter plan. It is not possible to detect and reason about these interactions if we plan for the subgoals independently; instead, we have to consider all possible operator choices and orderings, making it NP-hard to tell whether there is a plan of length $\leq k$.

Note also that throughout Table 1, whether or not negated preconditions are allowed does not affect the complexity of telling whether there is a plan of length $\leq k$. Again, what makes these particular problems difficult is how to handle multiple enabling-condition interactions—more specifically, how to choose opera-

tors that achieve several subgoals in order to minimize the overall length of the plan. For these problems, this task remains equally hard regardless of whether negated preconditions are allowed.

Conclusion

In AI planning research, the best-known goal interaction is the deleted-condition interaction, in which the side-effect of achieving one condition is to delete some other condition that will be needed later. Enabling-condition interactions are not as well known. In this paper, I have argued that enabling-condition interactions merit more attention, for the following reasons:

> The goal of finding good plans (rather than simply being satisfied with any plan we find) is important in many planning situations, and has been an implicit motivation behind several existing planning strategies. But if we are interested in finding good plans rather than poor plans, enabling-condition interactions can dramatically increase the difficulty of planning.

In addition, preliminary results suggest that even if we don't care about the quality of the plan we find, enabling-condition interactions can still dramatically increase the difficulty of planning. If this result is correct, then it suggests that enabling-condition interactions are just as important in planning as deleted-condition interactions. I hope to investigate this issue more fully in the future.

References

Bylander, Tom 1991. Complexity results for planning. In *IJCAI-91*.

Chang, T.C. and Wysk, R. A. 1985. Integrating cad and cam through automated process planning. *International Journal of Production Research* 22(5).

Chenoweth, Stephen V. 1991. On the NP-hardness of blocks world. In *AAAI-91: Proc. Ninth National Conf. Artificial Intelligence*. 623–628.

Erol, K.; Nau, D.; and Subrahmanian, V. S. 1991. Complexity, decidability and undecidability results for domain-independent planning. Submitted for publication.

Erol, K.; Nau, D.; and Subrahmanian, V. S. 1992. On the complexity of domain-independent planning. In *Proc. AAAI-92*. 381–386.

Foulser, David; Li, Ming; and Yang, Qiang 1992. Theory and algorithms for plan merging. *Artificial Intelligence* 57(2-3):143–182.

Gupta, Naresh and Nau, Dana S. 1992. On the complexity of blocks-world planning. *Artificial Intelligence* 56(2-3):223–254.

Hayes-Roth, B. and Hayes-Roth, F. 1990. A cognitive method of planning. In Allen, James; Hendler, James; and Tate, Austin, editors 1990, *Readings in Planning*. Morgan Kaufman. 245–262. Originally appeared in *Cognitive Science* 3(4), 1979.

Hayes, Caroline 1987. Planning in the machining domain: Using goal interactions to guide search. Master's thesis, Carnegie-Mellon University, The Robotics Institute, Pittsburgh, PA.

Kambhampati, Subbarao 1990. A theory of plan modification. In *AAAI-90*.

Mantyla, Martti and Opas, Jussi 1988. Hutcapp—a machining operations planner. In *Proc. Second International Symposium on Robotics and Manufacturing Systems*.

Nau, D. S.; Karinthi, R.; Vanecek, G.; and Yang, Q. 1988. Integrating AI and solid modeling for design and process planning. In *Second IFIP Working Group 5.2 Workshop on Intelligent CAD*, Cambridge, UK. University of Cambridge.

Nau, D. S. 1987. Automated process planning using hierarchical abstraction. *TI Technical Journal* 39–46. Award winner, Texas Instruments 1987 Call for Papers on AI for Industrial Automation.

Pollack, Martha 1992. The uses of plans. *Artificial Intelligence* 57(1):43–68.

Sacerdoti, E. D. 1977. *A Structure for Plans and Behavior*. American Elsevier Publishing Company.

Sellis, T. 1988. Multiple-query optimization. *ACM Transactions on Database Systems* 13(1):23–52.

Shim, K.; Sellis, T.; and Nau, D. 1991. Improvements on a heuristic algorithm for multiple-query optimization. Submitted for publication.

Sussman, G.J. 1975. *A Computational Model of Skill Acquisition*. American Elsevier, New York.

Tate, Austin 1977. Generating project networks. In *Proc. 5th International Joint Conf. Artificial Intelligence*.

Waldinger, R. 1990. Achieving several goals simultaneously. In Allen, James; Hendler, James; and Tate, Austin, editors 1990, *Readings in Planning*. Morgan Kaufman. 118–139. Originally appeared in *Machine Intelligence 8*, 1977.

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

Wilkins, David E. 1988. *Practical Planning: Extending the Classical AI Planning Paradigm*. Morgan-Kaufmann Publishers, Inc., San Mateo, CA.

Yang, Q.; Nau, D. S.; and Hendler, J. 1990. Optimization of multiple-goal plans with limited interaction. In *Proc. DARPA Workshop on Innovative Approaches to Planning, Scheduling and Control*.

Yang, Q.; Nau, D. S.; and Hendler, J. 1993. Merging separately generated plans with restricted interactions. *Computational Intelligence* 9(1). To appear.