# Preconditions and appropriateness conditions*

**Timothy M. Converse** and **Kristian J. Hammond**
Artificial Intelligence Laboratory
Computer Science Department
University of Chicago
1100 E. 58th St.
Chicago, IL 60637
(312) 702-8584
converse@cs.uchicago.edu

## Abstract

Classical plan preconditions implicitly play a dual role, both documenting the facts necessary for a plan to be sound and listing the conditions under which it should be used. As the closed-world assumption is relaxed these two roles begin to diverge, particularly when attempts are made to use plans in situations other than those for which they were originally constructed. Rosenschein and Kaelbling exploit one aspect of the divergence by suggesting that some logical preconditions can be considered in the design phase of building an agent, but "compiled away" so that the agent need not explicitly consider them [Rosenschein and Kaelbling, 1986]. We suggest an alternative view whereby an agent can explicitly reason and learn about which conditions are the best cues for employing standard plans, and discuss the idea in the context of the RUNNER project.

## Introduction

### Plan execution and the role of preconditions

The closed-world assumption of classical planning allowed the assumption that plans could have a small and explicit set of preconditions which, if true, would ensure that the plan worked. The most straightforward application of this idea to the execution of such plans is that an executor should know or verify the truth of the preconditions before starting the plan. This is particularly unproblematic if, in addition to a closed world assumption, the plan is being generated for exactly the circumstances in which it is to be used; in that case, everything that is known about the state of the world can be taken into account during the synthesis of the plan, and the process of construction can (given the closed-world assumption) itself guarantee the soundness of the plan.

In recent years, greater awareness of the intractability of generative planning [Chapman, 1985], coupled with greater concern about time pressure in activity, has led to attempts to amortize the cost of planning over repeated instances of activity, either by "pre-compiling" action decisions [Rosenschein and Kaelbling, 1986, Drummond, 1989], or by re-using the fruits of previous planning attempts [Hammond, 1989].

At the same time it has been widely acknowledged that the set of logical preconditions for plans in many real-world situations is effectively infinite. This been called the "qualification problem" (defined variously in [McCarthy, 1977, Shoam, 1986, Ginsberg and Smith, 1987]). That is, given any attempt at enumeration of logical statements that need to be true for a given plan to be guaranteed to work, it is usually possible to come up with an additional potential fact that would render the plan unworkable.

Early planning research tried to confront this directly using large numbers of frame axioms. Most generative planning systems since [Fikes and Nilsson, 1971] have used the more optimistic and tractable STRIPS assumption that primitive actions can have associated lists of the facts that are changed by applying them.

Precondition sets for classical plans implicitly play a dual role. They

1. describe the initial conditions under which the plan as described can be expected to be sound

*This work was supported in part by AFOSR grant number AFOSR-91-0112, DARPA contract number F30602-91-C-0028 monitored by Rome Laboratories, DARPA contract number N00014-91-J-4092 monitored by the Office of Naval Research, Office of Naval Research grant number N00014-91-J-1185

(under certain assumptions, and because the facts were used in the plan's construction), and

2. describe the facts that an executor should know to be true before beginning execution of the plan.

Let us keep the existing term of *precondition* for the first sort of fact above, and use the term *appropriateness condition* for the second sort.

### Appropriateness conditions

Under assumptions of perfect knowledge and a closed world, there is little divergence between preconditions and appropriateness conditions. When these assumptions are relaxed, however, there are several different ways in which the divergence can become important in plan execution and reuse:

- A precondition can effectively be "always true". This means that the plan may depend upon it for correctness, but an executor will never run into trouble by not worrying about its truth value. This sort of fact should not be an "appropriateness condition", since consideration of it cannot help in the decision whether to use the plan.
- A precondition may be almost always true, and it may be difficult to know or check in advance. If the consequences of an abortive attempt at performing the plan are not too severe, then this sort of fact should not be an appropriateness condition, since the utility of knowing its truth is outweighed by the cost of acquiring the knowledge.
- A precondition may be intermittently true, but may be easily "subgoaled on" in execution, and achieved if false. (This of course depends strongly on representation of plans, and how flexible the execution is.) To the extent this can be handled in "execution", the condition should not be an appropriateness condition, since whether or not the condition holds the plan is likely to succeed.
- A particular condition may not be a precondition *per se*, but may be evidence that the plan will be particularly easy to perform, or will produce results that are preferable to the usual default plan for the relevant goals. This *should* be an appropriateness condition, even though it is not a precondition.

### Action nets and appropriateness conditions

Rosenschein and Kaelbling noted the first possibility in the above list, that some preconditions might be "always true", and realized that, while such facts may need to be explicitly considered in the design of an agent for some domain and task, there is no reason why the agent itself need consider them. [Rosenschein and Kaelbling, 1986]. Such facts can essentially be "compiled away" in the design of an agent that will behave appropriately.

To summarize the argument so far:

- The set of facts that an agent should consider before embarking on a given plan is interestingly different from both the (possibly infinite) set of facts that need to be true for the plan to work, and the set of facts explicitly used in the plan's construction. This is true particularly when *de novo* plan construction is impossible or too costly, and plans must be reused.
- One possible approach that recognizes this is to explicitly design an agent so that it only considers the conditions that are actually relevant for action, either by hand-crafting its decision procedure, or by a mixture of hand-crafting and clever compilation of declarative specifications, as in Rosenschein and Kaelbling's work.

And the point we want to make (which will occupy us for the rest of the paper):

There is a large potential middle ground between an approach that requires explicit reasoning about all preconditions on the one hand, and approaches that compile in any needed reasoning of that sort in advance. In particular, even if an agent is assumed to have a largely immutable library of plans and behaviors that will determine its competence, there is still room for learning the particular appropriateness conditions that govern when to invoke particular plans.

## An example

To make these distinctions clearer, let's look at a common sense example: the task of making buttered toast, in a well-equipped kitchen, with an electric toaster.

If we start to enumerate the preconditions that we can think of that are associated with this task, the most available ones have to do with the resources we would normally worry about it conjunction with it: possessing bread, possessing butter. Others that come to mind might have to do with available time, instruments (the toaster, a knife), or knowledge about these things (do we know where to find a knife?). As we strain to think of things that are not part of the concerns associated with the plan, we might think of possible "external" problems like an interruption in electric service. Finally, imaginable "preconditions" start to be explicitly counterfactual; what if gravity no longer operated, or heat conduction worked in a different way?

In practice, when deciding whether to make toast, one is probably aware of only the first few

considerations, and possibly none are a concern. The technical proposal we would like to make here is that we should trust the introspective availability of these conditions: there is a small set of facts that should be explicitly considered when deciding whether to embark on a given plan. In addition, we argue that these appropriateness conditions should be stored in association with the plan itself, and that learning to refine them is a significant part of the development of expertise in plan use.

## The utility of partial plan completion

Agre and Chapman [Agre and Chapman, 1988] have argued against the need by action systems for "plans" in the classical sense, and have pointed out the dangers of confusing the technical and commonsense meanings of the term. The Pengi system [Agre and Chapman, 1987] demonstrated a surprising capacity to exhibit "planful" behavior without explicit representation of plans. What this means is that sequences of actions that were particularly beneficial could be triggered by successive perceptual conditions, without the sequence itself being represented internally in any way.

A good analysis of the sources of domain support for Pengi's behavior can be found in [Chapman, 1990]. An additional one we would like to suggest is that, although the domain rewards successful completion of certain sequences of actions, it does not particularly penalize partial completion. The domain we are investigating in the RUNNER project [Hammond *et al.*, 1990] (performing simple tasks in a simulated kitchen) has the characteristic that many tasks will leave the agent in worse shape if partially completed than if the task had never been started. For example, many tasks require that milk be taken out of the refrigerator, but if that is all that is done, the only result will be sour milk. Note that this contrast is at the level of *domain* analysis, and is orthogonal to the question of how "planful" behavior is generated. We suspect that domains that have this sort of non-additiveness of utility over the course of action sequences may require *some* sort of explicit plan representation from their agents.

So far we have used the term "plan" in a common sense way. What we would like to mean by it is this: the collection of explicitly represented knowledge that is specifically relevant to repeated satisfaction of a given set of goals, and which influences action only when a decision has been made to use the plan. In the RUNNER project this explicitly represented plan serves both as a memory organization point for annotations about the current progress and problems of the use of the plan, and as a hook on which to hang past experiences of its use. Its appropriateness conditions determine whether it should be "active" and hence make suggestions about actions, but it is not the sole determinant of behavior.

## A taxonomy of attitudes toward preconditions

Given a plan that requires that certain propositions be true for it to work, there are a limited number of *attitudes* that an agent can take toward those propositions when deciding on whether to employ the plan. Whether explicitly or implicitly, given a certain precondition, the agent can

1. Assume it is always true (because it is always true).

   For example, the assumption of continued gravity in the toast example.

2. Assume it is always true (because the agent *enforces* it).

   For example, the toaster will not work if the power is not on, and the power may not be on if the electric bill is never paid. Nonetheless, most people are probably not aware of thinking of their electric bill when considering making toast. This is because other plans and habits ensure that it is always paid, and therefore electricity does not figure into the appropriateness conditions for making toast. Depending on the extent to which larder-stocking is taken care of by other plans and policies, possession of bread and butter may also be omitted. (For more discussion of the use of enforcement to simplify plan use, see [Hammond and Converse, 1991].)

3. Verify it before plan execution.

   If unbuttered toast is worse than no toast at all, and there is doubt as to the presence of butter in the refrigerator, then a good toast-making plan includes looking in the refrigerator first, and aborting the plan if none is found.

4. Know it to be true.

   "World models" should not be invoked without recognition of the cost of their construction and maintenance, and the cost of memory retrieval of a given fact (even if explicitly represented) may often outweigh the cost of active perceptual verification. Still, in our example, the question of bread and butter could be raised and then answered, for example, by memory of a recent shopping trip.

5. Subgoal on it if necessary.

   If the toaster is found to be unplugged, then it may be easy to plug back in and continue. If this particular state is unpredictable, there still may be no need to include mention of it in the appropriateness conditions for making toast.

We want to include conditions covered by items (3) and (4) above under our rubric of appropriateness conditions; they are the facts that should concern the agent in deciding on the viability of a plan.

## Appropriateness Conditions in RUNNER

Our work in the RUNNER project centers around plan use in a commonsense domain. Starting from a case-based planning framework [Hammond, 1989], we believe that an appropriate view of expertise development in many domains is the acquisition and refinement of a library of plans which have been incrementally debugged and optimized for the sets of conjunctive goals that typically recur.

One of the goals of the research is to use plans *flexibly*, where flexibility means that

- Step orderings can be suggested by environmental cues. Where not explicitly constrained, multiple steps can simultaneously suggest actions.
- Particular specializations of plans are environmentally cued.
- Multiple "top-level" plans can be active and each suggest actions, resulting in (unrepresented) interleaving.

The representation structure for RUNNER 's memory, as well as the bulk of the algorithm for marker-passing and activation, is based on Charles Martin's work on the DMAP parser (see [Martin, 1989]). The memory of RUNNER 's agent is encoded in semantic nets representing its plans, goals, and current beliefs. Each node in RUNNER 's plan net has associated with it a (disjunctive) set of *concept sequences*, which are a (conjunctive) listing of states that should be detected before that plan node can be suggested.

Nodes in the plan net become activated in the following ways:

- "Top-level" plans become activated when the goal they subserve is activated, and a concept sequence indicating appropriate conditions is completed.
- Specializations of plans are activated by receiving a *permission marker* from the abstract plan, in addition to the activation of a concept sequence.
- Parts (or steps) of plans are also activated by completion of a concept sequence, and by receiving a permission marker from their parent.

Once activated, many plans have early explicit verification steps which check if other conditions necessary for success are fulfilled, and abort the plan if not.

Passing of permission markers is not recursive, so that the state information indicating an opportunity to perform a sub-plan must be recognized for execution to proceed further. This means that individual subplans must have associated with them concept sequences that indicate opportunities to be performed. (For a fuller explication, see [Hammond *et al.*, 1990].

Appropriateness conditions in RUNNER, then are the union of the concept sequences of plan nodes and the initial verification steps (if any). Since activation of a plan node is sufficient to send activation to its parts and specializations, which can in turn eventually bottom out in primitive actions, it is important that these conditions be well-matched to the conditions under which it is appropriate to invoke the plan.

## Implications for Learning

On our view, a large component of expertise in complex domains is the result of the development of a library of conjunctive goal plans, with the simultaneous tuning of the plans, their appropriateness conditions, and the environment itself to maximize the effectiveness of the plans.

Our position is that the state an agent should strive for is one in which

1. its plan library has optimized plans for the different sets of goals that typically recur.
2. each plan (and subplan) has an associated set of appropriateness conditions, which are easily detectable and indicate the conditions under which the plan is appropriate to invoke.
3. when possible, standard preconditions for standard plans are *enforced*, so that they can be assumed true, and do not need to be included as as appropriateness conditions.

Part of the process of refining appropriateness conditions can be taken care of by relatively simple "recategorization" of various conditions in the taxonomy we sketched above, in response to both failure and unexpected success. Here some ways in which this sort of recategorization can be applied.

- Drop appropriateness conditions that turn out to be always true. At its simplest, this is merely a matter of keeping statistics on verification steps at the beginning of plans.
- If a plan fails because some subplan of it fails, and that subplan failed because some appropriateness condition didn't hold, then *promote* that condition to the status of an appropriateness condition for the superordinate plan. That is, make use of the larger plan contingent on finding the condition to be true.

- If a plan is frequently found to have false appropriateness conditions in situations where it is needed, and the conditions are under the agent's control, consider including it in an *enforcement* plan that maintains it, so that it can assumed true by the plan that was failing.

## Conclusion

The implicit dual role of classical preconditions should be separated out, into the preconditions that underlie the plan structure and the conditions that signal appropriate use of the plans. This separation opens the door for tuning of the conditions under which an agent will use a given plan. This sort of learning, in conjunction with other methods for improving the match between an environment and a plan library, is a promising method for improving the use and reuse of plans.

## Acknowledgements

Daniel Fu gave us helpful comments, as did various anonymous reviewers.

## References

[Agre and Chapman, 1987] Phil Agre and David Chapman. Pengi: An implementation of a theory of activity. In *The Proceedings of the Sixth National Conference on Artificial Intelligence*, pages 268–72. AAAI, July 1987.

[Agre and Chapman, 1988] Phil Agre and David Chapman. What are plans for? Memorandum 1050, Massachusetts Institute of Technology Artificial Intelligence Laboratory, 1988.

[Chapman, 1985] David Chapman. Planning for conjunctive goals. Memo AI-802, AI Lab, MIT, 1985.

[Chapman, 1990] David Chapman. On choosing domains for agents. Position paper prepared for the Workshop on Benchmarks and Metrics, NASA Ames, 1990.

[Drummond, 1989] Mark E. Drummond. Situated control rules. In *Proceedings of Conference on Principles of Knowledge Representation*, Toronto, Canada, 1989.

[Fikes and Nilsson, 1971] R. Fikes and N.J. Nilsson. STRIPS: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence*, 2:189–208, 1971.

[Ginsberg and Smith, 1987] Matthew L. Ginsberg and David E. Smith. Possible worlds and the qualification problem. In *Proceedings of AAAI-87*, July 1987.

[Hammond and Converse, 1991] Kristian Hammond and Timothy Converse. Stabilizing environments to facilitate planning and activity: an engineering argument. In *The Proceedings of the 1991 National Conference of Artificial Intelligence*, July 1991.

[Hammond *et al.*, 1990] Kristian Hammond, Timothy Converse, and Charles Martin. Integrating planning and acting in a case-based framework. In *The Proceedings of the 1990 National Conference of Artificial Intelligence*, August 1990.

[Hammond, 1989] Kristian Hammond. *Case-Based Planning: Viewing Planning as a Memory Task*, volume 1 of *Perspectives in Artificial Intelligence*. Academic Press, San Diego, CA, 1989.

[Martin, 1989] Charles E. Martin. *Direct Memory Access Parsing*. PhD thesis, Yale University Department of Computer Science, 1989.

[McCarthy, 1977] John McCarthy. Epistemological problems of artificial intelligence. In *Proceedings of the Fifth International Conference on Artificial Intelligence*, 1977.

[Rosenschein and Kaelbling, 1986] Stanley J. Rosenschein and Leslie Pack Kaelbling. The synthesis of digital machines with provable epistemic properties. In *Proceedings of 1986 Conference on Theoretical Aspects of Reasoning About Knowledge*, March 1986.

[Shoam, 1986] Yoav Shoam. *Reasoning about Change: Time and Causation from the Standpoint of Artificial Intelligence*. PhD thesis, Yale University, 1986. RR #507.