Foundations of Rational Agency, ed. Rao and Wooldridge, published by Kluwer.

Planning Agents

John L. Pollock
Department of Philosophy
University of Arizona
Tucson, Arizona 85721
(e-mail: pollock@arizona.edu)

1. Introduction

Agents are entities that act upon the world. Rational agents are those that do so in an intelligent fashion. What is essential to such an agent is the ability to select and perform actions. Actions are selected by planning, and performing such actions is a matter of plan execution. So the essence of a rational agent is the ability to make and execute plans. This constitutes *practical cognition*. In order to perform its principal function of practical cognition, a rational agent must also be able to acquire the knowledge of the world that is required for making and executing plans. This is done by *epistemic cognition*. Rational agents embedded in a realistically complicated world (e.g., human beings) may devote more time to epistemic cognition than to practical cognition, but even for such agents, epistemic cognition is in an important sense subservient to practical cognition.

This general view of rational agents has become enshrined in what are called *BDI architectures*. These endow agents with beliefs, desires (goals), and intentions (plans). The first implemented BDI architecture was PRS (Georgeff and Lansky 1987). Other interesting architectures of this same genus are represented by HOMER (Vere and Bickmore 1990) and IRMA (Bratman, Israel, and Pollack 1988). A useful survey can be found in Wooldridge and Jennings (1995). The OSCAR architecture (Pollock 1990, 1995) is also of this same general structure. Indeed, the structure is the standard one assumed in philosophical theories of rationality, although philosophers have usually focussed on intentions to perform individual actions rather than multi-step plans.[1]

The AI literature on rational agents is full of references to *autonomous agents*, which are agents that learn. It is a commonly held view that only autonomous agents are "real" rational agents, because others are merely running a program—doing what their makers tell them to do—rather than figuring out for themselves what to do. This view seems to me to be incoherent. First, agents that learn are not somehow magically making up their own program—they are running a learning program. Second, I see no reason to believe that learning is in any way essential to an agent's performing in a sophisticated or flexible manner. The purpose of machine learning is to enable the agent to acquire general knowledge about the world, but that same knowledge can be acquired by *reasoning* from perceptual input if the agent is equipped with the ability to perform inductive and abductive reasoning. I have elsewhere talked about *Q&I (quick and inflexible) modules*,[2] which are special purpose modules enabling an agent to perform certain cognitive tasks rapidly (they are "quick") provided the environment cooperates by satisfying appropriate constraints (they are "inflexible"). The kind of learning provided by machine learning programs is best viewed as a Q&I module for inductive and abductive reasoning. Such learning *might* be faster than explicit inductive or abductive reasoning, but that remains to be shown. There is in

---

This work was supported by NSF grant no. IRI-9634106.

[1] See my [1992] and [1995] for a discussion of this difference.

[2] Pollock [1996], chapter one.

principle nothing that an agent running a machine learning program can do that an agent equipped with the ability to perform sophisticated epistemic reasoning cannot do. So I think the emphasis on autonomy is a red herring. The emphasis should instead be on planning and plan-execution, and on the kinds of epistemic cognition that the latter require.

Planning is one of the oldest subfields of AI, and it has been a very active subfield. Work in planning has traditionally proceeded in what is essentially a bottom up fashion. "Theories of planning" have been constructed on the basis of logical and mathematical theories of search. The construction of planners has then been viewed as an engineering task that proceeds by applying the basic science provided by these theories. This approach has led to valuable insights, but the resulting planners are in many ways crude. This is for a variety of reasons. First, planning theorists have been quick to identify planning with a certain kind of search, but that leads directly to the view that a correct plan is *guaranteed* to achieve its goals (provided, of course, that the background beliefs of the planning agent are not faulty). Human planning is not like that at all. Plans do sometimes fail because they are based upon false beliefs, but it is much more common for them to fail because they are based upon *incomplete beliefs*. When planning in the real world, we never know everything that is or might be relevant to a planning problem, so there is no way we could conceivably construct plans that are guaranteed to work. That does not stop us from planning. To accommodate such planning within a theory of planning, we need a more sophisticated view of what plans are and what the criteria of adequacy should be for good plans. In the last few years, a number of researchers have begun to address these problems, but so far nothing has emerged to replace the elegant but simplistic view of planning as search that has dominated work on planning.

To make planning more tractable, planning theorists have traditionally built their planners on the basis of some simplifying assumptions. These include most notably the assumptions (1) that the planner has all the relevant knowledge that it needs at the time it begins to plan, and (2) that nothing of importance happens in the world except as a result of the agent's actions. The second assumption is actually a corollary of the first, because exogenous events would be events about which the planner lacks knowledge. Planners based upon these assumptions have substantially advanced our knowledge of the processes by which plans can be discovered or generated, but it is universally acknowledged that these assumptions must eventually be jettisoned and planners constructed that do not depend upon them. Much recent work has been devoted to relaxing these assumptions, but this work is still in its relative infancy.

We have learned much from the bottom up approach to planning, but the problem of building a sophisticated planning agent is probably too difficult to solve by treating it as a pure engineering problem. We should also approach planning from the top down. The inadequacies of current planners are most easily revealed by looking at the kind of sophisticated planning that human beings perform, trying to describe various aspects of it precisely, and then addressing the question of how an artificial agent might be constructed that is capable of performing such sophisticated planning. An essential aspect of this approach is that planning is viewed as something performed by a planning agent, and planning is integrated into the overall cognitive architecture of the agent rather than being a stand-alone module.

2. Interleaving Planning and Reasoning

The first AI planner (Green, 1969) did its work by reasoning deductively about plans. But almost all subsequent AI planners have worked algorithmically rather than by reasoning. You give them the necessary information and the goal and they run a program that searches the space of possible plans (relative to the given information) until it finds a plan whose execution is guaranteed to achieve the goal. The algorithmic approach to planning has been dictated largely by AI's inability to solve the frame problem.

Although this is not its official formulation,[3] we can think of the frame problem as the problem of constructing a reasoner that can reason correctly about the consequences of actions. Ideally, a rational agent should come to a planning problem with some general knowledge about the world and some specific knowledge about the current situation, and then it should construct a plan by reasoning about what actions will achieve various results. But, of course, this is impossible if the agent is unable to reason correctly about the consequences of actions. To circumvent this problem, virtually all AI planning systems are based upon some variant of the STRIPS representation of actions.[4] On the STRIPS representation, the human operator does the reasoning about the consequences of actions, and then builds the results of that reasoning into the description of the actions, thus absolving the artificial agent from performing the reasoning it is unable to do. This has generally been accomplished by associating a list of *preconditions* and an *add-list* and *delete-list* with each action. The add-list is the list of literals that will be made true by performing the action when the preconditions are satisfied, and the delete-list is the list of literals that will cease to be true as a result of performing the action when the preconditions are satisfied.

This way of avoiding the frame problem is obviously a kludge. We really want to build agents that do all the relevant reasoning. However, it is a helpful kludge for the purpose of constructing useful planning systems. Unfortunately, its power is limited. First, note that it is important that the add- and delete-lists consist of literals. This is because the planning agent does no reasoning about the consequences of actions. If we were to put, say, a conjunction into the add-list, the agent would be unable to conclude that the conjuncts will be made true as well.[5] This restriction to literals means that the designing of a planning system cannot encode relevant knowledge in a straightforward way, e.g., as a set of first-order formulas. Existing knowledge must be carefully instantiated and manicured for the use of the planner.

Partially because of these knowledge representation problems, the STRIPS-representation encounters the *qualification problem* and the *ramification problem*. The qualification problem can be regarded as the problem of making the preconditions sufficiently inclusive to ensure that the add- and delete-lists really consist of *guaranteed consequences* of the action given the preconditions.[6] In a realistically complex environment, this desideratum is almost impossible to achieve. We can list consequences that will *normally* ensue when an action is performed, but it is virtually impossible to list consequences that will result no matter what. In the real world, unexpected eventualities can foil even the most sophisticated planning.

This aspect of the qualification problem is only a problem insofar as the task of planning is viewed as that of generating a plan that is *guaranteed* to succeed. Human beings do not view planning in that way. We search for plans that are *likely* to succeed. If we revise our aspirations for AI planners similarly, then this aspect of the qualification problem becomes much less serious.

There remains a residue of the qualification problem that is an obstacle to planning even if we do not insist that our plans are guaranteed to succeed. The world-knowledge that we want to bring to bear on a planning problem tends to be fairly general. But on the STRIPS representation, the list of preconditions and the add- and delete-lists must consist of literals. These are generated by instantiating our general knowledge in various ways. However, there will usually be infinitely

---

[3] See my [1996] for a discussion of how exactly the frame problem should be formulated.

[4] Fikes and Nilsson [1971] .

[5] Recently, some planners have been built that do a limited amount of reasoning about consequences and hence are able to use a more expressive formalism. For example, both UCPOP (Penberthy and Weld 1992, Weld 1994) and PRODIGY (Veloso et al, 1995) are able to reason about disjunctive goals and universally general goals. However, they are based upon algorithms requiring that the effects of an action be computable, so the logical consequence relation must be decidable. That rules out a truly expressive formalism like full first-order logic.

[6] The qualification problem originated with McCarthy [1977], where it was formulated more generally as a problem for axiomatizing real-world domain in the situation calculus.

many different ways of instantiating our general knowledge. Suppose, for example, I am making a simple plan about how to feed my cat. My general knowledge allows me to infer that I can usually do this by opening a can of catfood, placing the contents in a bowl, and placing the bowl before my cat. However, my general knowledge also allows me to infer that this will not work if my cat is chasing a mouse, or comatose, or frightened by the noise of the dishwasher, etc. These qualifications can be made more complex without limit. Each produces a different STRIPS action (because the preconditions, add- and delete-lists are part of the action). Which actions should be provided to the planner?

The ramification problem is closely related to this residue of the qualification problem. The ramification problem is the problem of deciding which consequences of actions to build into the STRIPS actions. For example, consider the consequences of striking a match. If we think about it for a while, we can enumerate among the effects such things as displacing air around the match, marginally depleting the ozone layer, raising the temperature of the earth's atmosphere, marginally illuminating Alpha Centauri, making that match unavailable for future use, etc. These are effects that we typically do not care about, and so we do not reason about them. But this does not mean that we can omit them from the add- and delete-lists with impunity, because occasionally we might care about one of them. What this illustrates is that which consequences we build into a STRIPS-action must be determined by which planning problem we are trying to solve. For example, I would not ordinarily think about the fact that feeding my cat results in his gaining a small amount of weight, but if he is precariously balanced on a board that will collapse if he gains that weight, then my planning must take that into account and include a step to reinforce the board or move the cat.

Human planners solve both these problems by reasoning. They take account of only those qualifying circumstances that they think are likely to occur, and only those consequences that are relevant to their present interests. On the STRIPS-representation, this puts a very heavy burden on the reasoning of the human operator, and means in effect that the planner must be re-tailored for each planning problem. Unfortunately, it is generally impossible to decide what STRIPS-actions to provide for the planner without first solving the planning problem. For example, when reasoning by goal-regression, a human planner considers what he or she might do to achieve a certain goal, and instantiates his or her general knowledge in ways that will answer this question. What is important here is that the planning is interleaved with epistemic reasoning aimed at finding appropriate information for use in the planning, and the course of the epistemic reasoning is directed by where the agent is in its planning. In effect, requiring the human operator of an AI planner to perform this epistemic reasoning in advance requires him or her to already know how the planning problem will be solved, but this makes the planner superfluous.

I take it that what the qualification and ramification problems illustrate is the importance of interleaving planning and epistemic reasoning, and performing the epistemic reasoning in a way that is directed by or focused by the practical considerations involved in the planning. The epistemic reasoner must be *interest-driven* in the sense that it tries to answer particular queries that are passed to it by practical cognition. These will be factual queries of relevance to the planning problem, and will arise *in the course of* planning. We cannot assume that the planner comes to the planning problem equipped with precisely the knowledge it needs to solve the planning problem (and without the ability to do further reasoning) without ignoring essential aspects of planning and cognition.

3. Schematic Plans

What I have been arguing is that epistemic cognition and planning cannot be separated and performed by different unrelated cognitive modules. The ramification and qualification problems illustrate that one reason for this is that realistic planners cannot be supposed to come to their planning problems equipped with all the knowledge they need to solve them. Instead, *in the course of planning*, they must adopt epistemic interest in various questions whose answers impinge

on the planning process, and undertake epistemic investigations aimed at answering those questions. This means that *the course of planning effects the course of epistemic cognition.* The converse is true as well. Plans do not spring full-blown, with all their details specified, from a single planning session. This is because, no matter how much epistemic cognition we have performed, we never have all the relevant knowledge. We cannot know all the details relevant to even so simple a plan as walking across the room (e.g., will the cat run underfoot?). Human beings accommodate this by adopting plans that are to varying degrees schematic, and then try to make them more precise as further knowledge is acquired. This has not gone overlooked in AI planning. One aspect of it is captured by hierarchical planning.[7] In hierarchical planning, plans are constructed using "coarse-grained operations" like FLY_TO_LA or GET_A_PHD_IN_COMPUTER_SCIENCE or MAKE_A_CUP_OF_COFFEE or WALK_ACROSS_THE_ROOM. On the basis of general knowledge, we think it likely that we will be able to perform these operations. However, in order to perform these operations we must engage in further planning. We may not be able to do that until we acquire further knowledge that is not currently available. For example, in order to fly to LA I must buy a plane ticket, and in order to do that I must acquire knowledge about flight schedules. By constructing a plan using the coarse-grained operator FLY_TO_LA, I am able to go ahead and plan without first acquiring knowledge about flight schedules. When such knowledge later becomes available, I can refine my plan to take account of it.

Hierarchical planning captures only one aspect of the schematicity of human planning. Another respect in which human plans are schematic is scheduling. Most practical AI planners separate planning and scheduling. First they adopt a plan, and then they construct a schedule for when to perform the plan-steps. This is fairly obviously inadequate. The acceptability of a plan is certainly dependent upon our being able to schedule it in a reasonable way. Thus far I have not remarked on the fact that human planning is in some sense decision-theoretic. I have observed that we do not require plans to be guaranteed to achieve their goals—we only require it to be likely that they will do so. But even that is an oversimplification. We weigh the value of the goals discounted by the probability of success against the expected cost of executing the plan. This yields an expected value for the plan, and we decide whether to adopt a plan by comparing its expected value with that of competing plans.[8] It is apparent that the schedule that is attached to a plan can profoundly affect its expected value. Thus the schedule must be an integral part of the plan when it is being evaluated for adoption.

Although the schedule must be part of the plan, human beings rarely schedule the steps of a plan with much precision. For example, I may plan to do something *this afternoon*. Rarely do I plan to do something at 3:07:13 PM. This is not because I am being intellectually sloppy. It is good rational policy. First, most small differences in scheduling do not affect the expected value of a plan to any appreciable degree. Second, by leaving unimportant scheduling decisions indeterminate, one maintains flexibility. An agent residing in a realistically complicated world will continually receive new information through perception, and some of this new information will present new opportunities that provide new goals and call for new planning. Ideally, planning for new goals should disrupt previous planning as little as possible. This is, to my mind, the principal virtue of *least commitment planning*.[9] By constructing plans that do not make unnecessary commitments, either to the ordering of plan-steps, or to the times of execution of plan-steps, or to precisely how to execute coarse-grained plan-steps, we maximize the likelihood that the plans will not conflict with new planning.

I have argued that it is often good rational policy to construct schematic plans. But of course, at some point the schematicity must be eliminated. In order to execute a plan step, the agent must decide precisely how and when to do it. It was once common to suppose that planning

---

[7] Hierarchical planning was introduced by Sacerdotti [1975, 1977] .

[8] See chapter five of my [1995] for a discussion of the logic of the reasoning involved in the comparison.

[9] See Weld [1994] for a general discussion of least commitment planning.

and plan execution were separate processes, the latter being essentially simple—something like running a computer program. As planning theorists have come to appreciate that planning is generally based on incomplete and sometimes inaccurate information, they have also come to realize that plan execution is not so mechanical. At the very least it involves monitoring the course of the plan execution and replanning if things do not go as expected. But in fact, it involves more than that. The schematicity of plans generally extends into the period of plan execution. For example, suppose I plan to walk into the kitchen and fix a cup of coffee. My plan may initially consist of three steps: (1) walk to the kitchen door, (2) walk from there to the vicinity of the espresso machine, (3) prepare a cup of espresso. Each step is highly schematic. In executing the first step, I start walking in the direction of the kitchen door, but I do not plan precisely where I am going to place my feet before I begin the walk. Instead, the details of each step are determined on the fly as obstacles appear in my course or in response to my own sense of balance and knowledge of how I must place my feet in order to avoid falling down. Once I find myself in the vicinity of the espresso machine, I may then and only then decide on the order in which I perform the various operations required to make espresso. For example, whether I grind the beans first or put water in the boiler first may depend upon just where I find myself, in what order the various operations occur to me, or mere whim. As far as scheduling, I may never construct a precise schedule. At some point I decide to do something *now*, and then after I have done that I decide to do something else *now*, and so on.[10]

4. Planning by Reasoning

The upshot of the previous sections is that epistemic cognition, planning, and plan execution must all be interleaved. The course of each affects the course of the others. Perception and epistemic cognition based upon it gives rise to knowledge of opportunities, and this gives rise to goals. Adopting goals initiates planning for their achievement. Planning poses queries that are passed to epistemic cognition in an effort to acquire the knowledge needed for planning. Plan execution requires monitoring the world (epistemic cognition) and refinement, revision, or abortion of the plan as execution proceeds. Thus a realistic rational agent must perform all of these tasks in unison, not sequentially.

As I remarked above, virtually all contemporary planners are algorithmic planners. Plans are found by running a search algorithm rather than by reasoning. It is extremely difficult to see how to interleave algorithmic planning and epistemic cognition. An algorithmic planner depends upon already having all the information it needs, and if some of that information changes during the course of the plan search, it appears that an algorithmic planner will simply have to start all over again. The only obvious way to interleave planning and epistemic cognition is to do the planning by reasoning, just as human beings do.

In order to build a planner that plans by reasoning we must solve three problems. First, we must have a reasoner that is adequate to the task. AI has many automated deductive reasoners, but deductive reasoning does not provide an adequate basis for epistemic cognition. This has been acknowledged in philosophical epistemology for the last thirty years, and I hope I am right in thinking that this is now generally appreciated in AI as well. If not, all I can do is point my AI colleagues to the philosophical literature on the topic.[11]

There is a large AI literature on non-deductive reasoning. The earlier literature goes under the name "nonmonotonic logic", and more recently the term "defeasible reasoning" (taken from philosophy) has become popular. The importance of defeasible reasoning has been stressed by

---

[10] HOMER (Vere and Bickmore 1990) is an agent architecture that is designed to accommodate this observation about schematicity and plan execution. HOMER constructs hierarchical plans and produces subsidiary plans for executing high level operators as additional knowledge is acquired during the course of plan execution.

[11] As a start, I would suggest my own [1987] and [1995].

many important AI theorists, but there has not been a consensus on how defeasible reasoning should work and how it can be implemented. Without an implemented defeasible reasoner, we cannot implement a system of epistemic cognition, and without that we cannot implement a planner that plans by reasoning.

Given an automated defeasible reasoner, the second problem that must be solved is the construction of a general agent architecture that incorporates the system of defeasible reasoning and integrates it with planning and epistemic cognition. The third problem is that planning by reasoning runs afoul of the frame problem. Without a solution to the frame problem, we will be unable to build an artificial agent that is able to construct plans by reasoning.

Fortunately, I do not think that any of these problems are as serious as they once were. The OSCAR architecture for rational agents is based upon a general purpose defeasible reasoner, and is designed to integrate epistemic and practical cognition. I cannot claim that there is general agreement that OSCAR does it right, but I am convinced that OSCAR at least does it for the most part right, and perhaps OSCAR's success at providing the inference engine required for many AI purposes will inspire increasing confidence in OSCAR's correctness. That still leaves the frame problem as an obstacle to building planners that plan by reasoning. Fortunately, the frame problem is no longer so mysterious as it once was. In the last couple of years several authors have proposed solutions.[12] In particular, I have proposed a solution that is implemented in OSCAR and enables reasonably efficient reasoning about the consequences of actions. So OSCAR provides a basis upon which to at least begin the process of building agents that plan by reasoning. In the next section I will give a brief description of the OSCAR architecture, and in the following section I will sketch my proposed solution to the frame problem. Then I will return to the question of how to build a planning agent.

5. The OSCAR Architecture

OSCAR is an architecture for rational agents, based upon the world's only implemented general-purpose defeasible reasoner.[13] The architecture divides cognition into epistemic cognition (cognition about what to believe) and practical cognition (cognition about what to do). Epistemic cognition is interest-driven in the sense that it proceeds bidirectionally. The reasoner reasons forwards from perceptual input, and backwards from queries it is trying to answer. Initially, queries are posed by practical cognition, and then as reasoning proceeds, new epistemic interests are generated by reasoning backwards from the initial ones.

The architecture for epistemic cognition is diagramed in figure 1. Reasoning begins with perceptual input and ultimate-epistemic-interests. The latter are queries that are passed to epistemic cognition from practical cognition, without being derived from other interests by backwards reasoning. Conclusions are stored as nodes ("inference-nodes") in the *inference-graph*, which records both inference-relations between conclusions and defeat-relations between conclusions. Interests are stored in the *interest-graph*, which records their inference-relations. Conclusions have degrees of justification attached to them (numbers between 0 and 1) and interests have degrees of interest. These numbers are used in three ways. First, degrees of justification are used in adjudicating conflicts and computing defeat statuses. Second, they are used in interest discharge. The higher the degree of interest, the more serious the matter at

---

[12] See Shoham [1987], Gelfond and Lifschitz [1993], Lin and Reiter [1994] and [1995], Shanahan [1996].

[13] The theory underlying OSCAR is described in detail in my [1995]. The implementation is described in *The OSCAR Manual*. Both the manual and the implemented system can be downloaded from http://www.u.arizona.edu/~pollock/.

issue, and so the more justified a conclusion must be to settle the matter. So for a conclusion to discharge an interest, its degree of justification must be at least as great as the degree of interest. Backwards reasoning from an interest creates *interest-links*, which link a set of interests to the original interest. The significance of an interest-link is that if conclusions are drawn discharging the link-interests, then a conclusion discharging the original interest can be inferred from those conclusions. The third way in which degrees of interest and degrees of justification are used is in prioritizing the reasoning. Newly drawn conclusions and newly constructed interests are placed on the *inference-queue*, and then the next step of reasoning is performed by retrieving the first member of the inference-queue and processing it. The ordering of the inference-queue is based, among other things, on the degrees of justification and the degrees of interest of the items on it. Other things being equal, the more certain a conclusion or the more serious an interest, the higher its priority.

Reasoning is mediated by reasons. Forwards-directed reasons mediate forwards-reasoning (reasoning from conclusions to conclusions), and backwards-directed reasons mediate backwards-reasoning (reasoning from interests to interests). OSCAR appears to be unique in that its forwards-directed reasons are entirely distinct from its backwards-directed reasons. For example, ADJUNCTION is a backwards-directed reason, instructing the reasoner that if it is interested in $(P \ \& \ Q)$ then it should become interest in P and in Q. SIMPLIFICATION is a forwards-directed reason instructing the reasoner that if it has concluded $(P \ \& \ Q)$ then it should infer P and Q from that conclusion. This structure of reasoning makes OSCAR very efficient.[14]

---

[14] In a recent comparison of OSCAR with more standard resolution-based theorem provers, OSCAR proved to be forty times faster than Otter, a highly regarded state-of-the-art resolution-based theorem prover. The comparison was arranged by Geoff Sutcliffe, librarian of the TPTP library of theorem proving problems. It is noteworthy that OSCAR did so well despite the fact that OSCAR only does theorem proving as a side effect of its general reasoning, and is burdened with a great deal of extra machinery that is required for defeasible reasoning. In addition, Otter is written in C while OSCAR is written in LISP. This should give Otter an advantage of as much as an order of magnitude .

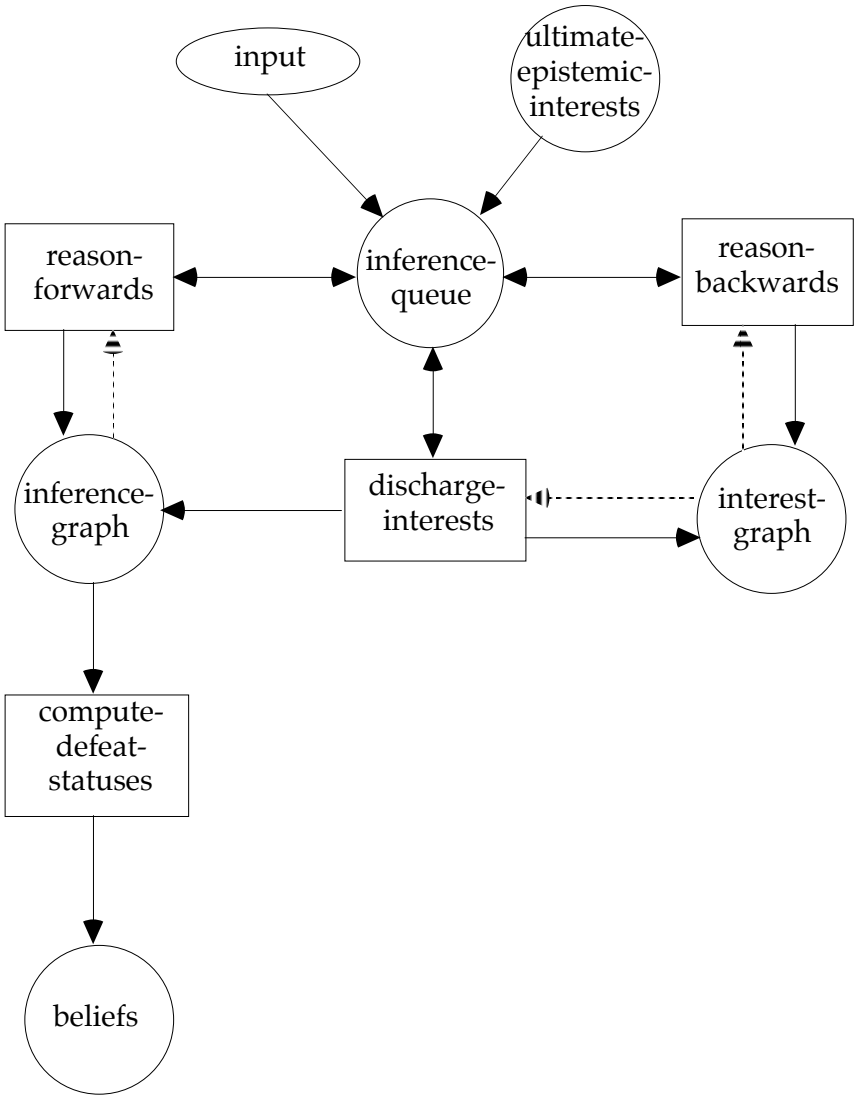

Figure 1. The defeasible reasoner

Reasoning proceeds in terms of reasons. *Backwards-reasons* are used in reasoning backwards, and *forwards-reasons* are used in reasoning forwards. Forwards-reasons are data-structures with the following fields:

- reason-name.
- forwards-premises — a list of forwards-premises.
- backwards-premises — a list of backwards-premises.
- reason-conclusions — a list of formulas.
- conclusions-function — an optional function used for computing the conclusions of an inference made in accordance with the reason. If the function is not supplied, the conclusions are computed by instantiating the reason-conclusions.
- defeasible-rule — T if the reason is a defeasible reason, NIL otherwise.
- reason-variables — variables used in pattern-matching to find instances of the reason-premises.

- reason-strength — a real number between 0 and 1, or an expression containing some of the reason-variables and evaluating to a number.
- reason-description — an optional string describing the reason.

Forwards-premises are data-structures encoding the following information:
- fp-formula — a formula.
- fp-kind — :inference, :percept, or :desire (the default is :inference)
- fp-condition — an optional constraint that must be satisfied by an inference-node for it to instantiate this premise.
- clue? — explained below.

Similarly, *backwards-premises* are data-structures encoding the following information:
- bp-formula
- bp-kind

Backwards-reasons will be data-structures encoding the following information:
- reason-name.
- forwards-premises.
- backwards-premises.
- reason-conclusions — a list of formulas.
- conclusions-function.
- reason-variables — variables used in pattern-matching to find instances of the reason-premises.
- strength — a real number between 0 and 1, or an expression containing some of the reason-variables and evaluating to a number.
- defeasible-rule — T if the reason is a defeasible reason, NIL otherwise.
- reason-condition — a condition that must be satisfied by the sequent of interest before the reason is to be deployed.

Simple forwards-reasons have no backwards-premises, and *simple backwards-reasons* have no forwards-premises. Given inference-nodes that instantiate the premises of a simple forwards-reason, the reasoner infers the corresponding instance of the conclusions. Similarly, given an interest that instantiates the first conclusion of a simple backwards-reason, the reasoner adopts interest in the corresponding instances of the backwards-premises. Given inference-nodes that discharge those interests, an inference is made to the conclusions from those inference-nodes.

In deductive reasoning, with the exception of a rule of reductio ad absurdum, we are unlikely to encounter any but simple forwards- and backwards-reasons.[15] However, the use of backwards-premises in forwards-reasons and the use of forwards-premises in backwards-reasons provides an invaluable form of control over the way reasoning progresses. This will be illustrated below. *Mixed* forwards- and backwards-reasons are those having both forwards- and backwards-premises. Given inference-nodes that instantiate the forwards-premises of a mixed forwards-reason, the reasoner does not immediately infer the conclusion. Instead the reasoner adopts interest in the corresponding instances of the backwards-premises, and an inference is made only when those interests are discharged. Similarly, given an interest instantiating the first conclusion of a mixed backwards-reasons, interests are not immediately adopted in the backwards-premises. Interests in the backwards-premises are adopted only when inference-nodes are constructed that instantiate the forwards-premises.

Reasons are most easily defined in OSCAR using the macros DEF-FORWARDS-REASON and DEF-BACKWARDS-REASON:

(def-forwards-reason *symbol*

---

[15] This is discussed at greater length in Chapter Two of my [1995a].

:forwards-premises *list of formulas optionally interspersed with expressions of the form (:kind ...) or (:condition ...)*
:backwards-premises *list of formulas optionally interspersed with expressions of the form (:kind ...) or (:condition ...)*
:conclusions *list of formulas*
:conclusions-function
:strength *number or a an expression containing some of the reason-variables and evaluating to a number.*
:variables *list of symbols*
:defeasible? *T or NIL (NIL is the default)*
:description *an optional string (quoted) describing the reason*)

(def-backwards-reason *symbol*
:conclusions *list of formulas*
:conclusions-function
:forwards-premises *list of formulas optionally interspersed with expressions of the form (:kind ...) or (:condition ...)*
:backwards-premises *list of formulas optionally interspersed with expressions of the form (:kind ...) or (:condition ...)*
:condition *this is a predicate applied to the binding produced by the target sequent*
:strength *number or an expression containing some of the reason-variables and evaluating to a number.*
:variables *list of symbols*
:defeasible? *T or NIL (NIL is the default)*
:description *an optional string (quoted) describing the reason*)

The use of these macros will be illustrated in the next two sections.

6. A Solution to the Frame Problem

The OSCAR architecture provides the first cog in the machinery with which I propose to build a planning agent. The second cog must be a solution to the frame problem. I have recently proposed such a solution, and implemented it in OSCAR.[16] The proposal consists of a triumvirate of principles.

The first is a principle of temporal projection. This is a precise version of what has also been called "the common sense principle of inertia". What this principle tells us is that there is a presumption that things do not change. Many authors have suggested that some such principle is a necessary constituent of a solution to the frame problem. Because of its popularity, I will not defend it further here.[17] The version I employ is as follows:

TEMPORAL-PROJECTION
If P is temporally-projectible and $\Delta t < \log(.5)/\log(\rho)$ then believing P-at-t is a defeasible reason of strength $2\cdot(\rho^{\Delta t} - .5)$ for the agent to believe P-at-$(t+\Delta t)$.

ρ is a constant (a probability less than 1.0) that I call *the temporal-decay factor*. It is used in the principle to guarantee that the presumption of nonchange weakens as the time interval increases. The temporal-projectibility constraint is a syntactic constraint that is required for much the same reason projectibility constraints are required in principles of induction. Its main function here is to preclude disjunctions. This is discussed in detail in my [1996] and [1996a]. This principle can be implemented in OSCAR as follows:

(def-backwards-reason TEMPORAL-PROJECTION
:conclusions "(p at time)"
:condition (and (temporally-projectible p) (numberp time))
:forwards-premises

---

[16] Pollock [1996] and [1996a].

[17] See my [1996] for a full defense of this principle.

```
    "(p at time0)"
    (:condition (and (time0 < time*) ((time* - time0) < 693)))
  :variables p time0 time
  :defeasible? T
  :strength  (- (* 2 (expt *temporal-decay* (- time time0))) 1)
  :description
  "It is defeasibly reasonable to expect temporally projectible truths to remain unchanged.")
```

The frame problem arises in connection with causal reasoning. We want to reason about what changes would be caused by an action. I have argued (in my [1996] and [1996a]) that the causal relation involved in this reasoning is *causal-sufficiency*, where "*A* when *P* is causally sufficient for *Q* after an interval ε" means "if action *A* is performed at a time *t* and *P* is true at *t*, then it follows from causal laws that *Q* is true throughout some open interval beginning at *t*+ε". The latter can be symbolized as

$$(\forall t)\{(A\text{-}at\text{-}t \ \& \ P\text{-}at\text{-}t) \Rightarrow (\exists \delta) Q\text{-throughout-}(t+\varepsilon \ , \ t+\varepsilon+\delta)\}.$$

where "≡>" is the conditional that results from instantiating causal laws.

Suppose that *A* when *P* is causally sufficient for *Q* after an interval ε. If *A* and *P* are true at time *t*, it follows that *Q* is true at *some* time $t^* > t+\varepsilon$, but it does not follow that *Q* is true at any particular time. This is because *t*+ε is not contained in the open-interval (*t*+ε , *t*+ε+δ), and as δ is existentially quantified, for any time $t^* > t+\varepsilon$, it could be that $t^* > t+\varepsilon+\delta$. However, for any time $t^* > t+\varepsilon$, it does follow that there is a time t^{**} such that $t^* > t^{**} > t+\varepsilon$ and *Q* is true at t^{**}. Consequently, if *Q* is temporally-projectible, temporal-projection gives us a defeasible reason to expect *Q* to be true at t^*. This reasoning is captured by the following principle:

> *CAUSAL-IMPLICATION*
> If *Q* is temporally projectible, $(t^{**}-(t+\varepsilon)) < \log(.5)/\log(\rho)$, and $(t+\varepsilon) < t^*$, then "(*A* when *P* is causally sufficient for *Q* after an interval ε) & *A*-*at*-*t* & *P*-*at*-*t*" is a defeasible reason for "*Q*-*at*-t^*.

This is implemented as follows:

```
(def-backwards-reason CAUSAL-IMPLICATION
  :conclusions  "(Q at time*))"
  :forwards-premises
  "(A when P is causally sufficient for Q after an interval interval)"
    (:condition (every #'temporally-projectible (conjuncts Q)))
  "(A at time)"
  (:condition
   (and (eq op 'clopen) ((time + interval) < time*) ((time* - (time + interval)) < 693))
  :backwards-premises
  "(P at time)"
  :variables  A P Q interval time time* op
  :strength  (- (* 2 (expt *temporal-decay* (- time* time))) 1)
  :defeasible?  T)
```

At one time, many researchers thought that a principle of temporal projection might be sufficient by itself to solve the frame problem, but that hope was dashed by Steve Hanks and Drew McDermott [1986]. They illustrated the difficulty with an example that has become known as the Yale Shooting Problem. The general form of the problem is this. Suppose we have a causal law to the effect that if *P* is true at a time *t* and action *A* is performed at that time, then *Q* will be true shortly thereafter. Suppose we know that *P* is true now, and *Q* false. What should we conclude about the results of performing action *A* in the immediate future? Hanks and McDermott illustrate this by taking *P* to be "The gun is loaded and pointed at Jones", *Q* to be

"Jones is dead", and A to be the action of pulling the trigger. We suppose (simplistically) that there is a causal law dictating that if the trigger is pulled on a loaded gun that is pointed at someone, that person will shortly be dead. Under these circumstances, it seems clear that we should conclude that Jones will be dead shortly after the trigger is pulled.

The difficulty is that all we can infer from what we are given is that when A is performed either P will no longer be true or Q will be true shortly thereafter. Intuitively, we want to conclude (at least defeasibly) that P will remain true at the time A is performed and Q will therefore become true shortly thereafter. But none of our current machinery enables us to distinguish between P and Q. Because P is now true and Q is now false, we have a defeasible reason for believing that P will still be true when A is performed, and we have a defeasible reason for believing that Q will still be false shortly thereafter. We know that one of these defeasible conclusions will be false, but we have no basis for choosing between them, so this becomes a case of collective defeat. This is diagramed in figure 2, where the thin arrows indicate inference-relations and the fuzzy arrows indicate defeat-relations. We have equally good defeasible arguments for "Jones is alive at 50" and "Jones is not alive at 50", and no independent source of defeat, so both conclusions are defeated. That, however, is the intuitively wrong answer.

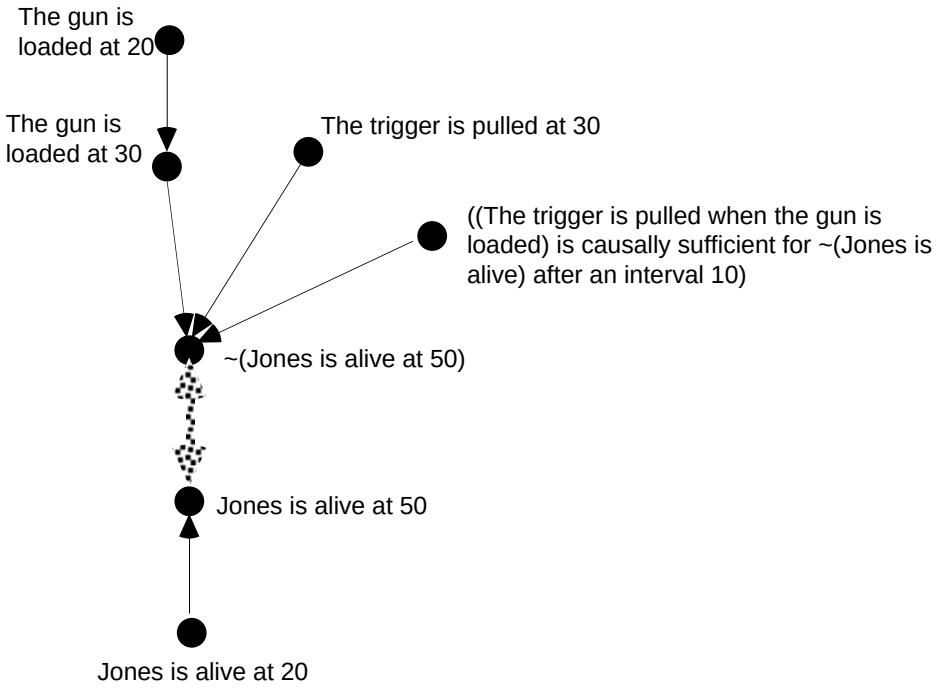

Figure 2. The Unsolved Yale Shooting Problem

A solution to the Yale Shooting Problem must give us some reason to favor the upper defeasible argument over the lower one in figure 2. My proposal is this. Whenever one event causes another, temporal projection gives us a reason for expecting the change not to occur. Consequently, when reasoning about a causal system, part of the force of describing it as causal must be that the defeasible presumption against the effect occurring is somehow removed. Thus, although we normally expect Jones to remain alive, we do not expect this any longer when he is shot. To remove a defeasible presumption is to defeat it. This suggests that there is some kind of general "causal" defeater for temporal projection.

When we reason about causal mechanisms, we think of the world as "unfolding" temporally, and changes only occur when they are forced to occur by what has already happened. In our

example, when A is performed, nothing has yet happened to force a change in P, so we conclude defeasibly that P remains true. But given the truth of P, we can then deduce that at a slightly later time, Q will become true. Thus when causal mechanisms force there to be a change, we conclude defeasibly that the change occurs in the later states rather than the earlier states. This seems to be part of what we mean by describing something as a causal mechanism. Causal mechanisms are systems that force changes, where "force" is to be understood in terms of temporal unfolding.[18] The idea is that in defeating a temporal projection, we can use things that have already happened but not things that are about to happen. I propose to capture this intuition with the following defeater for temporal projection:

> *CAUSAL-UNDERCUTTER*
> Where $t_0 \leq t_1$ and $(t_1+\varepsilon) < t$, "A-*at*-t_1 & Q-*at*-t_1 & (A when Q is causally sufficient for $\sim P$ after an interval ε)" is a defeasible undercutting defeater for the inference from P-*at*-t_0 to P-*at*-t by TEMPORAL-PROJECTION.

This can be implemented as follows:[19]

```
(def-backwards-undercutter CAUSAL-UNDERCUTTER
   :defeatee *temporal-projection*
   :forwards-premises
      "(A when Q is causally sufficient for ~p after an interval interval)"
      "(A at time1)"
        (:condition (and (time0 <= time1) ((time1 + interval) < time)))
   :backwards-premises
      "(Q at time1)"
   :variables  A Q p time0 time time1 interval
   :defeasible?  T)
```

With the addition of this principle, the Yale Shooting Problem is solved as in figure 3.

---

[18] This intuition is reminiscent of Shoham's [1987] "logic of chronological ignorance", although unlike Shoham, I propose to capture the intuition without modifying the structure of the system of defeasible reasoning. This is also related to the proposal of Gelfond and Lifschitz [1993]. This same idea underlies my analysis of counterfactual conditionals in Pollock [1969] and [1984].

[19] DEF-BACKWARDS-UNDERCUTTER is a variant of DEF-BACKWARDS-REASON that computes the form of the conclusion as the form of the undercutting defeater for the reason listed.

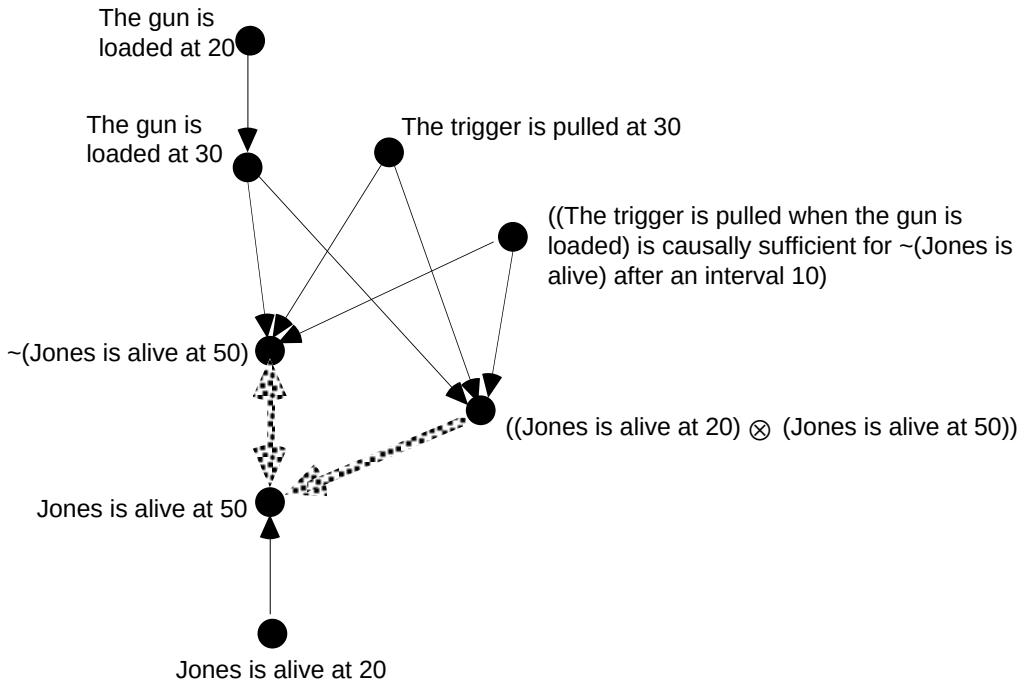

Figure 3. The Solved Yale Shooting Problem.

7. Reasoning Defeasibly about Plans

I have argued that planning should be done by reasoning rather than algorithmically. The two most obvious difficulties for producing a planner that works in this way are that (1) we require an agent architecture that can provide the requisite inference-engine, and (2) if an agent based upon this architecture is to be able to reason about plans, we must build a solution to the frame problem into the reasoning. Sections five and six sketched my implemented proposals for meeting these two difficulties. Does this mean that we are now in a position to build a planning agent?

Unfortunately, the answer is "Not quite." Several large problems remain. The first is that in reasoning about plans, a planning agent must reason about what *would happen if* an action *were* performed. This is *counterfactual* or *subjunctive* reasoning. Section six is about *indicative* causal reasoning—reasoning about what will be the case if an action *is* performed. The difference is a subtle one, but the literature on counterfactual or subjunctive conditionals makes it apparent that it is also an important one.[20] Before we are in a position to build a planning agent, we must incorporate the solution to the frame problem into an implemented account of subjunctive reasoning. This is a task that I have not yet undertaken. When I do, I propose to pursue an account of subjunctive reasoning based upon my theory of subjunctive conditionals in my [1984].

Even though we lack an implementation of subjunctive reasoning, we can still make some progress in exploring how planning might be done by reasoning. For this purpose, let us write subjunctive conditionals in the form "P => Q". Although we lack a full theory of how to reason

[20] See Lewis [1973], Pollock [1969], [1984].

about these conditionals, a few principles are obvious. For example, a version of SIMPLIFICATION holds for the consequents of these conditionals:

```
(def-forwards-reason SIMPLIFY-=>
   :forwards-premises "(P => (Q & R))"
   :conclusions  "(P => Q)" "(P => R)"
   :variables P Q R)
```

So for present purposes I will allow myself to make use of these conditionals even without a complete theory of precisely how they work.

Let us turn to planning then. I will sketch the construction of a goal-regression planner within OSCAR. The plans produced by this planner are essentially the same as those produced by SNLP.[21] *Plans* will be data-structures with the following fields:

- plan-steps — *a list of plan-nodes*
- plan-goals — *the list of goals the plan aims to achieve*
- a set of ordering constraints.

Plans will be nonlinear, in the sense that the plan-steps may be only partially ordered. The only ordering imposed is that required by causal-links and that contained in the explicit ordering constraints. The interpretation of a nonlinear plan is that every linearization of the plan (i.e., every total ordering of the plan-steps consistent with the explicit ordering constraints and causal-links) is expected to achieve the goals of the plan.

Plan-nodes will be data-structures with the following fields:

- plan-node-action
- plan-node-goals — *the list of goals the plan-node aims to achieve*
- call-set — *the list of causal-links having the node as their target.*

Finally, *causal-links* will be data-structures with the following fields:

- causal-link-root—*a plan-node*
- causal-link-goal
- causal-link-target—*a plan-node.*

The causal-link "*n --g--> n**" signifies that the root *n* aims to achieve the goal *g* which is a precondition for the target *n** to achieve whatever goal it aims to achieve.

OSCAR will find plans by attempting to answer queries of the form (?*p*)(plan-for *p goal*), meaning "find a *p* such that *p* is a plan for *goal*". The fundamental operation of goal-regression is provided by a backwards-reason:

```
(def-backwards-reason GOAL-REGRESSION
   :conclusions "(plan-for plan goal)"
   :condition (interest-variable plan)
   :backwards-premises
      "((precondition & action) => goal)"
      "(plan-for subplan precondition)"
      "(define plan (extend-plan action goal subplan))"
   :variables precondition action goal plan subplan)
```

The "define" premise works like a "let", temporarily binding the variable *subplan*. The function (*extend-non-linear-plan action goal subplan precondition*) constructs a new plan-node having *action* as its action and *precondition* as its goal, and then constructs a new plan by adding this plan-node to *subplan*.

The simplest case of goal-regression planning occurs when the goal is already true. In that case we conclude that it is achieved by a null-plan, which is a plan with no plan-steps. This reasoning is accomplished by the following backwards-reason:

---

[21] McAllester and Rosenblitt [1991].

```
(def-backwards-reason NULL-PLAN
   :conclusions "(plan-for plan goal)"
   :condition (interest-variable plan)
   :backwards-premises
      "goal"
      "(define plan (null-plan goal))"
   :variables goal plan)
```

(*null-plan goal*) is a plan with no plan-steps and with a goal-node that is a plan-node having *goal* as its plan-node-action, and having a single causal-link *start* --*goal*--> *goal*.

Goal-regression planning is made complicated by conjunctive goals. When faced with a conjunction of goals, humans plan separately for the individual goals, and then try to merge the plans. However, that does not always work because the plans for the individual goals may interfere with each other. In decision-theoretic planning, separate plans can interfere subtly by affecting each other's expected values, but at this stage I am only dealing with non-decision-theoretic planning. In this case, the way one plan interferes with another is by *clobbering one of its causal-links.* Suppose $plan_2$ contains the causal-link n --g--> n^*. This means that g is a subgoal required for node n^* to do its job, and g is achieved in $plan_2$ by node n. $Plan_1$ clobbers this causal-link iff $plan_1$ contains a subplan p consisting of nodes from $plan_1$ where the final step of p could (consistent with the ordering constraints) be executed between n and n^*, and p is a plan for $\sim g$. This means that there is a linearization of the plan that results from combining $plan_1$ and $plan_2$ in which node n makes g true, and then the last node of p is executed and makes g false again before it is used by node n^*.

How should we deal with this problem? It is tempting to suppose that when planning for conjunctive goals, we should construct plans for the individual goals, and then verify that they do not clobber one another *before* we merge them into a single plan and conclude that the merged plan is a plan for the conjunctive goal. However, this is not the way human planning works. Humans assume defeasibly that the separate plans do not clobber one another, and so infer defeasibly that the merged plan is a good plan for the conjunctive goal. Having made that defeasible inference, they then look for possible clobberings that would defeat it, but they do not regard it as essential to establish that there are no such clobberings before they make the inference. The difference is an important one. The plan might be such that if it is to be executed, execution must begin before the search for clobberings terminates. The agent that assumes defeasibly that the plan is a good one will go ahead and begin execution, while the more conservative agent that requires proof that there are no clobberings will be unable to execute the plan.

Which kind of agent is preferable? It might seem that the conservative agent is the more rational one. But in fact, the conservative agent cannot be made to work. This is related to the difference between algorithmic planners and planners that plan by reasoning. An algorithmic planner makes the assumption that it has all the relevant knowledge before planning begins. Given that assumption, it can compute whether there is any clobbering. But as we have seen, that is an unrealistic assumption. A realistic planning agent will guide its search for knowledge in light of the planning it is doing. In particular, when it constructs two plans and wants to merge them, it will then begin an investigation into whether there is any clobbering. However, this can be an indefinitely complex epistemic investigation. There will not in general be any point at which the agent can stop the investigation and conclude definitively that there is no clobbering. All it can conclude at any given point is that so far it sees no reason to think there is any clobbering. This is just one more example of the fact that a realistic agent that reasons defeasibly can never stop reasoning, because there is always the possibility that further reasoning will reveal heretofore unsuspected defeaters. Of course, we do not want the agent to spend all its time looking for defeaters instead of acquiring new beliefs, so the prioritization of the inference-queue must be adjusted to give the search for defeaters a relatively low (but still significant) priority.

The conclusion to be drawn from this is that the only way to make planning work in a realistically complex environment is by assuming defeasibly that separate plans do not clobber each other's causal-links. This point is of more far ranging significance than might at first be

realized. It arose in the context of conjunctive *sub*goals generated by goal-regression. But it arises equally from more general features of planning. Traditional planning systems have often adopted the fiction that all of the goals are given from the beginning, and the planner plans for them all simultaneously. This is certainly not true of human beings. We acquire new goals all the time. When we acquire a new goal, we do not throw out all of our previous plans and begin planning all over again for a single new overarching conjunctive goal that includes both our previous goals and our new goal. No planner could work that way. It is computationally too difficult. Rather, humans plan for the new goal and try to merge that plan with the plans that have already been made for other goals. Artificial agents must work that way as well. Of course, sometimes plans cannot be merged and we may have to modify some of our earlier plans and possibly replan for some of our earlier goals, but the point is that we do not do that unless we have to. What makes this possible is the defeasible assumption that plans can be merged without clobbering.

Planning separately for the different conjuncts of a conjunctive goal is accomplished by using the following backwards-reason:

```
(def-backwards-reason SPLIT-CONJUNCTIVE-GOAL
   :conclusions  "(plan-for plan& (goal1 & goal2))"
   :condition  (interest-variable plan&)
   :backwards-premises
      "(plan-for plan1 goal1)"
      "(plan-for plan2 goal2)"
      (:condition (or (plan-steps plan1) (plan-steps plan2)))
      "(define plan& (merge-plans plan1 plan2 goal1 goal2))"
   :defeasible? t
   :variables  goal1 goal2 plan1 plan2 plan&)
```

Inferences made in accordance with this reason are defeated by finding that the merged plan clobbers one of the causal-links in one of the constituent plans. This is accommodated by adopting the following defeater for SPLIT-CONJUNCTIVE-GOAL:

```
(def-backwards-undercutter CLOBBER-CAUSAL-LINKS
   :defeatee  split-conjunctive-goal
   :backwards-premises
   "(define links (union (call-set (goal-node plan&)) (unionmapcar #'call-set (plan-steps plan&))))"
   "(plan-clobbers-causal-links plan& links)"
   :variables plan& links)
```

This is supplemented with principles enabling the reasoner to reason about plans clobbering links.

Nonlinear plans attempt to make as few ordering commitments as possible, for the reasons discussed in section three. But sometimes that is the source of the clobbering, and the clobbering can be avoided by adding ordering constraints. When clobbering is discovered, the following reason attempts to add ordering constraints that avoid the clobbering by promoting or demoting the offending plan-nodes:

```
(def-backwards-reason ADD-ORDERING-CONSTRAINTS
   :conclusions  "(plan-for plan& (goal1 & goal2) goals)"
   :condition  (interest-variable plan&)
   :forwards-premises
      "(plan-for plan1 goal1)"
      "(plan-for plan2 goal2)"
      "(merged-plan plan plan1 plan2)"
      (:clue? t)
      "(plan-clobbers-causal-link plan R node link)"
      (:condition
        (or (member link (call-set (goal-node plan)))
```

```
        (some #'(lambda (n) (member link (call-set n))) (plan-steps plan))))
    (:clue? t)
    "(define plan& (add-not-between node link plan plan1 plan2))"
    (:condition (not (null plan&)))
  :defeasible? t
  :variables  plan1 plan2 plan& plan node link goal1 goal2 R)
```

Discovering the clobbering *occasions* the inference to "(plan-for *plan+ goal goals*)" without being a premise of the inference. This is captured in OSCAR by making the last premise of the reason a *clue*. When a premise is a clue, it must be instantiated by an inference-node before a new conclusion is drawn, but the basis from which the new conclusion is inferred does not include the inference-node instantiating the clue. Note also that this is a degenerate backwards-reason.

Although I have not discussed all of the reasons required for reasoning about nonlinear plans, it may still be useful to consider a simple example. Suppose Horatio is at the library, and wants to know Beethoven's birthday and also what time it is. Suppose Horatio is given the following conditionals:

(∀x)[((at-library x) & (ask-librarian x)) => (know-beethoven-birthday x)]
(∀x)[((at-clock x) & (read-clock x)) => (know-time x)]
(∀x)[((at-library x) & (go-to-clock x)) => ((at-clock x) & ~(at-library x))]

Using the reason-schemas described above, OSCAR finds the following plan:

```
PLAN-STEPS:
  (2) (ask-librarian horatio)
      causal-links:
        0 --(at-library Horatio)--> 2
  (4) (go-to-clock horatio)
      causal-links:
        0 --(at-library Horatio)--> 4
      ordering constraints:
        4 > 2
  (6) (read-clock horatio)
      causal-links:
        4 --(at-clock Horatio)--> 6
GOAL:   ((know-beethoven-birthday Horatio) & (know-time Horatio))
      established by:
        2 --> (know-beethoven-birthday Horatio)
        6 --> (know-time Horatio)
```

This is, of course, a very simple problem, but this approach to planning works equally well for more complex examples. As remarked above, it produces essentially the same plans as SNLP, but produces them by reasoning rather than algorithmically. The addition of a few more reason-schemas enables OSCAR to plan at the richer level of UCPOP (Penberthy and Weld, 1992). As we have seen, there is a tremendous advantage to this approach in that it automatically interleaves planning and epistemic reasoning and thereby achieves all the benefits described above. At the same time, there is a cost. Doing anything by reasoning is slow. OSCAR is roughly an order of magnitude slower than a comparable algorithmic planner.[22] However, the slowdown is a linear one, and the planning is still much faster than a human being can accomplish, so the cost does not seem too onerous.

---

[22] I have used UCPOP as my standard of comparison. See Penberthy and Weld [1992], or Weld [1994].

8. Sophisticated Planning

The planning system described in the preceding section may actually be useful, but it falls far short of being a solution to the problem of constructing a planning agent that can plan at the level of sophistication of a human being. Although it is a start, the planner described above should be viewed as a proof of concept for the idea of planning by reasoning rather than as a solution to the problem of building a truly sophisticated planner.

This planner does make some significant advances. The most important is that it integrates epistemic cognition and planning. Thus rather than requiring the planner to have all relevant knowledge before planning begins, the course of planning can direct the agent to address various epistemic issues that were not previously addressed and then return to the planning problem once answers have been acquired. Because planning is just part of reasoning, and the reasoning is prioritized by factors relating to its importance, this means, for example, that a planning agent can begin planning for one goal, break that off in the middle when a more important goal arises, and then return to the original planning problem later without losing the work that has already been done.[23] This is an automatic consequence of the OSCAR architecture. It also means that the information the planner uses in planning can come in any logical form. It does not have to be artificially restricted to lists of literals, as in the STRIPS-representation.

So what's missing? Unfortunately, quite a lot. First let me mention a relatively simple omission (at least, simple relative to the others). This is that the planner described above does not construct conditional plans.[24] Human plans are full of conditionals. For example, in planning the route I will drive across town, I may plan to go one way unless the traffic is too heavy, but if the traffic is heavy I will go another way. The problem for building a conditional planner lies in deciding which eventualities should be treated as subgoals and thereby initiate further goal regression and which should be left as conditions. Existing conditional planners tend to handle this in an *ad hoc* manner. The human operator is required to list the eventualities that can go into conditions. Obviously, we want more. It is not even correct to suppose that there is any kind of precise division between eventualities that are treated as subgoals and those that are treated as conditions. How they are treated depends, among other things, on whether we can think of a subplan for achieving them. For instance, while visiting in someone else's home I may spy a dish of particularly succulent candy. This may generate in me the goal of eating a piece, and this may inspire me to form the plan to accept a piece if my host offers it to me. Depending on the circumstances (e.g., how well I know my host), my planning may stop there with no attempt to plan further for the subgoal of getting my host to make such an offer, or I might construct the further plan of looking fondly at the candy and saying, "Um, that looks good!". Whether I can propound a plan like the latter depends entirely on the circumstances.

A much more serious lacuna in the preceding planner is that it is not decision-theoretic. Human planning is profoundly decision-theoretic. We attach values to goals, probabilities to possible outcomes of actions, and expected values to plans. When plans conflict, we must decide which plans to adopt, and we do that in part by comparing their expected values.[25] The decision-theoretic character of plans affects all aspects of planning. My proposal for solving the frame problem must be generalized to accommodate reasoning about the probabilistic consequences of actions. I have not yet even addressed the question of how to reason about subjunctive conditionals, but when such an account is produced it must immediately be generalized to extend it to subjunctive probabilities (probabilities that something *would be* the case if something

---

[23] This is one of the desiderata Beetz and McDermott [1966] stresses.

[24] This is to be distinguished from planning with actions having conditional effects. OSCAR handles the latter in the same manner as UCPOP.

[25] The logic of this is more complex than planning theorists generally realize. We cannot simply pick the plan with the higher expected value. See chapter six of my [1995] for a discussion of this.

else *were* the case). The reasons employed in searching for plans must also be revised to take account of decision-theoretic dimensions. Furthermore, the conclusion of the reasoning involved in the plan search will no longer be that the plan discovered is a plan for the goal, but rather that it is a plan for the goal and has a certain expected value. In non-decision-theoretic planning, drawing the conclusion that the plan is a plan for the goal can perhaps be identified with adopting the plan, but that is no longer adequate in decision-theoretic planning. We may conclude that a plan has a certain expected value but should not be adopted because it conflicts with other plans having even higher expected values. Thus we need a whole new tier of rational machinery built on top of the planning reasoning that decides which plans are to be adopted.

An equally serious lacuna is the omission of any serious machinery aimed at scheduling. The planner of the preceding section can do only the most rudimentary scheduling, that in terms of ordering constraints and causal links. Serious scheduling must be combined with decision-theoretic planning, because choices between schedules will often profoundly affect the expected value of a plan, and that is the basis upon which scheduling decisions should be made.

I have talked about how to build agents that plan, but one of the most important properties of planning agents is that they plan as little as possible. Planning is computationally difficult, and accordingly both slow and exhaustive of cognitive resources that could be employed elsewhere. Human beings solve simple planning problems by reusing previous plans. Plan-schemas are stored in memory and reinstantiated to handle new planning problems without engaging in new planning. Planning theory has taken this observation to heart, and there is a large literature on plan reuse. There is reason to hope, however, that planning by reasoning will simplify the problem of plan reuse. If the reasoning is such as to generate general beliefs about plans rather than very specific beliefs about fully concrete plans, then those general beliefs can be retained and reused as needed in new planning problems. However, this is a dimension of planning by reasoning that I have not yet explored.

Earlier, I stressed the importance of interleaving planning and plan execution, but I have said nothing further about plan execution. That is a topic that must be explored at length. My general approach to the topic will be that sketched in chapter eight of my [1995].

9. Overview of a Planning Agent

Rational agents are agents that adopt and execute plans in an intelligent fashion. Such an agent, embedded in a realistically complex world, will reason endlessly, and the bulk of the cognitive work involved in planning and plan-execution will be carried out by reasoning. Reasoning from perceptual input will lead the agent to beliefs that will apprise it of opportunities and lead to the adoption of goals. The adoption of a goal initiates planning. Planning proceeds by posing queries for epistemic cognition regarding how the goal can be achieved. In attempting to find a plan, the agent may be led far afield into epistemic endeavors that are only related to the original planning problem very indirectly. Thus theoretical physics may ultimately emerge from trying to catch lunch. Most (perhaps all) of the complexities of epistemic cognition have their ultimate source in trying to answer the questions that arise in the course of planning and plan-execution. The bulk of cognition will be epistemic cognition, but there remains an ineliminable residue of practical cognition. All epistemic cognition can do is provide us with beliefs about the expected values of plans and about how different plans might be related to one another (and so be in competition in various ways). There are still non-epistemic decisions to be made regarding which plans to adopt.

The most important conclusion of this paper is that planning itself must be done by reasoning in a sophisticated agent. It cannot be done algorithmically, because planning, plan-execution, and epistemic cognition must all be done in unison rather than sequentially. They cannot be performed by isolated modules, because the course of each drives the others and is in turn driven by feedback from the others. A rational agent must be an agent that adopts and executes plans by employing its faculties of epistemic cognition to reason about them.

References

Beetz, Michael, and McDermott, Drew
1996 "Local planning of ongoing activities", *Proceedings of the Third International Conference on Artificial Intelligence Planning Systems.*, ed. Brian Drabble. AAAI Press.
Bratman, M., Isreal, D., and Pollack, M.
1988 "Plans and resource-bounded practical reasoning", *Computational Intelligence* **4**, 349-355.
Fikes, R. E., and Nilsson, N. J.
1971 STRIPS: a new approach to the application of theorem proving to problem solving, *Artificial Intelligence* **2** 189-208.
Gelfond, Michael, and Lifschitz, Vladimir
1993 "Representing action and change by logic programs", *Journal of Logic Programming* **17**, 301-322.
Georgeff, Michael, and Lansky, Amy
1987 "Reactive reasoning and planning", *Proceedings AAAI-87*, 677-682.
Green, C.
1969 "Application of theorem-proving to problem solving". *Proceedings IJCAI-69*, 219-239.
Hanks, Steve, and McDermott, Drew
1986 "Default reasoning, nonmonotonic logics, and the frame problem", AAAI-86.
Lewis, David
1973 *Counterfactuals*. Cambrdige, Mass.: Harvard University Press.
Lin, Fangzhen, and Reiter, Raymond
1994 "How to progress a database (and why) I. Logical foundations." In *Proceedings of the Fourth International Conference on Principles of Knowledge Representation (KR'94).* 425-436.
1995 "How to progress a database II: The STRIPS connection." *IJCAI-95*. 2001-2007.
McAllester, David, and Rosenblitt, David
1991 "Systematic nonlinear planning", *Proceedings of AAAI-91*, 634-639.
McCarthy, John
1977 "Epistemological problems in artificial intelligence". *Proceedings IJCA-77.*
Penberthy, J. Scott, and Weld, Daniel
1992 "UCPOP: a sound, complete, partial order planner for ADL". *Proceedings 3rd International Conference on Principles of Knowledge Representation and Reasoning*, 103-114.
Pollock, John
1984 *The Foundations of Philosophical Semantics*, Princeton University Press.
1987 *Contemporary Theories of Knowledge*, Rowman and Littlefield.
1990 "OSCAR: a general theory of rationality", *Journal of Experimental and Theoretical AI* **1**, 209-226.
1992 "New foundations for practical reasoning", *Minds and Machines*, **2**, 113-144.
1995 *Cognitive Carpentry*, MIT Press.
1996 "Reason in a changing world", in *Practical Reasoning*, ed. Dov M. Gabbay and Hans Jürgen Ohlbach, Springer, 495-509. This can be downloaded from http://www.u.arizona.edu/~pollock/.
1996a "Perceiving and reasoning about a changing world". Technical report of the OSCAR Project. This can be downloaded from http://www.u.arizona.edu/~pollock/.
Sacerdotti, E. D.
1975 The non-linear nature of plans. *Proceedings IJCAI-75.*
1977 *A Structure of Plans and Behavior*. Amsterdam: Elsevier-North Holland.
Shanahan, Murray
1996 "Robotics and the common sense informatic situation", *Proceedings of the 12th European Conference on Artificial Intelligence*, John Wiley & Sons.
Shoham, Yoav
1987 *Reasoning about Change*, MIT Press.
Veloso, Manuela, Carbonell, Jaime, Perez, Alicia, Borrajo, Daniel, Fink, Eugene, and Blythe, Jim

1995 "Integrating planning and learning: the PRODIGY architecture", *Journal of Experimental and Theoretical Artificial Intelligence* **7**.
Vere, S., and Bickmore, T.
1990 "A basic agent", *Computational Intelligence* **6**, 41-60.
Weld, Daniel
1994 "An introduction to least commitment planning", *AI Magazine*, **15** 27-62.
Wooldridge, M., and Jennings, N.
1995 "Intelligent agents: theory and practice", *The Knowledge Engineering Review* **10**, 115-152.