Reprinted from *Proceedings of the DARPA Workshop on Innovative Approaches to Planning, Scheduling and Control*, November, 1990, Katia P. Sycara, editor, San Mateo: Morgan Kaufmann Publishers, pp. 28-36.

Rational Distributed Reason Maintenance for Planning and Replanning of Large-Scale Activities (Preliminary Report)

Jon Doyle*
MIT Laboratory for Computer Science
545 Technology Square
Cambridge, MA 02139
doyle@zermatt.lcs.mit.edu

Michael P. Wellman
USAF Wright R&D Center
WRDC/TXI
Wright-Patterson AFB, OH 45433
wellman@wrdc.af.mil

Abstract

Efficiency dictates that plans for large-scale distributed activities be revised incrementally, with parts of plans being revised only if the expected utility of identifying and revising the subplans improve on the expected utility of using the original plan. The problems of identifying and reconsidering the subplans affected by changed circumstances or goals are closely related to the problems of revising beliefs as new or changed information is gained. But the current techniques of reason maintenance—the standard method for belief revision—choose revisions arbitrarily and enforce global notions of consistency and groundedness which may mean reconsidering all beliefs or plan elements at each step. We outline revision methods that revise only those beliefs and plans worth revising, and that tolerate incoherence and ungroundedness when these are judged less detrimental than a costly revision effort.

1 Introduction

Planning is necessary for the organization of large-scale activities because decisions about actions to be taken in the future have direct impact on what should be done in the shorter term. But even if well-constructed, the value of a plan decays as changing circumstances, resources, information, or objectives render the original course of action inappropriate. When changes occur before or during execution of the plan, it may be necessary to construct a new plan by starting from scratch or by revising a previous plan. In fact, replanning may be worthwhile even when the new situation does not deviate significantly from prior expectations. The original plan may have been constructed to perform acceptably over a wide range of possible circumstances, and knowing more about the particular situation encountered may enable construction of strategies which are better suited to the case at hand.

There are two central decisions surrounding the replanning process. First, given the information accrued during plan execution, which remaining parts of the original plan should be salvaged and in what ways should other parts be changed? Incremental modification is more efficient than wholesale replanning, but a restriction to local changes can compromise the value of the revised plan. Second, to what extent should the planner attempt to avoid the need for replanning by anticipating contingencies and providing for them in the original plan? Contingency planning improves the capacity for response when replanning time is limited, but the return on up-front investment rapidly diminishes as the likelihood of particular contingencies decreases.

In the following, we describe an approach to replanning which addresses the first question by applying the decision-theoretic conception of rationality to the plan revision tradeoff. Characterizing the computational costs and performance of the revision process contributes toward solutions to the second problem, development of a contingency planning strategy. Our techniques center on a reason maintenance system or RMS (also known as TMS for "truth maintenance system" [de Kleer, 1986; Doyle, 1979]), redesigned for more rational and flexible control.

2 Rational replanning

To replan effectively in crisis situations, replanning must be *incremental*, so that it modifies only the portions of the plan actually affected by the changes. Incremental replanning first involves *localizing* the potential changes or conflicts by identifying the subset of the extant beliefs and plans in which they occur. It then involves *choosing* which of the identified beliefs and plans to keep and which to change. For greatest efficiency, the choices of what portion of the plan to revise and how to revise it should be *rational* in the sense of decision theory. This means that the replanner employs expectations about

*Jon Doyle is supported by National Institutes of Health Grant No. R01 LM04493 from the National Library of Medicine.

and preferences among the consequences of different alternatives to choose the best one.

2.1 Explicit and implicit rationality

According to decision theory, a choice is rational if it is of maximal expected utility among all alternatives. But planning and replanning involve at least two different sorts of decisions, and applying the standard of rationality to each yields different notions. The fundamental distinction is that between *result* rationality and *process* rationality. Rationality of result measures how efficiently the plan achieves specified objectives. Complementing this, rationality of process measures how efficiently the planner expends its efforts in constructing the plan. While most investigations of planning have focused on one or the other, both elements are essential to the overall rationality of the planning system.

Making any process rational is not easy, for straightforward mechanizations of decision-theoretic definitions can require more information than is available and more computation than is feasible to use that information. Sophisticated mechanizations are more tractable, but the main tool for achieving rationality in reasoning is to distinguish between *explicit* and *implicit* rationality in processes. Computational mechanisms may calculate and compare expected utilities in order to make explicitly rational choices. Explicit rational choice promises to be most useful in guiding some of the larger meta-level decisions about whether to replan globally or incrementally, and in choosing which contingencies call for planned responses. For the more numerous small decisions that arise, however, explicitly representing and calculating expected utilities may not be worth the cost. Instead, the more useful approach is to apply non-decision-theoretic reasoning mechanisms whose results may be justified as rational by separate decision-theoretic analyses. Such mechanisms may be viewed as "compiling" the results of explicit rational analysis into directly applicable forms. Each of these ways of implementing rationality is best in some circumstances, since compilation is not always possible or worthwhile.

Examples of implicitly rational procedures abound in AI under the name of heuristics. For instance, the "status quo optimality" heuristic [Wellman, 1990a, Section 6.4.1] constrains the set of possible revisions under the assumption that the current plan is optimal. In particular, the replanner need only respond to the specific changes. A related example is application of the basic theorem of optimization that says that if the only change is a tightening of constraints, the currently optimal plan remains optimal if it remains feasible. Another example, of somewhat different character, is provided by the assumptions made by nonmonotonic reason maintenance systems. The default rules or reasons justifying these assumptions are important forms of heuristics, and the RMS examines them to come up with a coherent set of assumptions and logical conclusions. Though the algorithms for determining these sets of conclusions do not involve any explicit rationality calculations, the conclusions drawn by the RMS can be shown to be Pareto optimal sets, that is, rational choices of conclusions when the reasons are interpreted as preferences over states of belief [Doyle, 1985]. Viewed this way, default rules or reasons encode compiled preferences, and reason maintenance is an example of an implicitly rational choice mechanism.

Thus one approach to the application of rationality principles in planning and replanning is to identify the principles and look for computational mechanisms that implement them, preferably implicitly. Another is to develop seemingly effective computational mechanisms and then figure out under what conditions they are rational. We are pursuing both approaches.

2.2 Rational guidance of replanning

Process rationality enters the task of planning in numerous ways. For example, in the development of a plan, contingency plans should be included only when the expected utility of preparing them is sufficiently great: if the contingency is likely to occur and if the costs of developing it in advance are less than the costs of constructing it under the tighter constraints existing while executing the enclosing plan. Similarly, a portion of a large plan should be revised only if, given the new information, the expected costs and benefits of identifying which plan elements need revising outweigh those expected for either using the original portion or replanning from scratch.

Making these judgments requires information about the likelihoods, costs, and benefits of different sorts of contingencies and planning responses. This includes the likelihood of specific contingencies arising, their importance if they do arise, and the costs of planning for them; similarly, the likelihood of one part of the plan being affected by changes in another, the importance of those changes, and the costs of determining and effecting them.

While many of the likelihoods involved in planning derive from the specifications of the task, the costs and benefits of reasoning steps involved in planning are functions of the underlying representational and reasoning architecture. The theory of computation supplies some abstract notions of computational costs, such as worst-case time and space taken by Turing machines. However, significant differences in reasoning time and space can be lost in the translation to Turing machines, and the worst case is not the only one of interest. Use of the theory of rational decisions effectively in making judgments about plan revision requires realistic measures of computational costs and benefits appropriate to the particular architecture of the planner, as well as expectations appropriate to the domain of planning. Our development of the planning architecture attempts to make formalization and estimation of these measures more direct.

Process rationality must be evaluated with respect to the combined planning/replanning system. In our model of the plan construction process, depicted in Figure 1, the planner and replanner continually evaluate and re-

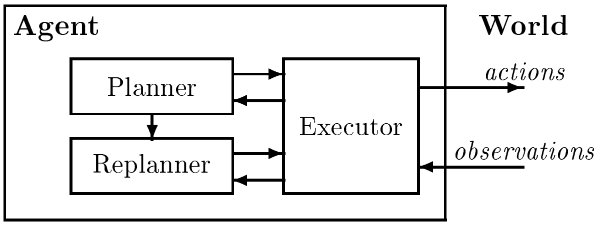

Figure 1: An integrated planning, replanning, and execution system.

vise the existing plan in light of what happens in the world. The distinction between planning and replanning is that the latter uses the existing plan to focus attention on a restricted set of decisions about actions to be performed. The tight coupling of the planning and replanning modules is indicative of the strong interactions between their designs. Knowledge about the capability of the replanner dictates where up-front planning effort should be spent anticipating particular contingencies. And the replanner requires access to the planner's reasons for adopting the current strategy in order to intelligently adapt it for changing situations.

To do this, the planning procedures routinely identify the assumptions made during planning and connect plan elements with these assumptions. In addition, to achieve true flexibility in the sorts of changes the replanner can accommodate, we permit any element of information to change, including the problem specification, background knowledge, and preferences. This allows the replanner to benefit from knowledge of what other specifications, beliefs, and preferences were adopted as consequences or choices from the changed items. This also makes it important that implicitly rational planning procedures indicate the original expectations, preferences, and subplans from which they were "compiled."

3 Planning framework

Our approach combines a dominance-proving architecture for planning [Wellman, 1990a] with a reason maintenance facility for replanning. We start from a constraint-posting view of the plan construction process. Plans consist of a set of *actions*, which can be specified at varying levels of detail. Constraints posted by the planner dictate the inclusion or exclusion of particular actions, and specify features of the actions included. For example, unary constraints on an action may determine the resources allocated to it, its spatiotemporal location, or some other details about its implementation process. Inter-activity constraints may identify shared objects or establish temporal relations among actions. The class of expressible constraints defines the plan construction language. The planning language itself is a restricted subset of this, limited by input requirements of the execution module.

Each posted constraint represents a *decision* made by the planner, choosing the class of plans satisfying the constraint over those that do not. To support rationality in planning, we require that every decision be associated with a reason, of one of the following types:

1. *Dominance* reasons indicate decision-theoretic arguments that plans violating the constraints are inadmissible [Wellman, 1987].
2. *Feasibility* reasons justify posting constraints because they are required for plan executability. For example, we must enforce preconditions of included actions.
3. *Completeness* reasons indicate the constraints are required to fill out plans so that they can be interpreted by the execution module. For example, all shipment actions must specify a source and destination. The degree of incompleteness permitted depends on the reactive capabilities of the executor.
4. *Default* reasons directly associate decisions with other conditions on the planning situation. While all planning decisions are defeasible, we distinguish those not based on explicit rationality arguments.

All reasons specify the beliefs, preferences, and other planning decisions on which they depend. Because these elements in turn are supported by reasons, the composite argument for a planning decision can include a variety of these justification types. For example, a decision might be derived from a decision-theoretic dominance proof with some premises representing default intentions premised on some default intentions which in turn were triggered by the need to complete an insufficiently specified action description.

The dominance-proving architecture offers several advantages as the basis for a rational replanning system. Foremost, it accommodates use of decision-theoretic criteria for choice among plans, which is the central basis of result rationality. In addition, its dominance relation is defined over abstract plan classes, so that these criteria can be associated with isolated planning decisions (that is, individual constraints). Attaching reasons to dominance conditions generalizes the current architecture and directs the replanner to the appropriate regions for modification when things change.

Though recording the reasons for plans is a first step towards efficient incremental replanning, this alone is not sufficient, as we see by a closer examination of reason maintenance techniques.

4 Replanning and reason maintenance

The problem of revising plans to account for changed conditions has much in common with backtracking and the problem of revising beliefs in light of new information. In both cases, one must determine which existing beliefs or plans are in conflict with the new information, what these existing beliefs or plans depend on, and what gaps in plans or beliefs appear as the revisions or updates are made. That is, one must localize the potential changes or conflicts by identifying the subset of the extant beliefs and plans in which they occur. Similarly, both belief revision and plan revision involve choosing which of the identified beliefs and plans to keep and which to change. In addition, the problem of providing for contingencies has much in common with the problem of choosing rules for reasoning by default, for both involve setting up primary plans or beliefs and the secondary plans or beliefs to use when the primary ones are not applicable. In both plan revision and belief revision, we seek to make these choices of where and how to revise rational in the sense of decision theory.

The standard approach to belief revision, backtracking, and default reasoning is to use a reason maintenance system to connect original information with derived conclusions and assumptions. Reason maintenance may be used in a similar way to revise plans as well as beliefs by indicating the dependence of plans on beliefs and on other plans, thus indicating the relevant portions for revision and the conflicts between prior plans and new circumstances. This possibility was, in fact, one of the original motivations for reason maintenance systems (see [de Kleer *et al.*, 1977]).

4.1 Rational reason maintenance

But the extant architectures for reason maintenance require reassessment. In the first place, essentially all the choices made by current RMSs are irrational since they are made without reference to any preferential information about what choices are better than others. The most obvious decisions concern backtracking: whether observed conflicts warrant resolution and if so, which assumption to retract in order to resolve them. Approaches to each of these decisions play prominent roles in the design of different reason maintenance systems. But if we are to achieve the efficiency required for revising large plans, reason maintenance must be redesigned to make these choices rationally whenever possible. Accordingly, we have begun to develop formal foundations for the theory of rational belief revision [Doyle, 1988; Doyle, 1990], and are developing techniques for encoding probabilistic and preferential information within the RMS and methods by which the RMS can use this information to backtrack in a rational manner. In this, we build on techniques for qualitative representation of probabilistic information [Wellman, 1990b].

But to really make reason maintenance techniques efficient, we must do more than choose rationally among assumptions in backtracking. We must in addition undertake a fundamental reconsideration and redesign of reason maintenance systems to make them much more incremental than extant architectures. Current algorithms for revising beliefs are based on making unbounded (potentially global) optimizing computations that in some cases may reconsider the status of *every* item in the plan and knowledge base, even though very few of these statuses may change as the result of the revision. Put another way, extant systems maintain global coherence (propositions are believed if and only if there is a valid argument for them) and global groundedness (all believed propositions have a well-founded argument from premises). While these unbounded computations have been manageable in the relatively small knowledge bases explored to date, they would appear to be infeasible for use in systems manipulating very large plans. Instead of global computations, we need some way of controlling how much effort is spent on revision. If reason maintenance is to be rational, the system must be able to trade off coherence and groundedness for time or other resources. Specifically, it must be able to decide whether the benefits of updating some arguments or consequences justify the costs of updating them.

To make the RMS amenable to rational control, we divide the knowledge base into parts, each of which may be revised or preserved separately. Each module of this *distributed* RMS contains its own set of beliefs and plans (as well as other information) corresponding to different elements and purposes of the overall plan or to different dimensions of structure (hierarchical abstraction, overlapping views, spatial separation, temporal separation, flow of material and information, etc.). Decomposition of knowledge in this way is a familiar element of many representational schemes (e.g., those based on Minsky's [1975] original frame-systems idea). The use of locality in planning is illustrated most explicitly by the encapsulation mechanisms of Lansky's [1988] GEMPLAN system.

4.2 Distributed reason maintenance

Along with the general benefits of decomposition, there are several additional reasons for distributing reason maintenance across different processors. In the first place, the information and effort required may be too great to store or perform on a single machine. In managing very large activities, for example, the most effective representations may spread information across machines or storage media of different speeds and access times (e.g., disk storage, large spatial separations). Even when the information resides on a single processor, the most convenient representation may be a modular, distributed organization as described above. But more generally, the information and actions involved in some task may be naturally distributed. For example, the necessary information may come from geographically separated sensors or databases. If communication is either unreliable or costly, effective action may require on-site processing. Similarly, there may be numerous people or devices carrying out parts of the task. For example, in the task of operating a large manufacturing complex, plans are executed by line or cell managers acting independently except as coordinated by the plan. When changes occur, at least some of the changes in plan must be determined by the line or cell managers, since the complex manager will not be able to keep track of all of the activities or to respond quickly enough. Because authority is delegated and distributed, reactions to deviations may be completely decentralized and uncoordinated.

In addition, distributed reason maintenance may be valuable because different beliefs and plans may serve different purposes. These purposes may dictate careful maintenance of some beliefs and more casual maintenance of others. A common case of this arises when reasoning is accomplished by different modules operating at different rates. Even if they share a common database, it is often natural to view each module as having distinct inputs, outputs, and local state. In this setting, different rates of inference or action in the modules call for differing treatment of the information in computing updates and checking support. For example, outputs which change rapidly compared with how often they are used as inputs need not demand reconsideration of consequences each time they change. Instead, it may be much more efficient to leave the consequences untouched and to have the consuming module recheck the support immediately prior to use—and then only if the risks of unjustified action are great enough. In many cases, we may expect that the success of the overall plan will not be adversely affected if the beliefs of one module about plans involving some distant module are mistaken.

For example, suppose one part of a manufacturing plan calls for receiving parts from San Diego at Los Angeles and then flying them to Detroit. If local difficulties promise to delay the parts from San Diego, the origination portions of the plan might be revised to reroute similar parts in San Francisco to Los Angeles. As long as this plan patch attaches appropriate shipping orders for the Los Angeles authorities, there is no need to notify them in advance about the change in plans. Indeed, if the origination plans change several times (say from San Diego to San Francisco, back to San Diego, etc.), notifying Los Angeles in advance just leads to wasted effort in revising the latter portion of the plan.

4.3 The reason maintenance service

The extant RMS architectures make reason maintenance the base-level stratum upon which all other reasoning procedures are erected. To enable belief revision, one must encode every bit of information that might change in reasons and tell these reasons to the RMS (cf. [Rich, 1985; Vilain, 1985]). This can present an excessive burden, as manifest by the observation that the RMSs supplied in expert system shells all too often go unused. If one must apply it to every step of reasoning, at every level down to the smallest inference, reason maintenance becomes a demanding duty rather than a flexible service to use or ignore as appropriate. To integrate existing application tools and systems that do not use reason maintenance into AI systems that do, the RMS must be able to use other databases and processes to effect its revisions. In particular, the RMS must be able to treat external databases as the authorities about certain beliefs, and it must be able to operate even though other processes may be changing these databases independently of the RMS. This makes the RMS just one of a set of distributed databases.

5 Rational distributed reason maintenance

Putting these observations together, we seek to facilitate revision of large plans by employing a *rational distributed reason maintenance service*, or RDRMS. The purpose of the RDRMS is to maintain a description of the overall system's state of belief that is as good as possible given the reasoner's purposes and resources. This description may be approximate, partial, or imperfect, and it may be improved by performing further computation as the resources supplied to the RDRMS increase.

There are many motivations for using an RMS: as a way of providing explanations, as a way of answering hypothetical questions, and as a way of maintaining coherence, groundedness, and consistency. These also motivate the RDRMS, but its primary purpose is to enable the reuse of past computations in whole or in part without having to repeat the possibly lengthy searches that went into constructing their results. That is, we view reasons as information about past computations or conditions which may be used to reconstruct results in changed circumstances, either exactly or in modified form (as in derivational analogy [Carbonell, 1986] or case-based reasoning). Treating reasons as aids to recomputation is in marked contrast with the traditional use of reasons in RMSs, where they are treated as rigid requirements that belief states must satisfy instead of information which may be used or ignored as suits the reasoner's purposes. Naturally, in this setting the RDRMS is not expected to determine completely and accurately what the system believes. Instead, it only offers a theory of what the overall system believes—an "autoepistemic" theory, in the sense of Moore [1985], but not necessarily a complete or correct one.

5.1 RDRMS Operations

The basic operation of the RDRMS is to record reasons and other information, and, when so instructed, to revise beliefs in accordance with the expectations and preferences supplied by the reasoner. Put another way, the default operation of the RDRMS is to ignore the information it records until it is told to revise beliefs, and then to revise them only as far as can be justified by purposes of the reasoner. We do not require that all inference be rationally controlled. Some amount of automatic inference is acceptable if it represents strictly bounded amounts of processing.

In the RDRMS, reasons are ordinarily partial. That is, the reasoner need not register all inferences with the RDRMS. The RDRMS will therefore be unable to track all the consequences of all beliefs. Although knowledge is usually preferable to ignorance, this incompleteness of the beliefs of the RDRMS need not be detrimental since the underlying knowledge and inferences of the reasoner are incomplete anyway. Moreover, these consequences may not influence the reasoner's actions, in which case all effort expended in recording them would be wasted. The only discipline required of the reasoner is that any inferences that will not be performed by some other agency and that cannot be determined after the fact during backtracking should be described to the RDRMS.

Correspondingly, reasons may be incorrect in the RDRMS. That is, the reasoner may use a reason to describe the result of a computation, but may leave out some underlying assumptions. The result is a reason that is valid when those unstated assumptions hold, but which may be invalid otherwise. Incorrect reasons can be very troublesome in a traditional RMS, since they would be enforced as requirements on the state of belief, but they need not cause special problems in the RDRMS. Since the RDRMS may obey or ignore reasons depending on its instructions and experience, all reasons are implicitly defeasible. Thus incorrect reasons pose no problems not already present in explicitly defeasible nonmonotonic reasons.

Just as reasons may be incomplete, so may be the theories of belief states constructed from them, since if reasons are ignored, their consequences will not be believed. More generally, the RDRMS makes it possible to vary how many conclusions are drawn from reasons. For example, the system will ordinarily use reasons to construct a single global set of beliefs, as in the original RMS. But for some specific sets of reasons, say those corresponding to a circumscribed problem, the RDRMS may determine all consistent sets of beliefs as in the ATMS [de Kleer, 1986]. Alternatively, only some consistent interpretations may be constructed, such as those maximal in some order (as in preferential nonmonotonic logics [Shoham, 1988]). In general, the aim is to use the recorded reasons to draw as many conclusions as the reasoner needs.

Similarly, the revisions performed by the RDRMS may be incomplete. In the absence of more specific instructions, the default revision is trivial, simply adding the new reasons and their immediate conclusions to the belief set. (In recognition of the partiality of reasons, the RDRMS also accepts commands to simply believe some proposition, independent of reasons. This corresponds to the "revision" operation in philosophical treatments of belief revision [Gärdenfors, 1988].) Specifically, without explicit instructions, the RDRMS does not propagate changes, does not ensure beliefs are grounded, and does not automatically backtrack to remove inconsistencies. To give some structure to these operations, we define revision instructions relative to the modules of the knowledge base. These instructions may indicate that changes should propagate within the module containing the belief, or to its neighbors, or globally; or that all beliefs in the module should be grounded with respect to the module, with respect to its neighbors, or globally; or that backtracking should be confined to the module, or should look further afield for assumptions to change.

5.2 RDRMS Behavior

One consequence of the incompleteness and incorrectness of reasons is that beliefs of the system may be inconsistent in routine operation. The overall set of beliefs may exhibit inconsistencies by including conflicting beliefs from different modules. Ordinarily the specialized beliefs corresponding to specific problems or subjects will be represented in modules that are internally consistent, but the RDRMS need not be forced to keep all these modules consistent with each other. In this case, the locally coherent modules can be interpreted as "microtheories" [Hewitt, 1986] (related to the idea of "small worlds" in decision theory [Savage, 1972]). But inconsistency can arise even within a module if too little inference is specified.

Another consequence is that the beliefs of the system may not be fully grounded. In the first place, the set of beliefs may be so large as to make global groundedness too costly. More fundamentally, large sets of beliefs always contain interderivable sets of propositions—alternative definitions provide the most common example—and which of these sets to choose as axioms can depend on the specific reasoning task being addressed. For example, the standard definition of nonplanar graphs is best for some purposes (e.g., teaching the concept), but Kuratowski's characterization is best for other purposes (e.g., recognition algorithms). Thus lack of global groundedness need not be cause for alarm. Ordinarily, however, specialized modules corresponding to specific problems will be kept grounded in the axioms formulating these problems. The system of beliefs can thus be thought of as "islands" of groundedness floating in a sea of ungrounded beliefs.

The aim of the RDRMS is to make all of its choices as rationally as possible. These include the choices of which reasons to use in reconstructing results, whether to propagate changes, whether to ground a conclusion, and whether to backtrack. Since reasons merely record some of the inferential history of the reasoner, they do not by themselves determine whether consequences are updated or supports are checked. Instead, to make these decisions the RDRMS uses annotations supplied by the reasoner which give instructions, expectations, and preferences about alternative courses of action. These include specification of the conditions under which the RDRMS should pursue consequences and check support. For example, local propagation may be expressed as processing changes within the module containing the changed belief, but not externally. Alternatively, changes might be communicated to neighboring modules (with or without local propagation). Other regimes are possible too, including the extreme of propagating the change globally. Similarly, the annotations may indicate to persist in believing the proposition without reevaluating the supporting reason, to check that the reason is not invalidated by beliefs within the containing module, or to check validity with respect to external beliefs.

It is this limited scope, along with the variety and fine grain of RDRMS operations, that makes the service amenable to rational control. For decisions about updating consequences and checking support, it is important that the individual operations be well-characterized computationally. Domain knowledge of probabilities and preferences should also be reflected in the revision policies. Because such information is not always available, the architecture provides default choices for each of these classes of decisions. Each domain may override these with other defaults that are more appropriate in its specific area. These default choices are then used whenever there is no evidence that a decision requires special treatment.

In addition to these decisions within the RDRMS, there are choices about whether to record specific reasons and about which propositions to adopt or abandon as premises of different modules. At present, the RDRMS embodies the same approach as do traditional RMSs, namely that these decisions are the responsibility of the external system (or systems). But since these decisions sometimes can depend on what reasons have already been recorded, we are investigating techniques by which the RDRMS can make some of these decisions for the external reasoner when the external reasoner informs the RDRMS of its purposes. Of course, these decisions may also depend on other facts, such as how hard it was for the reasoner to discover the belief, so we cannot expect the RDRMS to make all such decisions on its own.

6 Comparison with other work

Reason maintenance is the standard approach to belief revision, backtracking, and default reasoning [de Kleer *et al.*, 1977; Doyle, 1979; Goodwin, 1987]. Morris [1988] has shown that a standard RMS can support planning and dependency-directed replanning within the classical planning framework. But developing an architecture for reason maintenance and replanning subject to rational control will require significant modification of existing techniques.

As mentioned above, we use the RDRMS to extend the dominance-proving architecture for planning with partially satisfiable goals [Wellman, 1990a]. This decision-theoretic approach fits well with our goal of rational planning. We also make use of the methods, currently under active investigation, for decision-theoretic control of reasoning, in which the reasoner explicitly estimates and compares the expected utilities of individual search or inference steps [Dean, 1990; Horvitz *et al.*, 1989; Russell and Wefald, 1989]. These are very important in making some of the larger, nonroutine decisions arising in the planning and belief revision tasks. But our aim is to identify implicitly rational decision-making procedures whenever possible by separate, off-line decision-theoretic analyses based on the computational tradeoffs associated with RDRMS operations.

In the traditional, generative approach to planning, the planner takes an initial state and a goal, and constructs a sequence (or partially ordered set) of actions to achieve the goal. Most work on generative planning has concentrated on planning from scratch, though the replanning task has been studied off and on over the years [Fikes *et al.*, 1972; McDermott, 1978; Wilkins, 1988] with some success. But generative planning has focused—with a few recent exceptions—on planning without probability or utility information. Many of these techniques therefore require some reworking before they can be said to produce rational plans, and the issue of rational control of the planning process is just now beginning to be studied [Dean, 1990; Smith, 1988].

Another approach is the "reactive" approach to planning and action, which seeks to avoid execution-time planning by "compiling" all necessary behaviors into directly applicable forms [Brooks, 1986; Georgeff and Lansky, 1987; Rosenschein and Kaelbling, 1986; Schoppers, 1987]. Our approach fits well with such compilation, since we seek to develop implicitly rational planning and decision-making procedures. More specifically, decision-theoretic analyses of planning and replanning apply also to the tradeoff between planning and reacting. Since providing a compiled response for every contingency is usually not feasible, our approach is to provide explicit contingency procedures only when they increase the expected utility of the overall plan, taking both planning effort and execution-time utilities into account. In addition, our assumption of distributed execution authorities and replanners explicitly accounts for the reactive abilities of the distributed execution modules.

The constraint-based approach to scheduling [Fox, 1987] complements the generative planning approach in many ways, as it does focus on issues of utility and optimization. At the same time, it has somewhat lower aspirations, since the focus is on scheduling activities within the confines of an overall plan, rather than on selecting the activities in the first place. In addition, it has generally not addressed issues of probability, and its concepts of preference are not directly translatable to expected utility. But many of the fundamental optimization techniques have been refined and integrated with AI reasoning techniques in this area, and we will draw on these in constructing methods for rational control of the planning process and construction of rational plans.

The case-based approach to planning [Collins *et al.*, 1989; Hammond, 1986; Minton *et al.*, 1989] shares with ours the aim of incremental construction and repair of plans. Some case-based reasoners make significant use of recorded reasons for beliefs and plans, for example Carbonell's [1986] method of derivational analogy. By and large, however, most work on case-based reasoning focuses on issues of conceptual organization and retrieval rather than reason maintenance. In addition, it is not too inaccurate to say that research on case-based reasoning has largely ignored issues of rationality. Work in this area has generally aimed to make all planning operations habitual, so that plans are constructed simply by remembering old plans or plan fragments, along with patches that should be applied to these plans for specific circumstances. We also aim to develop habitual rules for plan construction whenever possible (for example, default plans and default decisions in guiding planning), but to produce and apply these rules in a principled way amenable to formal analysis and directed improvement. In particular, we use the same probabilities that guide decision-making in novel circumstances to also guide the formation and memorization of habitual rules, remembering and forgetting rules and past plans based on estimates of their incremental expected utility. We believe our approach will make it easier to combine techniques from the case-based literature with the more formal techniques developed in the generative planning and constraint-based scheduling literatures.

Most work on distributed AI has not addressed issues of belief or plan revision, focusing instead on distributing the effort involved in ordinary reasoning and planning [Bond and Glasser, 1988]. Very recently, however, some distributed RMSs have been developed. While these represent important first steps, they are not at present suitable bases for rational plan revision. For example, the distributed nonmonotonic TMS of Bridgeland and Huhns [1990] ensures global consistency among different agents about the information they share. Maintaining this degree of coherence is not always feasible in large databases, nor even desirable in cases in which the various agents have different information sources and perspectives. Another relevant work is the distributed ATMS of Mason and Johnson [1989]. This system permits a large degree of inconsistency among the different knowledge-bases, and so is closer to the aims of the RDRMS. But their system also does not address the issue of rationality, and limits the representation of reasons to monotonic implications.

7 Conclusion

Reason maintenance promises to play an important role in replanning, but to prove useful for large-scale activities, the techniques must be capable of incremental application that does not incur the costs of global reconsideration. Furthermore, reasons must reflect likelihoods and preferences about events related to the activity, and revision policies must be sensitive to computational tradeoffs inherent in the process of modifying plans and beliefs.

To support this behavior, we extend traditional reason maintenance techniques to make use of instructions, expectations, and preferences in deciding how to establish and revise beliefs and plan elements. In our conception, the *rational distributed reason maintenance service* maintains only as much coherence and grounded support as is called for by the planner's purposes. In essence, the fundamental operations of finding supporting arguments and pursuing consequences become flexible rather than routine, with different sorts of reasons indicating different sorts of processing during revisions.

Together with the dominance-oriented approach to decision-theoretic planning, the RDRMS represents a general architecture for reasoned replanning of large-scale activities. Although much remains to be worked out, the RDRMS concept provides both a tool for investigating representational issues in belief and preference specification and an analytical framework for studying computational issues in revising beliefs and plans. Because the issues of rationality highlighted by this approach are generally not even expressible within standard RMSs and classical models of planning, we expect this line of research to yield some new insights into the dynamics of the planning and replanning process.

References

[Bond and Glasser, 1988] Alan Bond and Les Glasser, editors. *Readings in Distributed Artificial Intelligence.* Morgan Kaufmann, San Mateo, CA, 1988.

[Bridgeland and Huhns, 1990] David Murray Bridgeland and Michael N. Huhns. Distributed truth maintenance. In *Proceedings of the National Conference on Artificial Intelligence.* AAAI, 1990.

[Brooks, 1986] Rodney A. Brooks. A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation*, 2:14–23, 1986.

[Carbonell, 1986] Jaime G. Carbonell. Derivational analogy: A theory of reconstructive problem solving and expertise acquisition. In Ryszard S. Michalski, Jaime G. Carbonell, and Tom M. Mitchell, editors, *Machine Learning 2.* Morgan Kaufmann, 1986.

[Collins *et al.*, 1989] Gregg Collins, Lawrence Birnbaum, and Bruce Krulwich. An adaptive model of decision-making in planning. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, pages 511–516, 1989.

[de Kleer, 1986] Johan de Kleer. An assumption-based TMS. *Artificial Intelligence*, 28:127–162, 1986.

[de Kleer *et al.*, 1977] Johan de Kleer, Jon Doyle, Guy L. Steele Jr., et al. AMORD: Explicit control of reasoning. In *Proceedings of the ACM Symposium on Artificial Intelligence and Programming Languages*, pages 116–125, 1977.

[Dean, 1990] Thomas Dean. Decision-theoretic control of inference for time-critical applications. Technical Report CS-90-44, Department of Computer Science, Brown University, Providence, RI, 1990.

[Doyle, 1979] Jon Doyle. A truth maintenance system. *Artificial Intelligence*, 12(2):231–272, 1979.

[Doyle, 1985] Jon Doyle. Reasoned assumptions and Pareto optimality. In *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*, pages 87–90, 1985.

[Doyle, 1988] Jon Doyle. Artificial intelligence and rational self-government. Technical Report CS-88-124, Carnegie-Mellon University Computer Science Department, 1988.

[Doyle, 1990] Jon Doyle. Rational belief revision. In *Proceedings of the Third International Workshop on Nonmonotonic Reasoning*, Stanford Sierra Camp, CA, June 1990.

[Fikes *et al.*, 1972] Richard E. Fikes, Peter E. Hart, and Nils J. Nilsson. Learning and executing generalized robot plans. *Artificial Intelligence*, 3:251–288, 1972.

[Fox, 1987] Mark S. Fox. *Constraint-Directed Search: A Case Study of Job-Shop Scheduling.* Pitman and Morgan Kaufmann, 1987.

[Gärdenfors, 1988] Peter Gärdenfors. *Knowledge in Flux: Modeling the Dynamics of Epistemic States.* MIT Press, Cambridge, MA, 1988.

[Georgeff and Lansky, 1987] Michael P. Georgeff and Amy L. Lansky. Reactive reasoning and planning. In *Proceedings of the National Conference on Artificial Intelligence*, pages 677–682, 1987.

[Goodwin, 1987] James W. Goodwin. *A theory and system for non-monotonic reasoning.* PhD thesis, Department of Computer and Information Science, Linköping University, Linköping, Sweden, 1987. Linköping Studies in Science and Technology, No. 165.

[Hammond, 1986] Kristian Hammond. *Case-based Planning: An Integrated Theory of Planning, Learning and Memory.* PhD thesis, Yale University, 1986.

[Hewitt, 1986] Carl Hewitt. Offices are open systems. *ACM Transactions on Office Information Systems*, 4:271–287, 1986.

[Horvitz *et al.*, 1989] Eric J. Horvitz, Gregory F. Cooper, and David E. Heckerman. Reflection and action under scarce resources: Theoretical principles and empirical study. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, pages 1121–1127, 1989.

[Lansky, 1988] Amy L. Lansky. Localized event-based reasoning for multiagent domains. *Computational Intelligence*, 4:319–340, 1988.

[Mason and Johnson, 1989] Cindy L. Mason and Roland R. Johnson. DATMS: A framework for distributed assumption based reasoning. In Les Gasser and Michael N. Huhns, editors, *Distributed Artificial Intelligence*, chapter 13, pages 293–317. Morgan Kaufmann, San Mateo, CA, 1989.

[McDermott, 1978] Drew McDermott. Planning and acting. *Cognitive Science*, 2:71–109, 1978.

[Minsky, 1975] Marvin Minsky. A framework for representing knowledge. In Patrick Henry Winston, editor, *The Psychology of Computer Vision*, chapter 6, pages 211–277. McGraw-Hill, 1975.

[Minton *et al.*, 1989] Steven Minton, Jaime Carbonell, Craig Knoblock, et al. Explanation-based learning: A problem-solving perspective. *Artificial Intelligence*, 40:63–118, 1989.

[Moore, 1985] Robert C. Moore. Semantical considerations on nonmonotonic logic. *Artificial Intelligence*, 25:75–94, 1985.

[Morris, 1988] Paul Morris. Truth maintenance-based planning with error recovery. In *Proceedings of the Rochester Planning Workshop*, pages 18–19, 1988. Extended Abstract.

[Rich, 1985] Charles Rich. The layered architecture of a system for reasoning about programs. In *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*, 1985.

[Rosenschein and Kaelbling, 1986] Stanley J. Rosenschein and Leslie Pack Kaelbling. The synthesis of digital machines with provable epistemic properties. In Joseph Y. Halpern, editor, *Theoretical Aspects of Reasoning About Knowledge: Proceedings of the 1986 Conference*, pages 83–98. Morgan Kaufmann, 1986.

[Russell and Wefald, 1989] Stuart Russell and Eric Wefald. Principles of metareasoning. In *First International Conference on Principles of Knowledge Representation and Reasoning*, pages 400–411, 1989.

[Savage, 1972] Leonard J. Savage. *The Foundations of Statistics.* Dover Publications, New York, second edition, 1972.

[Schoppers, 1987] M. J. Schoppers. Universal plans for reactive robots in unpredictable environments. In *Proceedings of the Tenth International Joint Conference on Artificial Intelligence*, pages 1039–1046, 1987.

[Shoham, 1988] Yoav Shoham. *Reasoning about Change: Time and Causation from the Standpoint of Artificial Intelligence.* MIT Press, 1988.

[Smith, 1988] David E. Smith. A decision theoretic approach to the control of planning search. Technical Report LOGIC-87-11, Department of Computer Science, Stanford University, 1988.

[Vilain, 1985] Marc B. Vilain. The restricted language architecture of a hybrid representation system. In *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*, pages 547–551, 1985.

[Wellman, 1987] Michael P. Wellman. Dominance and subsumption in constraint-posting planning. In *Proceedings of the Tenth International Joint Conference on Artificial Intelligence*, pages 884–890, 1987.

[Wellman, 1990a] Michael P. Wellman. *Formulation of Tradeoffs in Planning Under Uncertainty.* Pitman and Morgan Kaufmann, 1990.

[Wellman, 1990b] Michael P. Wellman. Fundamental concepts of qualitative probabilistic networks. *Artificial Intelligence*, 1990.

[Wilkins, 1988] David E. Wilkins. *Practical Planning.* Morgan Kaufmann, Los Altos, CA, 1988.