Principles of Metareasoning *

Stuart Russell and Eric Wefald
Computer Science Division
University of California
Berkeley, CA 94720

This paper is dedicated to the memory of Eric Wefald

Abstract

In this paper we outline a general approach to the study of metareasoning, not in the sense of explicating the semantics of explicitly specified meta-level control policies, but in the sense of providing a basis for selecting and justifying computational actions. This research contributes to a developing attack on the problem of resource-bounded rationality, by providing a means for analysing and generating optimal computational strategies. Because reasoning about a computation without doing it necessarily involves uncertainty as to its outcome, probability and decision theory will be our main tools. We develop a general formula for the utility of computations, this utility being derived directly from the ability of computations to affect an agent's external actions. We address some philosophical difficulties that arise in specifying this formula, given our assumption of *limited* rationality. We also describe a methodology for applying the theory to particular problem-solving systems, and provide a brief sketch of the resulting algorithms and their performance.

*This research has been supported by an equipment grant from the AT&T Foundation, and by funding from the Lockheed AI Center, California MICRO Program, and the National Science Foundation under grant no. IRI-8903146. Eric Wefald was supported by a GE Foundation Fellowship and more recently by a Shell Foundation Doctoral Fellowship. We gratefully acknowledge this assistance. In addition, we would like to thank Steve Bradley, Jack Breese, Murray Campbell, Jon Doyle, Michael Fehling, Michael Genesereth, Eric Horvitz, Maurice Karnaugh, Richard Karp, David McAllester, Devika Subramanian, Michael Wellman and two reviewers for their valuable comments and suggestions.

1 Introduction

> Blot out vain pomp; check impulse; quench appetite; keep reason under its own control.
>
> *Marcus Aurelius Antoninus*

The study of resource-bounded intelligent systems promises to be a major area of research in AI in the near future, with the potential for drastic revision of our understanding of learning, inference and representation. One reason for this is practical: few people believe that classical, normative models can scale up. A second, and more fundamental, reason is that existing formal models, by neglecting the fact of limited resources for computation, fail to provide an adequate *theoretical* basis on which to build a science of artificial intelligence. The 'logicist' approach to AI, exemplified by McCarthy's Advice Taker proposal [38], emphasizes the ability to reach correct conclusions from correct premises. The 'rational agent' approach, derived from philosophical and economic notions of rational behaviour, emphasizes maximal achievement of goals via decisions to act. When resource bounds come into play, direct implementation of either approach results in suboptimal performance. Instead, what we want is an optimal design for a *limited* rational agent. A view of artificial intelligence as a *constrained* optimization problem may therefore be more profitable.[1] The solutions to such a constrained design problem may look very different from those provided by the deductive and decision-theoretic models for the unconstrained problem.

Our long-term project is the design of robust software architectures for goal-driven, resource-limited intelligent systems (also known as *ralphs*[2]). Our approach has been to address the design problem with the finitude of resources as a *starting point*, rather than trying to lop corners off the deductive model. A major tool of this research is a normative meta-level theory for the value of computations, since this allows an agent to allocate scarce computational resources optimally; alternatively, it allows the AI researcher to show the optimality of a non-optimizing algorithm for decision-making, and to design such algorithms constructively. Progress on developing and applying such a theory forms the main subject matter of this paper. Given the absence of a satisfactory axiomatic system for computationally limited agents, our results have only a heuristic basis, strictly speaking. However, the general methodology and the structure of the theory should remain in place even when an axiomatic approach is found, and by taking the development all the way to practical implementations we have shown that there are no fundamental limitations to the realization of a theory of metareasoning.

We begin in Section 2 by defining the notion of real-time problem-solving, and discuss the various approaches that have been taken to the problem of boundedness in this context.

[1] This view is developed in greater depth in [27].

[2] Ralphs inhabit the RALPH (Rational Agents with Limited Performance Hardware) project at Berkeley.

We see that metareasoning has a vital role to play, along with meta-level learning and compilation. Section 3 introduces the idea of *rational* metareasoning, wherein computations are treated as actions to be selected among on the basis of their expected utilities. In turn, these utilities are derived from the expected effects of the computations, chief among which are the consumption of time and/or space and the possible revision of the agent's intended actions in the real world. We define three levels of analysis, specifying the performance system at increasing levels of detail. Section 4 sets up the equations for the most general level, and Section 5 covers the case of bounded agents that use utility estimates to decide on actions. Along the way, we also examine certain ways in which the assumption of limits on the agent's rationality forces us to depart from the standard axiomatic framework for decision theory. Section 6 outlines a methodology for applying the theory to specific classes of decision-making systems, and mentions the results we have obtained for search applications. While the framework we have developed allows the analysis and construction of a variety of bounded rational systems, we do not deal with rational selection of all types of computation; for example, learning by induction and compilation is not covered explicitly. Section 7 discusses these and other directions for future research.

2 Approaches to bounded rationality

Two conditions conspire to create what has been called the 'finitary predicament' [6]: first, real agents have only finite computational power; second, they don't have all the time in the world. One characterization of a 'real-time' problem situation is the following: the utility of performing a given action varies significantly over the time necessary for a complete solution to the decision problem. Typically, the utility of an action will be a decreasing function of time. Since the time at which an action can be carried out depends on the amount of deliberation required to choose the action to be performed, there is often a tradeoff between the *intrinsic utility* of the action chosen and the *time cost* of the deliberation (see section 5.2 below). As AI problems are scaled up towards reality, virtually all situations will become 'real-time'. As a result, system designs will have to be sufficiently flexible to manage such tradeoffs. One aim of this paper is to develop a methodology for constructing real-time algorithms that can be used as the 'building blocks' for more complex systems. The composition problem for real-time systems is discussed in section 7.

Standard algorithms in computer science either maximize intrinsic utility with little regard for the time cost, or minimize the time cost for achieving some fixed level of intrinsic utility. Work in *real-time AI* has traditionally followed a variant of the latter approach, focusing on delivering AI capabilities in applications demanding high performance and negligible response times. As a result, designers typically choose a fixed level of output quality, and then perform the necessary precompilation and optimization to achieve that level within a fixed time limit. Laffey *et al.* [32] survey a large number of application programs for real-time AI, and note, somewhat despairingly, that "Currently, *ad hoc* techniques are used for making a system produce a response within a specified time interval."

The inappropriateness of solution optimality as a criterion for success has been recognized in theoretical computer science. As exact methods have been shown to be intractable for a wide range of problems, theorists have looked at algorithms with one of two properties:

1. A guarantee that the solution returned will be within some ϵ (either relative or absolute) of the optimal solution. This is called *approximation*.

2. A probability of at least $1 - \delta$ that the algorithm will return the correct or optimal solution. These are often called *probably correct* algorithms.

Some researchers, notably Valiant and others in the field of inductive learning [22, 47] have studied 'probably approximately correct' algorithms, combining the above properties in the obvious way. However, as Horvitz [26] and Hansson and Mayer [19] have pointed out, what is needed is a theory of algorithms that maximize the *comprehensive value* of computation. In other words, since the utility of a computation and resulting action is a function of both the quality of the resulting solution and the time taken to choose it, we would like methods for designing algorithms that maximize this combined utility. A suitable generalization of complexity theory has not yet been developed.

A somewhat longer tradition of considering the effects of boundednesss on decision-making exists in economics and the decision sciences, where human characteristics must sometimes be considered. Since the 1960's, I. J. Good [16] has emphasized the conceptual distinction between classical or "type I" rationality, and what he called "type II" rationality, or the maximization of expected utility *taking into account deliberation costs*. Researchers in decision analysis, especially Howard [28], have studied the problem of the *value of information*. Although the theory given below was developed independently, there is a strong parallel with Howard's approach. In the field of economics, Simon [46] made clear the distinction between systems that compute the rational thing to do (procedural rationality), and systems that simply do the rational thing (substantive rationality). He pointed out that procedurally rational systems, whose execution architectures are based on explicit use of declarative knowledge to reach decisions to act, seem to suffer from a good deal of overhead, both in terms of time and extra cognitive machinery.

A notion popularized by Brooks [5] and by Agre and Chapman [1] is that all this deliberation is a waste of time — why don't we just build agents that "do the right thing"? Substantive rationality, however, does not come for free. Although it means that an agent can be perfectly rational despite limited computational resources, it can only arise by prior design or adaptation. In non-trivial environments, particularly those with significant variation, it will be the case that exact solutions to the decision problem are intractable for the designer and unreachable by simple adaptation. With situation-dependent allocation of resources by the agent, on the other hand, it may be possible to approximate rationality to some degree.

We therefore believe that flexible, autonomous systems in complex environments require the ability to reason about the appropriate resources to allocate to computation at any point, and about which computations will be most effective. This premise is shared by several other researchers in AI, and has been the source of a number of projects that have, until recently, developed more or less independently.. Doyle's 'rational psychology' project [10, 11] is based on the idea that computations, or internal state changes, are actions to be chosen like any other actions. He has applied this idea to clarify the notions of belief, intention and learning. Horvitz [24, 25] was also an early AI contributor to the study of rational choice of computation, recognizing its potential to provide a new foundation for the design of intelligent systems. His work on the control of probabilistic inference in medical decision-making parallels our own on the control of search. Heckerman and Jimison [23] have also used medical decision-making as an example domain, showing how to vary the depth of analysis of a therapy problem according to its expected benefits. A third independent project was started in 1986 by Hansson and Mayer [18, 19, 20, 21], who proposed the use of information value as a means of controlling heuristic search, which in turn is implemented as probabilistic inference using information from the heuristic function as evidence. Dean's work on real-time planning [7, 8] assumes a known variation of the intrinsic utility of the results of a computation method with the amount of resources allocated to that method, and hence allocates resources optimally in various resource-bounded scenarios. His survey paper [9] provides a good summary of much of the technical work on decision-theoretic control of inference. Fehling and Breese [13] have applied a decision-theoretic approach to the control of information-gathering actions in a mobile robot domain. Agogino [2] has investigated the use of decision-theoretic modelling of computation in the control of mechanical systems. Bratman, Israel and Pollack [4] sketch a system design combining decision theory with AI planning techniques, using plans to limit the set of actions to be considered. Lesser and his group have studied real-time problem solving in the context of a distributed vehicle-tracking application [35], modulating approximate problem-solving to achieve robust performance.

3 A framework for metareasoning

By *metareasoning* we mean deliberation concerning possible changes to the *computational* state of the agent. Our analysis in this paper is aimed primarily at *episodic* decisions at the base-level; that is, we are most concerned with changes to the internal state that are made in the service of selecting the next action in the real world. Other computational actions, such as inductive learning and compilation, should be covered in any general theory of metareasoning (cf. Doyle's analysis [11]). In principle, their utility can be defined in terms of their effect on the complete sequence of future actions, and by replacing single actions by sequences in our analysis we can evaluate this utility; but to attempt to derive practical consequences from such an analysis appears premature at this stage.[3] In the context of

[3] The formal treatment of computations that change the agent's utility function lies outside the framework of this paper. As in classical decision theory, we assume that the agent's actual utility function is fixed, although estimates of its value may change with further computation. Rational changes to utility functions are not well understood. Ultimately, it seems that an autonomous agent does not in general know its own utility function exactly, since this must be induced from specific experiences, such as pain, pleasure or death, whose interpretation as utility data points is definitional. Revisions to the utility function, then, should be seen primarily as attempts to bring one's predictions of 'happiness' into line with reality.

episodic decision-making, there are two vital, and complementary, roles for metareasoning: first, it allows us as designers to reason about the computational strategies we implement, and to discuss their rationality in a bounded context; second, when implemented explicitly in an agent, it allows that agent to allocate flexibly its limited computational resources depending on the decision situation and to design and compile its own substantively rational decision procedures. In this latter role the justifications for employing explicit metareasoning are the same, in effect, as those for declarative base-level systems, with the additional argument that the knowledge required — knowledge of the agent's own computational structure — can be made introspectively available with little difficulty.

All meta-level systems share a common methodology for implementation: the base-level problem solver operates via the explicit formulation and solution of meta-level problems. For example, the base-level problem-solver in Genesereth's MRS [14, 40] is essentially a theorem-prover, but one that takes the user's goal and, instead of simply running a resolution algorithm, sets up its own goal of finding a way to prove the user's goal. This new goal is a theorem-proving task in its own right, and is solved using meta-level knowledge.

In a *uniform* meta-level architecture, the meta-level problems are formulated using the same language as the base-level problems, and solved using the same mechanism — that is, a meta-meta-level problem is set up to decide how to solve a meta-level goal. The uniformity of language enables the meta-level rules to apply to meta-meta-level goals, and so on. This produces very flexible systems, but introduces the possibility of infinite regress. Regress is particularly problematic when we try to define a constructive notion of the optimal design for a limited rational agent, since the metareasoning done to control problem-solving optimally itself has costs, and therefore needs to be controlled. In other words, no computation can be executed until a computation has been executed to decide on it. Regress has been mentioned by many researchers concerned with bounded rationality [3, 11, 13, 27, 34] but so far only Lipman [36] has claimed any progress on the problem. Clearly we must back off from insisting on optimal control of *all* reasoning, just as we back off from insisting on optimal decisions to act. Some actions, whether computational or external, will have to be taken without being the *immediate* results of deliberation. Decisions at various points in the hierarchy can be hardwired or made by default, approximate decision methods can be used, and so on. However, if an action (including a computational action) is taken by an agent purporting to be an optimal limited rational agent, then unless the agent is extremely fortunate in its selections the action must be the result of prior deliberation or adaptation which has constructed a policy for an intensionally defined class of situations. For the above reasons, among others, two important topics in the RALPH project are inductive learning of meta-level policies and compilation of reasoning. Meta-level learning is discussed briefly in Section 5.4.1, and at greater length in [50, 51]. On compilation of decision-making see [41].

We now turn to the topic of how an agent can select its computations optimally without knowing their outcome — the topic of rational metareasoning.

3.1 Rational metareasoning

The construction of a system capable of rational metareasoning rests on two basic principles:

1. Computations are to be treated as actions, and are therefore to be selected among on the basis of their expected utilities.
2. The utility of a computation is derived from its expected effects, consisting of:
   (a) The passage of time (and associated changes in the external environment).
   (b) The possible revision of the agent's intended actions in the real world.

The ability of a computation to cause the agent to take a different course of action, that has been revealed by the computation to be superior to the agent's original intention, is the fundamental source of positive utility for computations.[4]

It is important to emphasize the obvious fact that the choice of which computation to make, and whether to continue computing, must be made in the absence of exact and immediately available knowledge of the outcome of the computation (else the computation would be pointless). Therefore it will only be possible at best to perform optimally on average, by computing an *expected value* of the computation. That is, computations are treated as if they were stochastic experiments, even when their outcomes are completely deterministic. Note that this already goes against the classical Bayesian assumption that the agent's probabilities and expectations are conditioned upon the sum total of his "knowledge", which is assumed to be deductively closed. For a philosophical discussion of this point which supports our view, see Hacking [17]. Therefore, from a formal standpoint, the results in sections 5 and 6 have as yet only a heuristic justification, borne out by practical results. We expect, however, that the structure of the theory will be retained when it is put on a firmer footing.

We assume in this paper that the outcome of each base-level action is known. The equations to be set forth here can be extended to cover the case of uncertainty about action outcomes, at some cost in terms of the complexity of the exposition. On the other hand, we consider the *utility* of each outcome to be uncertain in most cases. The utilities of a certain perhaps small set of possible states, such as game-ending positions or goal states, may be fully known, but the utilities of other states are defined as an expectation of a distribution over the known states. Again, the classical theory holds that the utility of a game position, for example, is logically determined by the utility of the game-ending positions via the minimax criterion, and hence accessible to the agent's knowledge — although in this case our approach of treating the exact utility of most positions as an inherently unknowable, random

[4] As mentioned above, the value of computations that change the agent's utility function is problematic. Not so problematic is the question of whether a computation is valuable if it increases the agent's confidence that it is taking the right course of action, rather than offering the possibility of revising that course of action. Clearly, if an as yet undone computation can increase the agent's confidence in a course of action, then it can also decrease it (else the optimality of the action must already be guaranteed). Hence there can be no utility to computations whose sole possible outcome is to increase confidence.

variable is more familiar to AI researchers. Just how the standard axioms of probability and utility theory [39, 48] should be revised to allow for the limited rationality of real agents without making them vulnerable to a charge of incoherence is an important open philosophical problem, which we shall not attempt to tackle here. However, we will suggest some of the goals that such a re-axiomatization should accomplish.

3.2 Models of Deliberation

There are many types of computations that may be used in refining a decision to act. In this section, we introduce three progressively more specific models of deliberation.

1. *External model*: At the most general level, we can analyze a system as an external observer, by ascribing utilities and probabilities to the system's actions and internal states. The formal discussion of Section 4 applies to any computation for deciding on an action, as long as there is at any given time a *default action* or "current best action", which we denote by α, that the agent would take if it ceased computing at that time. (It is important to emphasize that this is the action which *appears best* to the agent given its deliberations so far, not necessarily the action which would be truly best for the agent.) The goal of any further computation prior to action is then to refine the choice of default action. Note that *any* algorithm for deciding how to act can be assimilated to this model by assuming that the initial default action consists of a uniform random choice from the available options, which would be taken if the algorithm were interrupted before it produced any useful output.

2. *Estimated utility model*: One way the agent might select its current best action is by making explicit numerical estimates of the utilities of action outcomes. The agent's best action α at any given time is then the action whose current utility *estimate* is a maximum (we assume ties are broken somehow, perhaps randomly). In the AI literature, such utility estimation functions are often called evaluation functions. Deliberation then proceeds by revising and refining the utility estimates. This model will be discussed in Section 5. The principal theoretical problem introduced in this model is the dependence of the utility estimates on the computations that are done. This results in a potential ambiguity in defining a rational course of computation.

3. *Concrete model*: At the concrete level, the decision algorithm is specified as far as the way in which the results of a computation step revise the agent's intended action. In the class of estimated utility systems, this means updating one or more of the action utility estimates. In our applications work [42, 45, 51], we have concentrated on metareasoning in forward search programs — programs that revise the utility estimates for outcome states by generating and evaluating their successors. Analysis at the this concrete level is discussed briefly in Section 6.

3.3 Notation

We will use the following notation throughout the paper:

- A_i: one of a set of possible external actions available to the agent in the current state. (To streamline the notation, reference to a context-dependent "current state" is always implicit.)
- S_j: one of a set of possible computational actions available to the agent.
- W_k: a world state. This includes both external aspects and the internal state of the agent.
- $[X]$: the world state that results from taking action X in the current state, where the action can be internal (a computational action) or external. For simplicity, we consider the case in which each action has a deterministic outcome.
- $[X, W_k]$: the result of taking action X in world state W_k.
- $U(W_k)$: the agent's utility in the state W_k. Typically, U will depend only on the *external* portion of the world state.
- $\mathbf{S}$: a sequence of computational actions; typically, the sequence carried out between the previous external action and the current state. We will use $\mathbf{T}$ to refer to a potential future sequence of computational actions, particularly one ended by an external action.
- $\mathbf{S}.S_j$: the sequence of actions consisting of sequence $\mathbf{S}$ followed by action S_j.
- $\hat{Q}^{\mathbf{S}}$: the agent's *estimate* of a quantity Q, where the estimate results from a computation $\mathbf{S}$.
- α: the agent's current default 'intention'; typically, the external action considered so far to have the highest utility.
- $\alpha_{\mathbf{T}}$: the external action recommended by a computation $\mathbf{T}$.
- β_1, β_2, ...: the external actions currently ranked second-best, third-best etc.

4 The metalevel decision problem

In order for the concept of resource allocation to make sense, the system in question must have a choice of computations available to it, each of which can return a decision or can affect the ultimate decision made. The computations may vary along dimensions such as the amount of time used, the quality of the solution returned, the certainty that an adequate solution will be returned, and the usefulness of a partial computation if the process is interrupted. For our purposes, the amount of time used and the quality of solution returned will be the most

Figure 1: The metalevel decision situation

significant aspects. Figure 1 illustrates the choice situation in which the agent finds itself. At any given time, the agent can either decide to continue computing, by performing one of $S_1 \ldots S_k$, or to stop computing and take the current best external action α.[5] If the agent takes a computation step, then thereafter any sequence of steps may be taken, including the empty sequence, followed by an external action $\alpha_{\mathbf{T}}$, where $\mathbf{T}$ is the complete sequence of computation steps undertaken before acting. For example, a president faced with a difficult and potentially unpopular economic policy choice might request a coarse-grained simulation model to be run; if the results are still equivocal, a more detailed model might be run, but eventually the bullet will have to be bitten and taxes will have to be raised.

According to decision theory, an optimal action is one which maximizes the agent's expected utility, given by

$$E[U([A_i])] = \sum_k P(W_k)U([A_i, W_k]). \tag{1}$$

where $P(W_k)$ is the probability that the agent is currently in state W_k. The value of a computation step S_j is therefore defined in terms of the resulting state $[S_j]$. Computations, however, directly affect the system's internal state, and only indirectly the external world (except by consuming time), whereas a utility function usually refers only to aspects of the total situation that are external to the agent, such as budget deficits. We must therefore define $U([S_j])$ in terms of the changes that take place in the world while the computation occurs, *and the possible change in the agent's future action as a result of the computation.* The next section makes this more precise.

[5] It is important to recall that, at this stage, we are considering the meta-level choice from the viewpoint of an external observer, without suggesting that the agent itself must explicitly set up and solve the decision problem for every computation. In particular, we do not assume that the agent has access to the exact utility function U.

4.1 The value of computation

We define the *net value* of a computational action S_j to be the resulting increase in utility, compared to the utility of the default external action α that would be taken instead:

$$V(S_j) = U([S_j]) - U([\alpha]) \tag{2}$$

A major distinction that needs to be made in specifying $U([S_j])$ is between *partial* and *complete* computations. A partial computation is one that does not result in a commitment to external action; whereas a complete computation does.

If S_j is a complete computation, then the utility of S_j is just the utility of the action α_{S_j} chosen as a result of the computation, given that the action is carried out *after* S_j is completed. That is, $U([S_j]) = U([\alpha_{S_j}, [S_j]])$. Hence,

$$V(S_j) = U([\alpha_{S_j}, [S_j]]) - U([\alpha]) \tag{3}$$

For example, if the aforementioned economic simulation takes a week, then its value will be the difference between doing α (cutting spending, perhaps) now and raising taxes a week later.

In the general (partial) case, the computational action will bring about changes in the internal state of the agent that will affect the value of possible further computational actions. In this case, we want to assess the utility of the internal state in terms of its effect on the agent's ultimate choice of action. Hence the utility of the internal state is the *expected* utility of the base-level action which the agent will ultimately take, given that it is in that internal state. This expectation is defined by summing over all possible ways of completing the deliberation from the given internal state. That is, letting $\mathbf{T}$ range over all possible complete computation sequences following S_j, $\alpha_{\mathbf{T}}$ represent the action chosen by computation sequence $\mathbf{T}$, we have

$$U([S_j]) = \sum_{\mathbf{T}} P(\mathbf{T}) U([\alpha_{\mathbf{T}}, [S_j.\mathbf{T}]]) \tag{4}$$

where $P(\mathbf{T})$ is the probability that the agent will perform the computation sequence $\mathbf{T}$.

If the agent has a perfectly rational metalevel, then the computation sequence selected will be the one maximizing $U([\alpha_{\mathbf{T}}, [\mathbf{T}]])$, and this sequence will have probability 1 in the above equation.[6] In other words, the agent would carry out next the computation step at the beginning of the most valuable computation sequence. This leads to the standard minimax or "expecti-max" approach of decision analysis (see Pearl, [39], ch. 6). However, an agent with only limited rationality might not have any good reason to assume that she will actually succeed in taking the action with highest expected utility. Much less is such an assumption warranted when modeling an agent from the outside. Approaches that avoid this requirement of perfection are discussed in section 5.4.

[6]If several completion sequences have the same maximal utility, the probability of each occurring might be less than 1, but the result of the summation would be the same.

4.2 Ideal and approximate control

According to decision theory, the ideal solution to the meta-level decision problem is simply to perform whichever action from the set $\{\alpha, S_1, \ldots, S_k\}$ has the maximum expected utility. Equivalently, in terms of the net value of computation defined above, the *ideal control algorithm* is as follows:

1. Keep performing the available computation with highest expected net value, until none has positive expected net value.

2. Commit to the action α that is preferred according to the internal state resulting from step 1.

This algorithm, and the intuitive notion of the value of computation which we have defined precisely above, were also proposed by Good [15].

Obviously, the calculation of the expected values of the various possible computations cannot be instantaneous; in fact, as we describe below, it can be arbitrarily hard. It is, however, possible to *approximate* the ideal algorithm by making simplifying assumptions. In particular, we will show how it is possible to use the agent's own utility estimation function to estimate the expected net value of computations.

5 The value of utility estimate revisions

In this section, we will spell out transformations on the equations given in the previous section, which render them more useful to a less-than-omniscient designer or to a limited rational agent that is explicitly reasoning about its own problem-solving. This will correspond to the second level of analysis defined in Section 3.

Our first job is to replace the function U by the function $\hat{U}$, since we are assuming that the former is unknowable, and the agent is able to calculate only in terms of the latter. This will require some care, because the function $\hat{U}$ depends on the stage of the computation. For reasons that we discuss below, we choose to replace U in equation 4 by utility estimates for actions made in the state after the computation in question has been done. Then the estimated net value of computation S_j, from equation 2, is given by

$$\hat{V}(S_j) = \hat{U}^{\mathbf{S}.S_j}([S_j]) - \hat{U}^{\mathbf{S}.S_j}([\alpha]) \tag{5}$$

If we now assume that the agent will take whatever action appears best at the time it decides to act,[7] then equation 4 becomes

$$\hat{U}^{\mathbf{S}.S_j}([S_j]) = \sum_{\mathbf{T}} \hat{P}^{\mathbf{S}.S_j}(\mathbf{T}) \max_i \hat{U}^{\mathbf{S}.S_j.\mathbf{T}}([A_i, [S_j.\mathbf{T}]]) \tag{6}$$

[7]Note how this assumption differs from the more dubious assumption of classical decision theory that the agent will choose the action whose *true* utility is highest.

Of course, before the computation S_j is performed, $\hat{V}(S_j)$ is a random variable. The agent can't know ahead of time what the exact value of $\hat{V}(S_j)$ will be, but given sufficient statistical knowledge of the distribution of $\hat{V}$ for similar actions in past situations, the agent can estimate its *expectation*,

$$E[\hat{V}(S_j)] = E[\hat{U}^{\mathbf{S}.S_j}([S_j]) - \hat{U}^{\mathbf{S}.S_j}([\alpha])] \tag{7}$$

The next three subsections deal with the problem of estimating equation 7 for *complete* computations. Subsection 5.4 discusses the problem of estimating the expected value of partial computations, for which we have not yet found a fully satisfactory solution. The section concludes with a generic description of rational behaviour with respect to decisions between computation and action.

5.1 Analysis for complete computations

Suppose we know or assume that the agent will act after the computation step S_j in question (if it does not act immediately); i.e., suppose it is only choosing among complete computations. Then the utility of S_j will be equal to the utility of the action α_{S_j} believed by the agent to be optimal after S_j has been carried out, given that the action is carried out *after* S_j is completed. That is,

$$\hat{U}^{\mathbf{S}.S_j}([S_j]) = \hat{U}^{\mathbf{S}.S_j}([\alpha_{S_j}, [S_j]]) \tag{8}$$

The net value of S_j is therefore given by

$$\hat{V}(S_j) = \hat{U}^{\mathbf{S}.S_j}([\alpha_{S_j}, [S_j]]) - \hat{U}^{\mathbf{S}.S_j}([\alpha]) \tag{9}$$

and its expectation in the current state is given by

$$E[\hat{V}(S_j)] = E[\hat{U}^{\mathbf{S}.S_j}([\alpha_{S_j}, [S_j]])) - \hat{U}^{\mathbf{S}.S_j}([\alpha])] \tag{10}$$

From this equation, it is clear that the knowledge necessary to assign values to computations resides in the probability distribution for the future utility estimates of the top-level actions. A computation step S_j can in general affect the utility estimates for any of the actions A_i. Thus, let $\mathbf{u} = \langle u_1, \ldots, u_n \rangle$, and let $p_j(\mathbf{u})$ be the joint distribution for the probability that the actions A_1 through A_n get new utility estimates u_1 through u_n respectively. Let $p_{\alpha j}$ be the projection of this distribution for the current best action α — that is, the probability distribution for the random variable $\hat{U}^{\mathbf{S}.S_j}([\alpha])$. Finally, let $\max(\mathbf{u}) = \max\{u_1, \ldots, u_n\}$. Then, by equation 10, we have

$$E[\hat{V}(S_j)] = \int_{\mathbf{u}} \max(\mathbf{u}) p_j(\mathbf{u}) \mathrm{d}\mathbf{u} - \int_{-\infty}^{\infty} u\, p_{\alpha j}(u) \mathrm{d}u \tag{11}$$

The probability distributions can be obtained by gathering statistics on past computations or, in the case of computations yielding exact values, simply by using the agent's current probability distribution for the variable in question. In either case, the computation S_j is characterized as belonging to a given *class* of computations, such as an economic forecast provided by a certain model, or an additional ply of search in an iterative-deepening algorithm. The computation will also be characterized by some pre-determined set of features describing the situation in which it is carried out. Then we can characterize the distribution of the random variable $\hat{V}(S_j)$ by computing *post hoc* the net increase $\hat{U}([\alpha_{S_j}, [S_j]]) - \hat{U}([\alpha])$ for a large sample of computations in similar situations drawn from the same class.[8] If the sampling is done off-line, and the results stored in parameterized form, then the cost of applying formula 9 to estimate the expected net value of computation S_j can be orders of magnitude cheaper than carrying out S_j itself. In that case, the expected value calculation will be well worth doing, since it allows one to select among computations, to prune pointless branches, and to terminate deliberation in such a way as to maximize the overall utility of the agent.

This crude approach will fail when the value of the computation depends on more than a few aspects of the current state; either too much data will be needed to provide accurate statistical estimates, or some aspects will have to be ignored, resulting in large errors. To do better, we need to know the process by which the computation actually revises the utility estimates for external actions. Section 6 describes such improvements.

5.1.1 Discussion

Some deep issues that arise from the fact that the agents we are dealing with have only limited rationality. They manifest themselves particularly in questions about the theoretical status of utility estimates. The majority of work on limited rationality has been done in the context of probability estimates; since utility estimates can be viewed as estimating the probability of obtaining some exact rewards, the issues are the same. Put simply, the problem is that probability estimates, made by a particular computation sequence, do not obey the axioms of probability. For example, one axiom of probability states that the probability of a tautology is 1. Thus if, say, P-K4 is a winning opening move in chess, then the implication relationship between the rules of chess and this fact is tautologous. However, any probability estimate we can arrive at for the win will be less than 1. This is one reason why we adopted the superscript notation for the computation used to arrive at a utility estimate. Fehling and Breese [13] simply write the computation as additional conditioning in the conditional probabilities used to make decisions. While this seems natural, it is perhaps misleading since one is no longer dealing in probabilities.

The lack of a coherent theoretical basis for probability and utility estimates results in some tricky problems in formulating our approach to controlling computation. Note that in the above equations we use the later estimate $\hat{U}^{\mathbf{S}.S_j}$ to evaluate both α_{S_j} and α, the new and the old best moves. The reason for this is most obvious if we consider the case where

[8] This provides the *performance profiles* for the available computations, as used by Dean [7]; his *deliberation scheduling* algorithm follows from equation 10 with each S_j consisting of running one of the available decision procedures for a small increment of time.

$\alpha_{S_j} = \alpha$, but $\hat{U}^{\mathbf{S}.S_j}([\alpha_{S_j}]) > \hat{U}^{\mathbf{S}}([\alpha])$; i.e., S_j simply revises upward the estimated utility of α but does not alter the choice of move. Then the net value of S_j should depend only on the passage of time spent in deliberation, since the real intrinsic utility is of course constant. This would not be the case if we used the later utility estimate for the new best move and the current utility estimate for the current best move.[9]

Examining Howard's Information Value Theory [28] we find a formula which, in the above context, would amount to defining the *expected* net value of S_j as

$$E[\hat{V}(S_j)] = E[\hat{U}^{\mathbf{S}.S_j}([\alpha_{S_j}, [S_j]])] - \hat{U}^{\mathbf{S}}([\alpha]) \tag{12}$$

This will only be equivalent to our formula, given in equation 10, *provided* we have, in the current state $[\mathbf{S}]$, $E[\hat{U}^{\mathbf{S}.S_j}([\alpha])] = \hat{U}^{\mathbf{S}}([\alpha])$. This *coherence condition* will hold true of a perfectly rational agent, since any expected increase or decrease in its expected utility estimate due to further deliberation should already be reflected in the current estimate.

For an agent with only limited rationality, for instance an agent who relies on a fixed, easily computed evaluation function, it may not be safe to assume that the utility estimate is rational in this sense. In fact, it may not be the case that the coherence condition holds even for an *optimal* limited rational agent, since any evaluation function which satisfied the coherence condition might be complicated and expensive to compute. The underlying principle that must, however, be respected is that the real utility of carrying out a given action at a given time will not be changed just by thinking about it. This principle is even more important in *inductive meta-level learning*, wherein the agent evaluates computational actions *post hoc* in order to learn statistical distributions and a predictive function for the value of computation (see section 5.4.1). Here, we do not want to use the formula

$$\hat{V}(S_j) = \hat{U}^{\mathbf{S}.S_j}([\alpha_{S_j}, [S_j]]) - \hat{U}^{\mathbf{S}}([\alpha]) \tag{13}$$

whose expectation is taken in 12, since this will lead to erroneous evaluations. In particular, in the case where $\alpha_{S_j} = \alpha$, we ought to have $V(S_j) \leq 0$, since in this case the only external effect of the computation is to delay action. Equation 9 will be in agreement with this condition, but equation 13 need not be.

This gives a second reason not to employ the Bayesian conditional subjective expectation notation $E[U(A_i)|\mathbf{S}]$ for our concept $\hat{U}^{\mathbf{S}}(A_i)$, since the former denotes the expected utility of A_i, given all knowledge logically deducible by the agent from state $[\mathbf{S}]$, and trivially obeys the identity

$$E[E[U(A_i)|\mathbf{S}.S_j]|\mathbf{S}] = E[U(A_i)|\mathbf{S}] \tag{14}$$

In fact, our use of the expectation symbol $E[\bullet]$ in equation 10 must also be interpreted with some caution. Recall that we wish to view computations as if they were stochastic experiments whose outcomes are drawn from some perhaps unknown probability distribution. Given this assumption, equation 10 refers to the expectation in the objective or frequency sense.

[9] Of course, the desired effect would be obtained if we used the current estimates for both moves. But then every computation would have non-positive utility, since by definition α has a higher current utility estimate than the other moves.

5.2 Time and its Cost

Thus far we have captured the real-time nature of the environment by explicitly including the situation in which an action is taken in the argument to the utility function. Such a comprehensive function of the total state of affairs captures all constraints and trade-offs; in particular, any form of time constraint can be expressed in this way. However, the inclusion of this dependence on the overall state significantly complicates the analysis. Under certain assumptions, it is possible to capture the dependence of utility on time in a separate notion of the *cost of time*, so that the consideration of the quality of an action can be separated from considerations of time pressure.

For many estimated utility functions $\hat{U}$, we can define a related function, the estimated *intrinsic utility*, denoted by $\hat{U}_I$, along with a cost function C, that expresses the difference between total and intrinsic utility:[10]

$$\hat{U}([A_i, [S_j]]) = \hat{U}_I([A_i]) - C(A_i, S_j) \tag{15}$$

where $\hat{U}_I([A_i]) = \hat{U}([A_i])$.[11]

Of course there will always exist a function C which will satisfy equation 15, if only trivially. In order for $\hat{U}_I$ to qualify as an intrinsic utility, it must satisfy a further constraint. Namely, once the state $[S_j]$ has become the new current state, the agent's optimal action *at that time* should always be the one with highest intrinsic utility, independently of the cost C. A sufficient condition for this is that the cost of the computation be independent of the action being evaluated:[12]

$$\hat{U}([A_i, [S_j]]) = \hat{U}_I([A_i]) - C(S_j) \tag{16}$$

Moreover, in the cases we are considering the change in actual utility of an action that occurs during some computation S_j will depend only on $|S_j|$, the length (in elapsed time) of S_j, and on the course of events in the outside world during that time. Since, by definition, computations alter only internal state, S_j will not affect that course of events, so we can let C, and thus $\hat{U}$, depend on the length of the computation rather than the computation itself. Thus the cost function, which we will in this case call TC or "time cost", gives the loss in utility to the agent due to the temporal delay in performing any given action:

$$\hat{U}([A_i, [S_j]]) = \hat{U}_I([A_i]) - TC(|S_j|) \tag{17}$$

[10]Often in AI applications, we begin with an intrinsic utility function (such as a static evaluation function), and C is then defined to yield an accurate estimate of the true utility of an action under various time pressures. Only $\hat{U}$ is in fact independently definable from empirical observations of outcomes.

[11]We define intrinsic utility with reference to the current situation only for convenience; in fact, utility is well-defined only up to an arbitrary positive linear transformation.

[12]This is approximately true in many AI domains such as game-playing or path-planning in a fixed or slowly changing environment. It will not be true in domains such as hunting or war, where different possible actions will gain or lose value at very different rates over time. Even in chess this condition can sometimes fail, since as time is used up more complex positions become less valuable.

If such a function TC exists, then we can separate out the cost and benefit of a computation. We can therefore rewrite equation 9 as follows:

$$\begin{aligned}\hat{V}([S_j]) &= \hat{U}^{\mathbf{S}.S_j}([\alpha_{S_j},[S_j]]) - \hat{U}^{\mathbf{S}.S_j}([\alpha]) \\ &= \hat{U}_I^{\mathbf{S}.S_j}([\alpha_{S_j}]) - \hat{U}_I^{\mathbf{S}.S_j}([\alpha]) - TC(|S_j|) \\ &= \Delta(S_j) - TC(|S_j|) \end{aligned} \tag{18}$$

where

$$\Delta(S_j) = \hat{U}_I^{\mathbf{S}.S_j}([\alpha_{S_j}]) - \hat{U}_I^{\mathbf{S}.S_j}([\alpha]) \tag{19}$$

denotes the *estimated benefit* of the computation. That is, Δ is the estimated increase in intrinsic utility of the new best action over the old best action.

5.3 Simplifying assumptions

Ideally, at any given point in a computation we would like to be able to assess the expected value of all immediate continuations of the computation, without making any assumptions about what we would do afterward. But since computations can in general be arbitrarily long, such a complete analysis is infeasible. Moreover, it is difficult to estimate the value that computations provide by making further computations possible or more valuable. Thus, it is necessary to employ simplifying assumptions or approximations. Here we present two such simplifications. They are closely related to what Pearl [39] has called a "myopic policy". They can be validated only by consideration of the domain of application. Our experiments show that the resulting selection of computations is far better than random; in fact, better than any algorithm designed 'by hand', under certain conditions.

5.3.1 Meta-greedy algorithms

If explicit consideration of all possible complete sequences of computation steps is intractable, then an obvious simplification is to consider *single* primitive steps and to estimate their ultimate effect; we then choose the step appearing to have the highest immediate benefit. We call such algorithms *meta-greedy algorithms*. We get different variants depending on how we estimate the ultimate effect of the computation (see below). A meta-greedy algorithm, then, is one that effectively has a fixed meta-meta-level policy of a depth-limit of 1 on the meta-level decision problem. An analysis based on such an approach will be said to employ the "meta-greedy assumption"; i.e., the assumption that employing such a restricted horizon for the meta-level decision problem will provide an adequate framework for meta-level control. This should be compared with the corresponding assumption made in game-playing research, that limited-depth search is an adequate method for choosing moves.

A weaker form of this approach is to consider, for purposes of the meta-level analysis, some, but not all, finite sequences of computation steps. For instance, in the context of single-agent heuristic search [49], we consider the expansion of all leaf nodes to any given finite depth, up to some fixed depth horizon. This analysis is incomplete because it does not consider the possibility of expanding different nodes to different depths; since that would involve exponentially many possibilities in the number of leaf nodes, it would not lead to a tractable policy. However, by considering a simple subset of the possible computation sequences, up to a given finite size, we can provide a tractable approximation to the complete policy.

5.3.2 Single-step assumption

As we discuss in more detail in the next section, even when we restrict our attention to a limited set of possible computations, it can be very difficult to assess all ways in which a computation can increase utility. In particular, it is difficult to take account of the impact of a computation in making further computations possible or more valuable. On the other hand, it is not difficult to write down simple closed-form expressions for the expected value of *complete* computations, as we have seen. Thus, an obvious simplification is to use these equations to evaluate partial computations. That is, to assume that a computation's value as a complete computation is a useful approximation to its true value as a *possibly* partial computation. This assumption is tantamount to acting as if we had time for at most one more complete computation step; hence we call it the "single-step assumption". The assumption can cause underestimation of the value of some computations; in the case of game-playing, for trees that have grown to a certain size, there will be many node expansions that are valuable as partial computations but have no value as complete computations. This effectively limits the depth to which a meta-greedy algorithm employing the single-step assumption can search a game tree. In the case of single-agent search, however, this phenomenon does not occur, and there is no such restriction on search depth.

It is worth emphasizing here that if either the meta-greedy or single-step assumptions were completely relaxed, the other would become completely true. That is, if it were possible to consider all possible sequences of computations, thus completely relaxing the meta-greedy assumption, then it would be no restriction to consider those sequences as complete computations, assuming the process of deliberation must eventually terminate, so the single-step assumption would be exactly correct. On the other hand, if we could accurately compute the full expected value of single computation steps considered as possibly partial computations which could be followed by further computations, then we would not need to consider all possible computation sequences. Thus, the simplification lies in employing the two assumptions jointly; neither alone would be a restriction.

5.4 Partial computations

When we can assume that the agent will necessarily take the action α_{S_j} after performing computational action S_j, then the methods of sections 5.1-5.2 will suffice. However, in general this is not the case, since, as long as the agent does not arrive at complete certainty about its utility function — which we assume our agents almost never attain — in state $[S_j]$ the agent will still have a choice between taking action α_{S_j}, and continuing to deliberate. (Assume, for simplicity, that there is only one course of computational action open to the agent at each juncture). Thus the value of the computation S_j will be the value of having this choice. In this section, we will discuss possible ways of attempting to relax the single-step assumption and evaluate S_j accurately as a partial computation; our discussion will be preliminary, as the general problem remains unsolved. There are at least two ways to model the utility of being in the state of having a choice between actions, which we discuss below.

To see the *practical* effect of ignoring partial computation values, consider a computation S_1 which increases the utility estimate of the current second-best action β_1 to a point roughly midway between the current value of $\hat{U}_I(\beta_1)$ and that of $\hat{U}_I(\alpha)$. Since it does not change the choice of current best move, S_1 has no positive benefit as a complete computation. However, suppose that S_1 is followed by a further computation S_2 which decreases the utility estimate of α below that of β_1. In this case, the complete *sequence* $S_1.S_2$ has net benefit $\Delta(S_1.S_2) = \hat{U}_I^{S_1.S_2}(\alpha_{S_2}) - \hat{U}_I^{S_1.S_2}(\alpha)$, where $\alpha_{S_2} = \beta_1$. If we evaluate the two computations S_1, S_2 separately using the single-step assumption, we will decide that $\Delta(S_1) = 0$ while $\Delta(S_2) = \hat{U}_I^{S_1.S_2}(\alpha_{S_2}) - \hat{U}_I^{S_1.S_2}(\alpha)$. But intuitively it is clear that some of this benefit should be ascribed to S_1, since it made it possible for S_2 to have its effect. Just as in ordinary decision-making, it is best to consider 'plans', or action sequences, that do have some easily identifiable benefit.

5.4.1 An adaptive approach

If we assume that the agent will be able, when faced with a choice, to choose the best option, then the utility of having a choice between two actions is just the maximum of the utilities of the two actions. This is a standard approach taken in decision analysis; see for instance, Pearl [39]. Let S_k be the action of continuing to compute in state $[S_j]$. Then on this model, the utility of computational action S_j in the current state is given by

$$E[U([S_j])] = E[\max\{U([\alpha_{S_j}, [S_j]]), U([S_k, [S_j]])\}] \tag{20}$$

One difficulty with this formula is that it defines the value of a present computation in terms of the value of a possible future computation. Moreover, it is usually unreasonable for the agent to assume that a *limited* agent without perfect knowledge of its utility function will necessarily choose the action with maximum utility in state $[S_j]$.

On the other hand, if we replace U by $\hat{U}$ in equation 20, we obtain the following:

$$E[\hat{U}([S_j])] = E[\max\{\hat{U}([\alpha_{S_j}, [S_j]]), \hat{U}([S_k, [S_j]])\}] \tag{21}$$

It might be argued that equation 21 is indeed a principle of rationality, in the sense that it imposes a constraint on any *extension* of $\hat{U}$ from base-level actions to computational actions. For it is true almost by definition that the agent will pick the action with maximum $\hat{U}$, and hence in a sense equation 21 says that the agent should value an action the same as it values the computation that recommended the action.[13] As far as possible, then, the function $\hat{U}$ should obey this equation.

[13] Against this it might be argued that if the agent *knows* that its estimated-utility function is error-prone, it needn't think that the right-hand side of equation 21 gives the true expected utility of the computational

However, equation 21 still cannot define a *unique* extension of $\hat{U}$ to computational actions because it is "ungrounded"; i.e. it does not specify how the recursion bottoms out. One possible way of overcoming the non-uniqueness involves specifying a *conservative extension* of $\hat{U}$ from base-level to computational actions. In fact, all we need is a method for arriving at a reasonable *post hoc* evaluation of a given computational action, in order to gain knowledge about the distribution of values by statistical sampling. Note that at the time we are collecting our sample data, we can have available to us the complete outcome of any given decision-making event. Thus, if we are willing to assume, solely for the purpose of evaluating sample computations, that our current agent's decisions between action and computation are correct, then we can arrive at justifiable numerical values by evaluating completed computations using equation 18, and backing up the values obtained to determine the values of partial computations. These values can then be used as data points to induce an intensional characterization of the utility of computations. A more detailed description of this procedure is given in [44, 50].

Data derived in the above way will of course be error-prone, since the whole point of doing the analysis is that we think that the agent often makes incorrect judgments concerning when to stop and when to continue computing. But once we have done our sampling and equipped the agent with decision-theoretic search control knowledge, it will no longer be the same agent, since it will make different, and we hope better, choices in the same sorts of situations. But once an initial set of distributions is obtained, it would be a simple matter for the agent to revise those distributions incrementally as it gains more decision-making experience. In this way, the agent might adaptively converge on a state in which it would possess accurate knowledge of the value of its own computational procedures. If this knowledge can be applied with negligible overhead, then the system will have achieved a state of bounded optimality.

5.4.2 Probabilistic self-modelling

If we do not wish to assume that the agent will always be able to choose the best action in state $[S_j]$, we may assume instead that the agent has a certain probability of taking any given action. This probability may be directly related to the utility of the action; in an extreme case, if we assume that the probability is 1 for the action with highest utility, and 0 for other actions, we arrive at the model of equation 20. The better the agent's utility estimator $\hat{U}$ as an approximation to U, the closer will these probabilities come to this extreme case, but as long as $\hat{U}$ remains error-prone, the probabilities will lie in the open interval $(0, 1)$.

Let $p(\alpha_{S_j}|S_j)$, $p(S_k|S_j)$ be respectively the probabilities that in state $[S_j]$ the agent will immediately take the new best action α_{S_j}, and that it will continue deliberation with computational action S_k. Then on this model the expected utility of the computational

action S_j. We have found it very difficult to settle the sorts of philosophical questions raised by such considerations, and will not attempt to do so here. But we hope the dilemma raised here might convince the reader, as it has us, that the questions raised here are very deep.

action S_j is given by

$$E[U([S_j])] = E[p(\alpha_{S_j}|S_j)U([\alpha_{S_j}, [S_j]]) + p(S_k|S_j)U([S_k, [S_j]])] \tag{22}$$

Like equation 20, this formula gives the value of a current computational action in terms of the value of a future one. If we expand the action S_k recursively in a similar way to the expansion of S_j, and so on, we develop a decision tree as is usual for sequential decisions, except for the fact that all and only the peripheral branches represent external actions (see figure 1). The right-hand side of equation 22 then becomes the expectation of a sum of terms of the form

$$\begin{aligned}&p(\alpha_{S_j}|S_j)U([\alpha_{S_j}, [S_j]]) + p(\alpha_{S_jS_k}|S_jS_k)p(S_k|S_j)U([\alpha_{S_jS_k}, [S_jS_k]]) + \\ &+p(\alpha_{S_jS_kS_l}|S_jS_kS_l)p(S_k|S_j)p(S_l|S_jS_k)U([\alpha_{S_jS_kS_l}, [S_jS_kS_l]]) + \ldots.\end{aligned}$$

This expression thus averages over all possible complete computations (see equation 4).

If we then combine terms corresponding to situations in which the same action is chosen, we get an expression of the form $p(A_1)U(A_1) + \cdots, +p(A_n)U(A_n)$, where now $p(A_i)$ represents the *probability that action A_i is eventually chosen* after completing the computation. Thus, after transforming the equation in this way, we can then drop the expectation sign on the right-hand side to obtain the following constraint on $\hat{U}$:

$$\hat{U}([S_j]) = \sum_i p(A_i)\hat{U}^{\mathbf{S}}([A_i]|A_i \; chosen) \tag{23}$$

where we use the conditional notation to denote informally that the fact that an action will be chosen influences our estimate of its utility, and that the action's utility will depend on the time at which it is taken, and hence on the computation that ends in its being recommended. We believe it should be possible in practice to estimate the various probabilities and conditional expected utilities involved in this equation, although we have not yet attempted an implementation.

5.5 Qualitative behaviour

Before looking at specific implementations, we can describe the *qualitative* behaviour of any algorithm based on our approach. Clearly, an agent will tend to forego further consideration of an action whenever its current estimated value and that of the best candidate are too far apart; in this case, it is unlikely that further computation will provide useful information, since the probability of changing action preference with any reasonable amount of extra computation is negligible. But deliberation may also be pointless if the current estimated utilities of two actions are too close together, and the variance of the difference in values in small; in that case, it may be unlikely that further computation will reveal a *significant* difference between them. In an extreme case, the two actions may actually be symmetric, or nearly so, so that no amount of computation will differentiate significantly between them. This case has received scant attention in the literature, since many algorithm designers

(a) terminate (b) terminate (c) continue

Figure 2: Three basic situations

erroneously assume that the goal of deliberation is to identify the best action, rather than to maximize net expected utility. Lastly, if there is considerable uncertainty as to the values of the available actions, and considerable overlap, further computation is recommended. We illustrate the three major situations graphically in Figure 2.

6 Applications — the concrete level

Up to this point, we have been working at a very general level, making few assumptions about the nature of the base-level decision-making mechanism. Naturally, there are some attributes of certain mechanisms that make them amenable to meta-level control. The overall computation should be modular, in the sense that it can be divided into 'steps' that can be chosen between; the steps must be capable of being carried out in varying orders. Also, systems that use estimated action utilities in their decision-making are more amenable to analysis, as discussed in section 5. In this section, we give a brief description of our methodology and results in analysing and implementing limited rational systems at the concrete level described in section 3.2.

As mentioned in Section 5, decision-theoretic control of reasoning can be carried out using statistical knowledge of the probability distributions for the future utility estimates of external actions, as those estimates are changed by the computation in question. The crude approach, based on data giving these distributions directly, can be refined using knowledge of the base-level decision-making methods of the agent. Essentially, the principle is this:

1. Typically, a computation under consideration is known to affect only *certain components* of the agent's internal structure; for example, running a query about proposition p through a belief network will affect the probability that the agent assigns to p.

2. Changes in those components affect the agent's choice of action in known ways; for example, if the base-level is decision-theoretic, then a change in the probability assigned to some action outcome would affect the expected utility for that action, and hence the choice of action, in the manner prescribed by equation 1.

The empirical component of evaluating computations arises only in the first of these two stages; the more specific we can be about the structure of the base level, the easier it will be to focus on a well-defined, homogeneous population of computation episodes and the more accurate our value estimates will become.

The basic technique to achieve localization of the stochastic effect of the computation is to write the value of the top-level action as a function of the immediate output of the computation. For example, in a search algorithm, the immediate outcome when a node is expanded is the new backed-up value of the node. Recall that $p_j(\mathbf{u})$ is the probability density function for the vector of new values of the top-level actions following computation S_j. This is composed of a probability density function $p_{ij}(u)$ for each top-level action A_i. If the immediate outcome of computation S_j is to change the value of a node j in the tree, then we can model the effect of the computation with a density function $p_{jj}(u)$ for the new value of j.

Now we can define the *propagation function* f that transmits the new value of the node j to produce the new value of each top-level action:

$$\hat{U}_I^{\mathbf{S}.S_j}([A_i]) = f(\hat{U}_I^{\mathbf{S}.S_j}[j])$$

Thus we have a case of one random variable being defined as a function of another. We can therefore use a standard theorem to rewrite the density function p_{ij} in terms of the density function p_{jj}:

$$p_{ij}(x) = p_{jj}(f^{-1}(x)) \left| \frac{d}{dx}(f^{-1}(x)) \right| \tag{24}$$

This transformation can be applied for any base-level decision-making algorithm. Each algorithm will have a characteristic propagation function, that will depend in characteristic ways on the current state of the computation. The main effort involved in applying our theory consists of identifying the function f for the decision algorithm and proving simplifying theorems for the value of computation formula. For an extended example of this process, see [42].

6.1 Subtree Independence

Many base-level decision-making algorithms satisfy the general condition that a given computation affects the utility estimate of only one action. This is the case of *subtree independence*, and it enables us to obtain a further simplification of equation 11. In many domains, such as standard game-playing programs using minimax as a back-up method, the independence assumption will be straightforwardly true, if we consider individual node expansions as single computational steps.[14] Formally, subtree independence holds if and only if for all actions A_i

[14]However, if the search space is treated as a graph rather than a tree, as in some chess and go programs, then the analysis becomes slightly more complicated. In certain problem-solving systems the full analysis must be used. For example, if a computation involves refining a probability estimate used in an influence diagram, the new value may affect the utility of all the top-level actions. In the worst case, the metalevel may have to resort to simulating the computation in question.

Figure 3: Revision of action preference under subtree independence

except at most the one action whose utility estimate is affected,

$$\hat{U}_I^{\mathbf{S}.S_j}([A_i]) = \hat{U}_I^{\mathbf{S}}([A_i]) \tag{25}$$

In this case, examining equation 18, we see that there are essentially two distinct cases in which a computation can have positive benefit, by changing the agent's intention. Either further computation about some currently non-preferred move β_i causes its utility estimate to be raised above that of α, or computation on α causes its utility estimate to be lowered below that of β_1, the current second-best move. Let us call two such computations S_j and S_k respectively (see figure 3).

Suppose we are considering the computation S_j, which affects only the estimated utility of action β_i. The search action will only change our preferred move if $\hat{U}_I^{\mathbf{S}.S_j}([\beta_i]) > \hat{U}_I^{\mathbf{S}.S_j}([\alpha])$, or equivalently, given equation 25, only if $\hat{U}_I^{\mathbf{S}.S_j}([\beta_i]) > \hat{U}_I^{\mathbf{S}}([\alpha])$ (the shaded region to the right of $\hat{U}_I^{\mathbf{S}}([\alpha])$ in figure 3). If this happens, we expect to be better off by an amount $\hat{U}_I^{\mathbf{S}.S_j}([\beta_i]) - \hat{U}_I^{\mathbf{S}}([\alpha])$. If not, there is no gain since our move preference remains unchanged. Thus in this case

$$E[\hat{V}(S_j)] = \int_{\hat{U}_I^{\mathbf{S}}([\alpha])}^{\infty} p_{ij}(x)(x - \hat{U}_I^{\mathbf{S}}([\alpha]))\mathrm{d}x \quad - \quad TC(S_j) \tag{26}$$

Similarly, if we perform a computation S_k that affects only the utility estimate of the current best action α, our action preference is changed only if $\hat{U}_I^{\mathbf{S}.S_k}([\alpha])$, the new expected value of our current preferred move, is less than $\hat{U}_I^{\mathbf{S}}([\beta_1])$. In that case β_1 will become the new best action. (Although the new estimated utility of the new preferred action would be less than the current estimated utility of the current preferred action, the agent would still be *better off than it was*, since the computation will have revealed that α is a blunder.) Hence

$$E[\hat{V}(S_k)] = \int_{-\infty}^{\hat{U}_I^{\mathbf{S}}([\beta_1])} p_{\alpha k}(x)(\hat{U}^{\mathbf{S}}(\beta_1) - x)\mathrm{d}x - TC(S_k) \tag{27}$$

where $p_{\alpha k}(x)$ is the probability density function for $\hat{U}_I^{\mathbf{S}.S_k}([\alpha])$.

All the quantities in equations 26 and 27 are well-defined and are either directly available to the agent or can be estimated statistically. Further simplification and greater accuracy can be obtained by applying the transformation given in equation 24.

6.2 Results for search applications

In a search program, computation typically proceeds by expanding 'frontier nodes' of the partially-grown search tree. The value estimates for actions are calculated by backing up values from the new leaf nodes through their parents. The propagation function f therefore depends on the type of backing up that is used, and will typically involve examining the values of some 'critical' nodes adjacent to the path between the leaf node and the root. The density functions p_{jj} for the immediate effect of the computation depend on the nature of the node being expanded, and the nature of the expansion computation. Statistical information on the probability distribution can therefore be acquired by induction on a large sample of similar states using the same type of expansion computation. Here we only sketch the the nature of the applications and their performance. Details appear in the various papers cited below.

For single-agent search with an admissible heuristic, we have shown [49, 51] that only nodes in the subtree of the current best move should be expanded; derived a computable formula for the expected benefit for expanding a set of such nodes; shown that a best-first search (such as A*) is in fact the best policy given only the heuristic function; and derived an optimal stopping criterion for real-time search. Two common applications of this type of search are the familiar eight-puzzle, and path planning through an environment containing randomly-shaped polygonal obstacles. If we model the cost of computation by assuming a fixed cost-ratio between the cost of generating an edge in the search graph and the cost of the corresponding motion in the external world, we arrive at a figure for the total cost of a solution to such a problem, as a weighted sum of the cost of the solution path taken and the cost of the search employed to find it. We have constructed an algorithm, called DTA* for "decision-theoretic A*", which attempts to minimize the expected total solution cost using the ideas discussed here. We have tested DTA*, for which the cost-ratio is a given parameter, against both A*, which always finds shortest paths regardless of search cost, and RTA* [30], which uses a limited search horizon.[15] For each algorithm, the total solution cost is found for various values of the "rate of exchange" between computation cost and solution path length. Typical results for the eight-puzzle, and for two-dimensional path-planning problems with twenty random polygons, are shown in figure 4. Regardless of the rate of exchange, it seems that DTA* outperforms the other algorithms, and as the cost of computation tends to zero, its behaviour tends to that of A*, as we would expect.

For game-playing, we have derived a formula for the value of expanding a leaf node that can be computed with very little overhead, given the simplifying assumptions outlined above [45]. We implemented a search algorithm, MGSS*, using this formula and played five thirty-two-game tournaments against an alpha-beta algorithm with depth limits from two to six; both algorithms used the same evaluation function. The results, in terms of games won, nodes searched and cpu time used, are given in table 1. Decision-theoretic search control is shown to be up to thirteen times more effective than alpha-beta pruning. The results of the

[15]It should be noted that the depth limit for RTA* is an additional parameter chosen by the user, and that for many problem instances the best performance was obtained with the limit set to 1, i.e., hillclimbing.

Figure 4: 8-puzzle results Path-planning results

algorithm	wins	nodes/game	sec/game
MGSS*	24.5	3,666	40
α-β[2]	7.5	2,501	23
MGSS*	22	6,132	68
α-β[3]	10	9,104	82
MGSS*	20	12,237	170
α-β[4]	12	42,977	403
MGSS*	16.5	21,155	435
α-β[5]	15.5	133,100	1,356
MGSS*	17	45,120	1,590
α-β[6]	15	581,433	6,863

Table 1: Summary of results for Othello

more refined search control method in [42], though preliminary, show a factor of fifty-nine improvement. We do not, however expect this performance trend to persist against deeper-searching alpha-beta programs. As the MGSS* search tree grows larger it is highly likely to reach a situation in which no *single* node expansion can alter the best move choice; at that point, given our meta-greedy and single-step assumptions, the algorithm must conclude that no further search can be worthwhile. We call this phenomenon the "meta-greedy barrier". A promising approach in the short term is to use a two-stage search, consisting of, say, a depth 8 alpha-beta search extended by an MGSS* search, yielding an effective search depth of about 14. We are actively pursuing this line in collaboration with Hans Berliner. The theory has also been applied to probabilistic games, such as backgammon, where deep alpha-beta searches are not feasible in backgammon due to the enormous branching factor. We expect very significant performance improvements as a result.

7 Further work

Future research topics can be divided into two classes: those concerning the construction of a completely general framework for metareasoning, and those concerning the improvement and generalization of the methods that arise from the framework in its current state.

Before we can be said to have a satisfactory formal theory of metareasoning, the scope of the theory must be extended to govern any kind of computational action.[16] Computations such as learning and compilation result in long-term internal state changes rather than contributing to immediate decisions. Hence to include them in the framework we need to consider utility over lifetimes, rather than instantaneous states. Although this, and the resulting need to consider complete action sequences, pose no theoretical difficulty for a framework-builder, it remains to be seen whether useful simplifications can be found that yield practical insights.

The place of metareasoning in a general theory of bounded optimality must also be ascertained. Instantaneous metareasoning yields bounded optimality when it is available, but when it has costs itself, the picture muddies. The problem of infinite regress, mentioned in section 3, illustrates the complexities. One route to clarity may lie through the construction of feedback constraints for a learning procedure that can be shown to converge to bounded optimality even in an architecture with arbitrary levels of metareasoning.

Within the framework already established, we plan to extend the analysis from evaluation search to other forms of decision-making computations, and to construct a general problem-solving architecture that employs 'normative' control over all its activities. The concept of *universal subgoaling* [33, 34] is intended to capture the notion of a complete decision model, by making every aspect of the agent's deliberation recursively open to metareasoning. In

[16] Obtaining a satisfactory definition of computational action, as distinct from action in general, is non-trivial. Doyle [11] simply posits a division of the total world state into internal and external portions. The addition of time into such a framework is reasonably straightforward. Actions such as information-gathering experiments are problematic, since the rules governing their rationality are isomorphic to those for computations rather than ordinary external actions.

the SOAR system [34], however, the basic deliberation mode is goal-directed search. We intend to construct a problem-solving architecture in which decision-theoretic deliberation and its various possible compilations [41] are the basic modes of computation, and in which metareasoning is carried out in the principled fashion outlined above, rather than through hand-generated condition-action rules.

One can also consider the possibility of applying these ideas to control search in theorem provers. However, in order to do this one needs something amounting to a 'current best move'. As currently implemented, theorem provers have no partial information about the success of the branch under consideration (but see [12] for an application of conspiracy numbers to theorem-proving search). The notions of 'guaranteed solution' and 'proof' must be replaced by *tentative/abstract/partial solution* and *justification*. Algorithms using defaults, abstraction hierarchies and island-driving strategies thus seem more amenable to meta-level control and therefore much more robust in the face of complexity.

In addition to trying to relax the meta-greedy and single-step assumptions to obtain still better search control, there are some algorithmic extensions that would yield better performance. We can improve the efficiency of real-time interleaving of computation and action by preserving and extending the partial search tree developed by previous deliberations. Our current implementations successfully alternate deliberation and action, but do not retain any information between deliberations. By keeping and extending that part that part of the search tree that involves the action that is actually chosen, the system will approach standard optimizing algorithms in the limit of unbounded computation. In highly time-bounded situations, the system's behaviour would essentially become that of a reactive system adapting a dynamically-constructed plan to violations of its expectations at the look-ahead horizon.

We can also improve the space-efficiency of selective-search algorithms (and, by extension, any recursively-defined decision-making system). The selective search paradigm has come under criticism because of the need to keep a significant portion of the search tree in memory. A recursive, or problem-reduction algorithm can be implemented that still employs the metareasoning methods described above yet only keeps a small number of nodes in memory. The approach is based on *iterative expansion*, a generalization of iterative deepening. The algorithm is given a current state and a resource limit (say K nodes of search), and calls itself on the state's successors with resource allocations $k_1 \ldots k_n$. Depending on the results of the initial search, the algorithm can increase the resource allocation of some successor by a constant factor α. In this way the algorithm uses only linear space, and in the limit wastes a negligible amount of time in repeated subtree expansions. Furthermore, the use of information returned from each subtree to select the direction for further search gives the scheme a clear advantage over IDA* [31].

The decision-theoretic metalevel can be seen as a means to construct a near-optimal anytime algorithm, in Dean's sense [7], from atomic computation steps. Given several such anytime algorithms, together with their *performance profiles* — mappings from time allocation to output quality — one may wish to construct a more complex real-time system by applying simple composition operators, without having to explicitly specify time allocations among the subsystems. This is a natural generalization of the task of a compiler for a language with procedural abstraction — that is, subroutines. The "contract specification" between caller and callee now has an additional degreee of freedom. Consider, for example, the following definition of a treatment system:

```
(defun treat (x) (repair (diagnose x)))
```

Given the performance profiles of the two subsystems, it is mathematically straightforward to construct the optimal apportionment of resources for a given total allocation, and hence to construct the optimal anytime algorithm for the whole problem. We are currently investigating the extent to which programming constructs other than simple functional composition can be generalized in the same way.

8 Summary

We see computational resource limitations as a major influence on the design of optimal agents. This influence has been neglected in classical theories of normative behaviour, with the result that practical AI systems for non-trivial domains are constructed in an *ad hoc* fashion. A theory of the value of computation will play a central role in the design of optimal limited rational agents.

The basic insight behind normative control reasoning is that computations are actions. Choosing good actions involves reasoning about outcomes and utilities. The utility of a computational action *must be derived from its effect on the agent's ultimate choice of action in the real world*. When the computational action effects changes in internal state that amount to learning or compilation, the relevant effect will be on the long-term sequence of actions. The next problem is to assess this effect without actually performing the computation. Our method can be viewed as an extension and revision of information value theory to cover computations in resource-bounded agents. When the base-level problem-solver operates using value estimates for the real-world actions, the effects of computations can be assessed by using prior statistical knowledge of the distribution of the new value after the computation in question. The required distributions can be induced from direct observation of actual computations. Estimates of the value of computations can then be used to optimize a system's overall behaviour in real-time situations. We can also derive some general qualitative insights into principles of resource allocation, pruning and termination.

We formalized the notion of real-time problem-solving in our framework, and identified time cost as a useful abstraction enabling significant simplifications in the theory. We have applied the theory to analyze both single-agent problem-solving and competitive game-playing, in each case using knowledge of the decision-making algorithm to derive an efficiently-computable formula for the value of computation. The resulting algorithms exhibit significantly better performance than standard methods with negligible metalevel overhead. We indicated some areas for further research in this vein.

Underlying the idea of decision-theoretic control is a distinct approach to dealing with complexity in AI systems. A view prevalent in the inference community, and eloquently described by Kautz [29], has it that intractable problem classes must be avoided, and progress can be made by concentrating on finding polynomial-time subclasses that are as general as possible. Instead, we propose that systems should simply select the computations that will yield the highest return in the shortest time, using as much knowledge of the domain as possible to carry out the selection. The above theory identifies and utilizes one source of appropriate knowledge. Only in domains that have very few identifiable regularities will the formal intractability results apply.

A decision-theoretic, meta-level architecture, particularly one endowed with a varied set of forms of compiled knowledge, generates a rich space of possible agent configurations. Ideally, we would like such a system to converge to a state of bounded optimality. Full development of the theory of metareasoning should provide a set of 'regulative principles' to define and guide this convergence process, replacing the standard axioms of perfect rationality.

References

[1] Agre, P. and Chapman, D. (1987) Pengo: An implementation of a theory of activity. In *Proc. 6th National Conference on Artificial Intelligence*, Seattle, WA: Morgan Kaufmann.

[2] Agogino, A. M. (1989) Real-time reasoning about time constraints and model precision in complex, distributed mechanical systems. In *Proceedings of the AAAI Spring Symposium on AI and Limited Rationality*, Stanford, CA: AAAI.

[3] Batali, J. (1985) A computational theory of rational action (draft). Cambridge: MIT AI Lab.

[4] Bratman, M., Israel, D., and Pollack, M. (in press) Plans and Resource-Bounded Practical Reasoning. *Computational Intelligence*, to appear.

[5] Brooks, R. A. (1986) A robust, layered control system for a mobile robot. *IEEE Journal of Robotics and Automation*, **2**(1), 14-23.

[6] Cherniak, C. (1986) *Minimal Rationality.* Cambridge: MIT Press.

[7] Dean, T. (1987) Intractability and time-dependent planning. In Georgeff, M. P., and Lansky, A. L., (Eds.), *The 1986 Workshop on Reasoning about Actions and Plans.* Los Altos: Morgan Kaufmann, 245-266.

[8] Dean, T., and Boddy, M. (1988) An Analysis of Time-Dependent Planning. In *Proc. 7th National Conference on Artificial Intelligence*, Minneapolis, MN: Morgan Kaufmann, 49-54.

[9] Dean, T. (in press) Decision-theoretic control of inference for time-critical applications. *International Journal of Intelligent Systems*, to appear.

[10] Doyle, J. (1983) What is rational psychology? Toward a modern mental philosophy. *AI Magazine* **4** (3), 50-53.

[11] Doyle, J. (1988) *Artificial Intelligence and Rational Self-Government.* Technical report no. CMU-CS-88-124, Computer Science Department, Carnegie-Mellon University, Pittsburgh, PA.

[12] Elkan, C. (1989) Conspiracy numbers and caching for searching and/or trees and theorem-proving. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, Detroit, MI: Morgan Kaufmann.

[13] Fehling, M. R., and Breese, J. S. (1988) A computational model for decision-theoretic control of problem-solving under uncertainty. In *Proceedings of the Fourth Workshop on Uncertainty in Artificial Intelligence*, Minneapolis, MN: AAAI.

[14] Genesereth, M., and Smith, D. (1981) *Meta-level architecture.* Stanford Heuristic Programming Project, Memo HPP-81-6, Stanford University, Stanford, CA.

[15] Good, I. J. (1968) A five year plan for automatic chess. *Machine Intelligence*, **2**.

[16] Good, I. J. (1971) Twenty-seven principles of rationality. In Godambe, V. P., and Sprott, D. A. (Eds.) *Foundations of Statistical Inference.* Toronto: Holt, Rinehart, Winston, 108-141.

[17] Hacking, I. (1967) Slightly more realistic personal probability. *Philosophy of Science*, **34**, 311-325.

[18] Hansson, O., Holt, G., and Mayer, A. (1986) The comparison and optimization of search algorithm performance. Unpublished manuscript, Columbia University Computer Science Department.

[19] Hansson, O., and Mayer, A. (1988) The Optimality of Satisficing Solutions. *Proceedings of the Fourth Workshop on Uncertainty in Artificial Intelligence*, Minneapolis, MN, 1988.

[20] Hansson, O., and Mayer, A. (in press) Probabilistic Heuristic Estimates. *Annals of Mathematics and Artificial Intelligence*, to appear.

[21] Hansson, O., and Mayer, A. (1989) Heuristic Search as Evidential Reasoning. In *Proceedings of the Fifth Workshop on Uncertainty in Artificial Intelligence*, Windsor, Ontario, August 1989.

[22] Haussler, D. (1988) Quantifying inductive bias: AI learning algorithms and Valiant's learning framework. *Artificial Intelligence*, **36**(2), 177-221.

[23] Heckerman, D., and Jimison, H. (1987) A perspective on confidence and its use in focusing attention during knowledge acquisition. In *Proc. Third Workshop on Uncertainty in AI*, Seattle, WA: AAAI, 123-131.

[24] Horvitz, E. J. (1987) Problem-solving design: Reasoning about computational value, trade-offs, and resources. In *Proc. Second Annual NASA Research Forum*, Moffett Field, CA: NASA Ames, 26-43.

[25] Horvitz, E. J. (1988) Reasoning about beliefs and actions under computational resource constraints. In Uncertainty in Artificial Intelligence Vol. 3., (T. Levitt, J. Lemmer, and L. Kanal, eds.), Amsterdam: North Holland.

[26] Horvitz, E. J. (1988) Reasoning under varying and uncertain resource constraints. In *Proceedings of the Seventh National Conference on Artificial Intelligence*, Minneapolis, MN: Morgan Kaufmann, 139-144.

[27] Horvitz, E. J., and Russell, S. J. (forthcoming) What is to be done? In preparation.

[28] Howard, R. A. (1966) Information value theory. *IEEE Transactions on Systems Science and Cybernetics*, **SSC-2**(1), 22-26.

[29] Kautz, H. (1989) Computers and Thought Lecture, Eleventh International Joint Congress on Artificial Intelligence, Detroit, MI.

[30] Korf, R. E. (1987) Real-time heuristic search: First results. In *Proc. 6th National Conference on Artificial Intelligence*, Seattle, WA: Morgan Kaufmann, 133-138.

[31] Korf, R. E. (1985) Depth-first iterative deepening: An optimal admissible tree search. *Artificial Intelligence*, **27**(1), 97-109.

[32] Laffey, T. J., Cox, P. A., Schmidt, J. L., Kao, S. M., and Read, J. Y. (1988) Real-time knowledge-based systems. *AI Magazine*, **9**(1), 27-45.

[33] Laird, J. E. (1984) *Universal Subgoaling*. Doctoral dissertation, Computer Science Department, Carnegie-Mellon University, Pittsburgh, PA.

[34] Laird, J. E., Newell, A., and Rosenbloom, P. S. (1987) SOAR: An architecture for general intelligence. *Artificial Intelligence* **33**, 1-64.

[35] Lesser, V., Pavlin, J., and Durfee, E. (1988) Approximate processing in real-time problem-solving. *AI Magazine*, **9**(1), 49-61.

[36] Lipman, B. (1989) How to decide how to decide how to ... Limited rationality in decisions and games. In *Proc. AAAI Symposium on AI and Limited Rationality*, Stanford, CA: AAAI.

[37] McAllester, D.A. (1988) Conspiracy Numbers for Min-Max Search. *Artificial Intelligence*, **35**, 287-310.

[38] McCarthy, J. (1958) Programs with common sense. In *Readings in Knowledge Representation* (R. J. Brachman and H. J. Levesque, eds.), Los Altos: Morgan Kaufmann (1985), 300-307.

[39] Pearl, J. (1988) *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference* San Mateo, CA: Morgan Kaufmann.

[40] Russell, S. J. (1985) *The Compleat Guide to MRS* Technical Report no. STAN-CS-85-1080, Computer Science Department, Stanford University, Stanford, CA.

[41] Russell, S. J. (1989) Execution architectures and compilation. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, Detroit, MI: Morgan Kaufmann.

[42] Russell, S. J. (1990) Fine-grained decision-theoretic search control. To appear in *Proceedings of the Sixth Workshop on Uncertainty in Artificial Intelligence*, Cambridge, MA: Morgan Kaufmann.

[43] Russell, S. J., and Wefald, E. H. (1988) Multi-Level Decision-Theoretic Search. *Proceedings of the AAAI Spring Symposium Series on Computer Game-Playing*, Stanford, CA, March, 1988.

[44] Russell, S. J., and Wefald, E. H. (1988) *Decision-theoretic control of search: General theory and an application to game-playing.* Technical report UCB/CSD 88/435, Computer Science Division, University of California, Berkeley, CA.

[45] Russell, S. J., and Wefald, E. H. (1989) On Optimal Game-tree Search using Rational Metareasoning. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, Detroit, MI: Morgan Kaufmann.

[46] Simon, H. A. (1982) *Models of Bounded Rationality, Volume 2.* Cambridge: MIT Press.

[47] Valiant, L. G. (1984) A theory of the learnable. *Comm. A.C.M.* **18** (11), 1134-1142.

[48] von Neumann, J., and Morgenstern, O. (1947) *Theory of Games and Economic Behavior.* Princeton: Princeton University Press.

[49] Wefald, E. H., and Russell, S. J. (1989) Estimating the vaue of computation: The case of real-time search. In *Proceedings of the AAAI Spring Symposium on AI and Limited Rationality*, Stanford, CA, March, 1989.

[50] Wefald, E. H., and Russell, S. J. (1989) Adaptive Learning of Decision-Theoretic Search Control Knowledge. In *Proceedings of the Sixth International Workshop on Machine Learning*, Ithaca, NY: Morgan Kaufmann.

[51] Wefald, E. H., and Russell, S. J. (1989) Decision-Theoretic Control of Heuristic Search. U.C. Berkeley Technical Report, forthcoming.