Rationality and Intelligence: A Brief Update

Stuart Russell

Abstract The long-term goal of AI is the creation and understanding of intelligence. This requires a notion of intelligence that is precise enough to allow the cumulative development of robust systems and general results. The concept of *rational agency* has long been considered a leading candidate to fulfill this role. This paper, which updates a much earlier version (Russell, 1997), reviews the sequence of conceptual shifts leading to a different candidate, *bounded optimality*, that is closer to our informal conception of intelligence and reduces the gap between theory and practice. Some promising recent developments are also described.

1 Artificial Intelligence

AI is a field whose ultimate goal has often been somewhat ill-defined and subject to dispute. Some researchers aim to emulate human cognition, others aim at the creation of intelligence without concern for human characteristics, and still others aim to create useful artifacts without concern for abstract notions of intelligence.

My own motivation for studying AI is to create and understand intelligence as a general property of systems, rather than as a specific attribute of humans. I believe this to be an appropriate goal for the field as a whole, and it certainly includes the creation of useful artifacts—both as a spin-off from and a driving force for technological development. The difficulty with this "creation of intelligence" view, however, is that it presupposes that we have some productive notion of what intelligence is. Cognitive scientists can say "Look, my model correctly predicted this experimental observation of human cognition," and artifact developers can say "Look, my system is worth billions of euros," but few of us are happy with papers saying "Look, my system is intelligent."

A definition of intelligence needs to be *formal*—a property of the system's input, structure, and output—so that it can support analysis and synthesis. The Turing test does not meet this requirement, because it references an informal (and parochial) human standard. A definition also needs to be *general*, rather than a list of special-

Stuart Russell
University of California, Berkeley and Université Pierre et Marie Curie, e-mail: russell@cs.berkeley.edu

ized faculties—planning, learning, game-playing, and so on—with a definition for each. Defining each faculty separately presupposes that the faculty is *necessary* for intelligence; moreover, the definitions are typically not composable into a general definition for intelligence.

The notion of *rationality* as a property of *agents*—entities that perceive and act—is a plausible candidate that may provide a suitable formal definition of intelligence. Section 2 provides background on the concept of agents. The subsequent sections, following the development in Russell (1997), examine a sequence of definitions of rationality from the history of AI and related disciplines, considering each as a predicate P that might be applied to characterize systems that are intelligent:

- P_1: *Perfect rationality*, or the capacity to generate maximally successful behaviour given the available information.
- P_2: *Calculative rationality*, or the in-principle capacity to compute the perfectly rational decision given the initially available information.
- P_3: *Metalevel rationality*, or the capacity to select the optimal combination of computation-sequence-plus-action, under the constraint that the action must be selected by the computation.
- P_4: *Bounded optimality*, or the capacity to generate maximally successful behaviour given the available information and computational resources.

For each P, I shall consider three simple questions. First, are P-systems interesting, in the sense that their behaviour is plausibly describable as intelligent? Second, could P-systems ever exist? Third, to what kind of research and technological development does the study of P-systems lead?

Of the four candidates, P_4, bounded optimality, comes closest to meeting the needs of AI research. It is more suitable than P_1 through P_3 because it is a real problem with real and desirable solutions, and also because it satisfies some essential intuitions about the nature of intelligence. Some important questions about intelligence can only be formulated and answered within the framework of bounded optimality or some relative thereof.

2 Agents

In the early decades of AI's history, researchers tended to define intelligence with respect to specific tasks and the internal processes those tasks were thought to require in humans. Intelligence was believed to involve (among other things) the ability to understand language, the ability to reason logically, and the ability to solve problems and construct plans to satisfy goals. At the core of such capabilities was a store of knowledge. The standard conception of an AI system was as a sort of *consultant*: something that could be fed information and could then answer questions. The output of answers was not thought of as an *action* about which the AI system had a choice, any more than a calculator has a choice about what numbers to display on its screen given the sequence of keys pressed.

The view that AI is about building intelligent *agents*—entities that sense their environment and act upon it—became the mainstream approach of the field only in the 1990s (Russell and Norvig, 1995; Dean et al, 1995), having previously been the province of specialized workshops on "situatedness" and "embeddedness". The "consultant" view is a special case in which answering questions is a form of acting—a change of viewpoint that occurred much earlier in the philosophy of language with the development of speech act theory. Now, instead of simply giving answers, a consulting agent could refuse to do so on the grounds of privacy or promise to do so in return for some consideration. The agent view also naturally encompasses the full variety of tasks and platforms—from robots and factories to game-playing systems and financial trading systems—in a single theoretical framework.

What matters about an agent is what it *does*, not how it does it. An agent can be defined mathematically by an *agent function* that specifies how an agent behaves under all circumstances. More specifically, let $\mathbf{O}$ be the set of percepts that the agent can observe at any instant (with $\mathbf{O}^*$ being the set of observation sequences of any length) and $\mathbf{A}$ be the set of possible actions the agent can carry out in the external world (including the action of doing nothing). The agent function is a mapping $f : \mathbf{O}^* \rightarrow \mathbf{A}$. This definition is depicted in the upper half of Figure 1.

As we will see in Section 3, rationality provides a normative prescription for agent functions and does not specify—although it does constrain—the process by which the actions are selected. Rather than *assume* that a rational agent must, for example, reason logically or calculate expected utilities, the arguments for (Nilsson, 1991) or against (Agre and Chapman, 1987; Brooks, 1989) the inclusion of such cognitive faculties must justify their position on the grounds of efficacy in representing a desirable agent function. A designer of agents has, *a priori*, complete freedom in choosing the specifications, boundaries, and interconnections of subsystems, as long they they compose to form a complete agent. In this way one is more likely to avoid the "hallucination" problem that arises when the fragility of a subsystem is masked by having an intelligent human providing input to it and interpreting its outputs.

Another important benefit of the agent view of AI is that it connects the field directly to others that have traditionally looked on the embedded agent as a natural topic of study, including economics, operations research, control theory, and even evolutionary biology. These connections have facilitated the importation of technical ideas (Nash equilibria, Markov decision processes, and so on) into AI, where they have taken root and flourished.

3 Perfect Rationality

So which agent functions are intelligent? Clearly, doing the right thing is more intelligent that doing the wrong thing. The rightness of actions is captured by the notion of rationality: informally, an action is rational to the extent that is consistent with

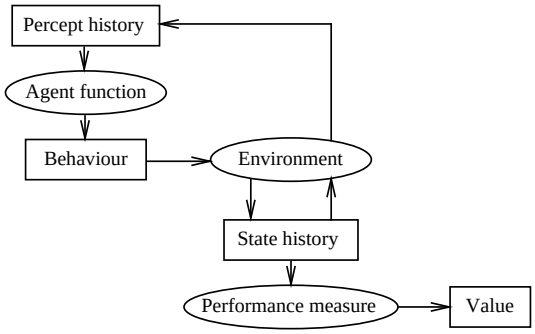

Fig. 1 The agent receives percepts from the environment and generates a behaviour which in turn causes the environment to generate a state history. The performance measure evaluates the state history to arrive at the value of the agent.

the agent's goals (or the task for which it was designed), from the point of view of the information possessed by the agent.

Rationality is, therefore, always understood relative to the agent's ultimate goals. These are expressed mathematically by a performance measure U on sequences of environment states. Let $V(f, \mathbf{E}, U)$ denote the expected value according to U obtained by an agent function f in environment class $\mathbf{E}$, where (for now) we will assume a probability distribution over elements of $\mathbf{E}$. Then a perfectly rational agent is defined by an agent function f_{opt} such that

$$f_{\text{opt}} = \text{argmax}_f V(f, \mathbf{E}, U) \tag{1}$$

This is just a fancy way of saying that the best agent does the best it can. The point is that perfectly rational behaviour is a well-defined function of the *task environment* fixed by $\mathbf{E}$ and U.

Turning to the three questions listed in Section 1: Are perfectly rational agents interesting things to have? Yes, certainly—if you have one handy, you prefer it to any other agent. A perfectly rational agent is, in a sense, perfectly intelligent. Do they exist? Alas no, except for very simple task environments, such as those in which *every* behavior is optimal (Simon, 1958). Physical mechanisms take time to perform computations, while real-world decisions generally correspond to intractable problem classes; imperfection is inevitable.

Despite their lack of existence, perfectly rational agents have, like imaginary numbers, engendered a great deal of interesting research. For example, economists prove nice results about economies populated by them and game-theoretic mechanism designers much prefer to assume perfect rationality on the part of each agent. Far more important for AI, however, was the reduction from a global optimization problem (Equation 1) to a local one: from the perfect rationality of agents to the perfect rationality of individual actions. That is, a perfectly rational agent is one that repeatedly picks an action that maximizes the expected utility of the next state. This reduction involved three separate and largely unconnected results:

the axiomatic utility theory of von Neumann and Morgenstern (1944) (which actually takes for granted the agent's ability to express preferences between distributions over immediate outcomes), Bellman's 1957 theory of sequential decisions, and Koopmans' 1972 analysis of preferences over time in the framework of multiattribute utility theory (Keeney and Raiffa, 1976).

While utility is central to the decision-theoretic notion of perfect rationality, *goals* are usually considered to define the task for a logic-based agent: according to Newell (1982), such an agent is perfectly rational if each action is part of a plan that will achieve one of the agent's goals. There have been attempts to define goals in terms of utilities, beginning with Wellman and Doyle (1991), but difficulties remain because goals are essentially incomplete as task specifications. They do not specify what to do when goal achievement cannot be guaranteed, or when goals conflict, or when several plans are available for achieving a goal, or when the agent has achieved all its goals. It may be better to interpret goals not as primary definitions of the agent's task but as subsidiary devices for focusing computational effort with an overall decision-theoretic context. For example, someone moving to a new city may, after weighing many alternatives and tradeoffs under uncertainty, settle on the goal of buying a particular apartment and thereafter focus their deliberations on finding a plan to achieve that goal, to the exclusion of other possibilities. At the moment we do not have a good understanding of goal formation by a decision-theoretic agent, but it is clear that such behavior cannot be analyzed within the framework of perfect rationality.

As discussed so far, the framework does not say where the beliefs and the performance measure reside—they could be in the head of the designer or of the agent itself. If they are in the designer's head, the designer has to do all the work to build the agent function, anticipating all possible percept sequences. If they are in the agent's head, the designer can delegate the work to the agent; for example, in the setting of reinforcement learning, it is common to equip the agent with a fixed capacity to extract a distinguished reward signal from the environment, leaving the agent to learn the corresponding utility function on states. The designer may also equip the agent with a prior over environments (Carnap, 1950), leaving the agent to perform Bayesian updating as it observes the particular environment it inhabits. Solomonoff (1964) and Kolmogorov (1965) explored the question of universal priors over computable environments; universality, unfortunately, leads to undecidability of the learning problem. Hutter (2005) makes an ambitious attempt to define a universal yet computable version of perfect rationality, but does not pretend to provide the instantaneous decisions required for an actual P_1-system; instead, this work belongs in the realm of P_2-systems, or calculatively rational agents.

Perhaps the biggest open question for the theory of perfect rationality lies in its extension from single-agent to multi-agent environments. Game theorists have proposed many *solution concepts*—essentially, definitions of admissible strategies—but have not identified one that yields a unique recommendation (up to tie-breaking) for what to do (Shoham and Leyton-Brown, 2009).

4 Calculative Rationality

P2-section

The theory of P_1, perfect rationality, says nothing about implementation; P_2, calculative rationality, on the other hand, is concerned with programs for computing the choices that perfect rationality stipulates.

To discuss calculative rationality, then, we need to discuss programs. The agent's decision-making system can be divided into the *machine* M, which is considered fixed, and the *agent program* l, which the designer chooses from the space $\mathscr{L}_M$ of all programs that the machine supports. (M need not be a raw physical computer, of course; it can be a software "virtual machine" at any level of abstraction.) Together, the machine M and the agent program l define an agent function $f = Agent(l,M)$, which, as noted above, is subject to evaluation. Conversely, l is an *implementation* of the agent function f on M; there may, of course, be many such implementations, but also, crucially, there may be none (see Section 6).

It is important to understand the distinction between an agent program and the agent function it implements. An agent program may receive as input the current percept, but also has internal state that reflects, in some form, the previous percepts. It outputs actions when they have been selected. From the outside, the behaviour of the agent consists of the selected actions *interspersed with inaction* (or whatever default actions the machine generates). Depending on how long the action selection takes, many percepts may go by unnoticed by the program.

Calculative rationality is displayed by programs that, *if executed infinitely fast*, would result in perfectly rational behaviour. That is, at time t, assuming it is not already busy computing its choice for some previous time step, the program computes the value $f_{\text{opt}}([o_1,\ldots,o_t])$.

Whereas perfect rationality is highly desirable but does not exist, calculative rationality often exists—its requirements can be fulfilled by real programs for many settings—but it is not necessarily a desirable property. For example, a calculatively rational chess program will choose the "right" move, but may take 10^{50} times too long to do so.

The pursuit of calculative rationality has nonetheless been the main activity of theoretically well-founded research in AI; the field has been filling in a table whose dimensions are the various environment properties (deterministic or stochastic, fully or partially observable, discrete or continuous, dynamic or static, single-agent or multi-agent, known or unknown) for various classes of representational formalisms (atomic, propositional, or relational). In the logical tradition, planning systems and situation-calculus theorem-provers satisfy the conditions of calculative rationality for discrete, fully observable environments; moreover, the power of first-order logic renders the required knowledge practically expressible for a wide range of problems. In the decision-theoretic tradition, there are calculatively rational agents based on algorithms for solving fully or partially observable Markov decision processes, defined initially atomic by atomic formalisms (e.g., transition matrices), later by propositional representations (e.g., dynamic Bayesian networks), and now by first-order probabilistic languages Srivastava et al (2014). For continuous

domains, stochastic optimal control theory (Kumar and Varaiya, 1986) has solved some restricted classes of problems, while many others remain open.

In practice, neither the logical nor the decision-theoretic traditions can avoid the intractability of the decision problems posed by the requirement of calculative rationality. One response, championed by Levesque (1986), is to rule out sources of exponential complexity in the representations and reasoning tasks addressed, so that calculative and perfect rationality coincide—at least, if we ignore the little matter of polynomial-time computation. The accompanying research results on tractable sublanguages are perhaps best seen as indications of where complexity may be an issue rather than as a solution to the problem of complexity, since real-world problems usually require exponentially large representations under the input restrictions stipulated for tractable inference (Doyle and Patil, 1991).

The most common response to complexity has been to use various speedup techniques and approximations in the hope of getting reasonable behaviour. AI has developed a very powerful armoury of methods for reducing the computational cost of decision making, including heuristic evaluation functions, pruning techniques, sampling methods, problem decomposition, hierarchical abstraction, compilation, and the application of metalevel control. Although some of these methods can retain guarantees of optimality and are effective for moderately large problems that are well structured, it is inevitable that intelligent agents will be unable to act rationally in all circumstances. This observation has been a commonplace since the very beginning of AI. Yet systems that select suboptimal actions fall outside calculative rationality *per se*, and we need a better theory to understand them.

5 Metalevel Rationality

Metalevel rationality, also called Type II rationality by I. J. Good (1971), is based on the idea of finding an optimal tradeoff between computational costs and decision quality. Although Good never made his concept of Type II rationality very precise—he defines it as "the maximization of expected utility *taking into account deliberation costs*—it is clear that the aim was to take advantage of some sort of *metalevel architecture* to implement this tradeoff. Metalevel architecture is a design philosophy for intelligent agents that divides the agent program into two (or more) notional parts. The *object level* carries out computations concerned with the application domain—for example, projecting the results of physical actions, computing the utility of certain states, and so on. The *metalevel* is a second decision-making process whose application domain consists of the object-level computations themselves and the computational objects and states that they affect. Metareasoning has a long history in AI, going back at least to the early 1970s (see Russell and Wefald, 1991a, for historical details). One can also view selective search methods and pruning strategies as embodying metalevel expertise concerning the desirability of pursuing particular object-level search operations.

The theory of *rational metareasoning* formalizes Good's intuition that the metalevel can "do the right thinking." The basic idea is that object-level computations are actions with costs (the passage of time) and benefits (improvements in decision quality). A rational metalevel selects computations according to their expected utility. Rational metareasoning has as a precursor the theory of *information value* (Howard, 1966)—the notion that one can calculate the decision-theoretic value of acquiring an additional piece of information by simulating the decision process that would be followed given each possible outcome of the information request, thereby estimating the expected improvement in decision quality averaged over those outcomes. The application to computational processes, by analogy to information-gathering, seems to have originated with Matheson (1968). In AI, Horvitz (1987, 1989), Breese and Fehling (1990), and Russell and Wefald (1989, 1991a,b) all showed how the idea of value of computation could solve the basic problems of real-time decision making.

Perhaps the simplest form of metareasoning occurs when the object level is viewed by the metalevel as a black-box *anytime* (Dean and Boddy, 1988) or *flexible* (Horvitz, 1987) algorithm, i.e., an algorithm whose decision quality depends on the amount of time allocated to computation. This dependency can be represented by a *performance profile* and the metalevel simply finds the optimal tradeoff between decision quality and the cost of time (Simon, 1955). More complex problems arise if one wishes to build complex real-time systems from anytime components. First, one has to ensure the *interruptibility* of the composed system—that is, to ensure that the system as a whole can respond robustly to immediate demands for output. The solution is to interleave the execution of all the components, allocating time to each component so that the total time for each complete iterative improvement cycle of the system doubles at each iteration. In this way, we can construct a complex system that can handle arbitrary and unexpected real-time demands just as if it knew the exact time available in advance, with just a small (≤ 4) constant factor penalty in speed (Russell and Zilberstein, 1991). Second, one has to allocate the available computation optimally among the components to maximize the total output quality. Although this is NP-hard for the general case, it can be solved in time linear in program size when the call graph of the components is tree-structured (Zilberstein and Russell, 1996). Although these results are derived in the simple context of anytime algorithms with well-defined performance profiles, they point to the possibility of more general schemes for allocation of computational resources in complex systems.

The situation gets more interesting when the metalevel can go inside the object level and direct its activities, rather than just switching it on and off. The work done with Eric Wefald looked in particular at search algorithms, in which the object-level computations extend projections of the results of various courses of actions further into the future. For example, in chess programs, each object-level computation expands a leaf node of the game tree and advances the clock; it is an action in the so-called *joint-state Markov decision process*, whose state space is the Cartesian product of the object-level state space (which includes time) and the metalevel state space of computational states—in this case, partially generated game trees. The ac-

tions available are to expand a leaf of the game tree or to terminate search and make a move on the board. It is possible to derive a greedy or *myopic* approximation to the value of each possible computation and thereby to control search effectively. This method was implemented for two-player games, two-player games with chance nodes, and single-agent search. In each case, the same general metareasoning scheme resulted in efficiency improvements of roughly an order of magnitude over traditional, highly-engineered algorithms (Russell and Wefald, 1991a).

An independent thread of research on metalevel control began with work by Kocsis and Szepesvari (2006) on the UCT algorithm, which operates in the context of Monte Carlo tree search (MCTS) algorithms. In MCTS, each computation takes the form of a simulation of a randomized sequence of actions leading from a leaf of the current tree to a terminal state. UCT is a metalevel heuristic for selecting a leaf from which to conduct the next simulation, and has contributed to dramatic improvements in Go-playing algorithms over the last few years. It views the metalevel decision problem as a multi-armed bandit problem (Berry and Fristedt, 1985) and applies an asymptotically near-optimal bandit decision rule recursively to make a choice of which computation to do next. The application of bandit methods to metalevel control seems quite natural, because a bandit problem involves deciding where to do the next "experiment" to find out how good each bandit arm is. Are bandit algorithms such as UCT approximate solutions to some particular case of the metalevel decision problem defined by Russell and Wefald? The answer, perhaps surprisingly, is no. The essential difference is that, in bandit problems, every trial involves executing a real object-level action with real costs, whereas in the metareasoning problem the trials are *simulations* whose cost is usually *independent* of the utility of the action being simulated. Hence UCT applies bandit algorithms to problems that are not bandit problems. A careful analysis (Hay et al, 2012) shows that metalevel problems in their simplest form are isomorphic to *selection problems*, a class of statistical decision problems studied since the 1950s in quality control and other areas. Hay et al develop a rigorous mathematical framework for metalevel problems, showing that, for some cases, hard upper bounds can be established for the number of computations undertaken by an optimal metalevel policy, while, for other cases, the optimal policy may (with vanishingly small probability) continue computing long past the point where the cost of computation exceeds the value of the object-level problem.

Achieving accurate metalevel control remains a difficult open problem in the general case. Myopic strategies—considering just one computation at a time—can fail in cases where multiple computations are required to have any chance of altering the agent's current preferred action. Obviously, the problem of optimal selection of computation *sequences* is at least as intractable as the underlying object-level problem. One possible approach could be to apply metalevel reinforcement learning, especially as the "reward function" for computation—that is, the improvement in decision quality—is easily available to the metalevel *post hoc*. It seems plausible that the human brain has such a capacity, since its hardware is unlikely to have a method of deriving clever new algorithms for new classes of decision problems. Indeed, there is a sense in which *algorithms are not a necessary part of AI systems*.

Instead, one can imagine a general, adaptive process of rationally guided computation interacting with properties of the environment to produce more and more efficient decision making.

Although rational metareasoning seems to be a useful tool in coping with complexity, the concept of metalevel rationality as a formal framework for resource-bounded agents does not seem to hold water. The reason is that, since metareasoning is expensive, it cannot be carried out optimally. Thus, while a metalevel-rational agent would be highly desirable (although not quite as desirable as a perfectly rational agent), it does not usually exist. The history of object-level rationality has repeated itself at the metalevel: perfect rationality at the metalevel is unattainable and calculative rationality at the metalevel is useless. Therefore, a time/optimality tradeoff has to be made for metalevel computations, as for example with the myopic approximation mentioned above. Within the framework of metalevel rationality, however, there is no way to identify the appropriate tradeoff of time for metalevel decision quality. Any attempt to do so via a metametalevel simply results in a conceptual regress. Furthermore, it is entirely possible that in some environments, the most effective agent design will do no metareasoning at all, but will simply respond to circumstances. These considerations suggest that the right approach is to step outside the agent, as it were; to refrain from micromanaging the individual decisions made by the agent. This is the approach taken in bounded optimality.

6 Bounded Optimality

The difficulties with perfect rationality and metalevel rationality arise from the imposition of optimality constraints on *actions* or *computations*, neither of which the agent designer directly controls. The basic problem is that not all agent functions are *feasible* (Russell and Subramanian, 1995) on a given machine M; the feasible functions are those implemented by some program for M. Thus, the optimization over functions in Equation 1 is meaningless. It may be pointed out that not all agent functions are computable, but feasibility is in fact much stricter than computability, because it relates the operation of a program on a formal machine model with finite speed to the actual temporal behaviour generated by the agent.

Given this view, one is led immediately to the idea that optimal feasible behaviour is an interesting notion, and to the idea of finding the program that generates it. P_4, bounded optimality, is exhibited by a program l_{opt} that satisfies

$$l_{\text{opt}} = \text{argmax}_{l \in \mathscr{L}_M} V(\mathit{Agent}(l, M), \mathbf{E}, U) \; . \tag{2}$$

Certainly, one would be happy to have l_{opt}, which is as intelligent as possible given the computational resources and structural constraints of the machine M. Certainly, bounded optimal programs exist, by definition. And the research agenda appears to be very interesting, even though it is difficult.

In AI, the idea of bounded optimality floated around among several discussion groups interested in resource-bounded rationality in the late 1980s, particularly those at Rockwell (organized by Michael Fehling) and Stanford (organized by Michael Bratman). The term itself seems to have been originated by Horvitz (1989), who defined it informally as "the optimization of computational utility given a set of assumptions about expected problems and constraints on resources."

Similar ideas also emerged in game theory, where there has been a shift from consideration of optimal decisions in games to a consideration of optimal decision-making programs. This leads to different results because it limits the ability of each agent to do unlimited simulation of the other, who is also doing unlimited simulation of the first, and so on. Depending on the precise machine limitations chosen, it is possible to prove, for example, that the iterated Prisoner's Dilemma has cooperative equilibria (Megiddo and Wigderson, 1986; Papadimitriou and Yannakakis, 1994; Tennenholtz, 2004), which is not the case for arbitrary strategies.

Philosophy has also seen a gradual evolution in the definition of rationality. There has been a shift from consideration of *act utilitarianism*—the rationality of individual acts—to *rule utilitarianism*, or the rationality of general policies for acting. The requirement that policies be feasible for limited agents was discussed extensively by Cherniak (1986) and Harman (1983). A philosophical proposal generally consistent with the notion of bounded optimality can be found in the "Moral First Aid Manual" (Dennett, 1986). Dennett explicitly discusses the idea of reaching an optimum within the space of feasible decision procedures, using as an example the Ph.D. admissions procedure of a philosophy department. He points out that the bounded optimal admissions procedure may be somewhat messy and may have no obvious hallmark of "optimality"—in fact, the admissions committee may continue to tinker with it since bounded optimal systems may have no way to recognize their own bounded optimality.

My work with Devika Subramanian placed the general idea of bounded optimality in a formal setting and derived the first rigorous results on bounded optimal programs (Russell and Subramanian, 1995). This required setting up completely specified relationships among agents, programs, machines, environments, and time. We found this to be a very valuable exercise in itself. For example, the informal notions of "real-time environments" and "deadlines" ended up with definitions rather different than those we had initially imagined. From this foundation, a very simple machine architecture was investigated in which the program consists of a collection of decision procedures with fixed execution time and decision quality. In a "stochastic deadline" environment, it turns out that the utility attained by running several procedures in sequence until interrupted is often higher than that attainable by any single decision procedure. That is, it is often better first to prepare a "quick and dirty" answer before embarking on more involved calculations in case the latter do not finish in time. In an entirely separate line of inquiry, Livnat and Pippenger (2006) show that, under a bound on the total number of gates in a circuit-based agent, the bounded optimal configuration may, for some task environments, involve two or more separate circuits that compete for control of the effectors and, in essence, pursue separate goals.

The interesting aspect of these results, beyond their value as a demonstration of nontrivial proofs of bounded optimality, is that they exhibit in a simple way what I believe to be a major feature of bounded optimal agents: the fact that the pressure towards optimality within a finite machine results in more complex program structures. Intuitively, efficient decision-making in a complex environment requires a software architecture that offers a wide variety of possible computational options, so that in most situations the agent has at least some computations available that provide a significant increase in decision quality.

One objection to the basic model of bounded optimality outlined above is that solutions are not *robust* with respect to small variations in the environment or the machine. This in turn would lead to difficulties in analyzing complex system designs. Theoretical computer science faced the same problem in describing the running time of algorithms, because counting steps and describing instruction sets exactly gives the same kind of fragile results on optimal algorithms. The $O()$ notation was developed to provide a way to describe complexity that is independent of machine speeds and implementation details and that supports the cumulative development of complexity results. The corresponding notion for agents is asymptotic bounded optimality (ABO) (Russell and Subramanian, 1995). As with classical complexity, we can define both average-case and worst-case ABO, where "case" here means the environment. For example, worst-case ABO is defined as follows:

Worst-case asymptotic bounded optimality
an agent program l is timewise (or spacewise) worst-case ABO in $\mathbf{E}$ *on* M *iff*

$$\exists k, n_0 \; \forall l', n \;\; n > n_0 \Rightarrow V^*(Agent(l, kM), \mathbf{E}, U, n) \geq V^*(Agent(l', M), \mathbf{E}, U, n)$$

where kM denotes a version of M speeded up by a factor k (or with k times more memory) and $V^*(f, \mathbf{E}, U, n)$ is the minimum value of $V(f, E, U)$ for all E in $\mathbf{E}$ *of complexity n.*

In English, this means that the program is basically along the right lines if it just needs a faster (larger) machine to have worst-case behaviour as good as that of any other program in all environments.

Another possible objection to the idea of bounded optimality is that it simply shifts the intractable computational burden of metalevel rationality from the agent's metalevel to the designer's object level. Surely, one might argue, the designer now has to solve offline all the metalevel optimization problems that were intractable when online. This argument is not without merit—indeed, it would be surprising if the agent design problem turns out to be easy. There is however, a significant difference between the two problems, in that the agent designer is presumably creating an agent for an entire class of environments, whereas the putative metalevel agent is working in a specific environment. That this can make the problem *easier* for the designer can be seen by considering the example of sorting algorithms. It may be very difficult indeed to sort a list of a trillion elements, but it is relatively easy to

design an asymptotically optimal algorithm for sorting. In fact, the difficulties of the two tasks are unrelated. The unrelatedness would still hold for BO as well as ABO design, but the ABO definitions make it a good deal clearer.

It can be shown easily that worst-case ABO is a generalization of asymptotically optimal algorithms, simply by constructing a "classical environment" in which classical algorithms operate and in which the utility of the algorithm's behaviour is a decreasing positive function of runtime if the output is correct and zero otherwise. Agents in more general environments may need to trade off output quality for time, generate multiple outputs over time, and so on. As an illustration of how ABO is a useful abstraction, one can show that under certain restrictions one can construct *universal* ABO programs that are ABO for any time variation in the utility function, using the doubling construction from Russell and Zilberstein (1991). Further directions for bounded optimality research are discussed below.

7 What Is To Be Done?

The 1997 version of this paper described two agendas for research: one agenda extending the tradition of calculative rationality and another dealing with metareasoning and bounded optimality.

7.1 Improving the calculative toolbox

The traditional agenda took as its starting point the kind of agent could be built using the components available at that time: a dynamic Bayesian network to model a partially observable, stochastic environment; parametric learning algorithms to improve the model; a particle filtering algorithm to keep track of the environment state; reinforcement learning to improve the decision function given the state estimate. Such an architecture "breaks" in several ways when faced with the complexity of real-world environments (Russell, 1998):

1. Dynamic Bayesian networks are not expressive enough to handle environments with many related objects and uncertainty about the existence and identity of objects; a more expressive language—essentially a unification of probability and first-order logic—is required.
2. A flat space of primitive action choices, especially when coupled with a greedy decision function based on reinforcement learning, cannot handle environments where the relevant time scales are much longer than the duration of a single primitive action. (For example, a human lifetime involves tens of trillions of primitive muscle activation cycles.) The agent architecture must support hierarchical representations of behaviour, including high-level actions over long time scales.

3. Attempting to learn a value function accurate enough to support a greedy one-step decision procedure is unlikely to work; the decision function must support model-based lookahead over a hierarchical action model.

On this traditional agenda, a great deal of progress has occurred. For the first item, there are declarative (Milch et al, 2005) and procedural (Pfeffer, 2001; Goodman et al, 2008) *probabilistic programming languages* that have the required expressive power. For the second item, a theory of hierarchical reinforcement learning has been developed (Sutton et al, 1999; Parr and Russell, 1998). The theory can be applied to agent architectures defined by arbitrary *partial programs*—that is, agent programs in which the choice of action at any point may be left unspecified Andre and Russell (2002); Marthi et al (2005). The hierarchical reinforcement learning process converges in the limit to the optimal completion of the agent program, allowing the effective learning of complex behaviours that cover relatively long time scales. For the third item, lookahead over long time scales, a satisfactory semantics has been defined for high-level actions, at least in the deterministic setting, enabling model-based lookahead at multiple levels of abstraction (Marthi et al, 2008).

These are promising steps, but many problems remain unsolved. From a practical point of view, inference algorithms for expressive probabilistic languages remain far too slow, although this is the subject of intense study at present in many research groups around the world. Furthermore, algorithms capable of learning new model structures in such languages are in their infancy. The same is true for algorithms that construct new hierarchical behaviours from more primitive actions: it seems inevitable that intelligent systems will need high-level actions, but as yet we do not know how to create new ones automatically. Finally, there have been few efforts at integrating these new technologies into a single agent architecture. No doubt such an attempt will reveal new places where our ideas break and need to be replaced with better ones.

7.2 Optimizing computational behaviour

A pessimistic view of Equation 2 is that it requires evaluating every possible program in order to find one that works best—hardly the most promising or original strategy for AI research. But in fact the problem has a good deal of structure and it is possible to prove bounded optimality results for reasonably general classes of machines and task environments.

Modular design using a hierarchy of components is commonly seen as the only way to build reliable complex systems. The components fulfill certain behavioural specifications and interact in well-defined ways. To produce a composite bounded-optimal design, the optimization problem involves allocating execution time to components (Zilberstein and Russell, 1996) or arranging the order of execution of the components (Russell and Subramanian, 1995) to maximize overall performance. As illustrated earlier in the discussion of universal ABO algorithms, the techniques for optimizing temporal behaviour are largely orthogonal to the *content* of the system

components, which can therefore be optimized separately. Consider, for example, a composite system that uses an anytime inference algorithm over a Bayesian network as one of its components. If a learning algorithm improves the accuracy of the Bayesian network, the performance profile of the inference component will improve, which will result in a reallocation of execution time that is guaranteed to improve overall system performance. Thus, techniques such as the doubling construction and the time allocation algorithm of Zilberstein and Russell (1996) can be seen as domain-independent tools for agent design. They enable bounded optimality results that do not depend on the specific temporal aspects of the environment class. As a simple example, we might prove that a certain chess program design is ABO for all time controls ranging from blitz to full tournament play.

The results obtained so far for optimal time allocation have assumed a static, offline optimization process with predictable component performance profiles and fixed connections among components. One can imagine far more subtle designs in which individual components must deal with unexpectedly slow or fast progress in computations and with changing needs for information from other components. This might involve exchanging computational resources among components, establishing new interfaces, and so on. This is more reminiscent of a computational market, as envisaged by Wellman (1994), than of the classical subroutine hierarchies, and would offer a useful additional level of abstraction in system design.

7.3 Learning and bounded optimality

In addition to combinatorial optimization of the structure and temporal behaviour of an agent, we can also use learning methods to improve the design:

- The *content* of an agent's knowledge base can of course be improved by inductive learning. Russell and Subramanian (1995) show that approximately bounded optimal designs can be guaranteed with high probability if each component is learned in such a way that its output quality is close to optimal among all components of a given execution time. Results from statistical learning theory, particularly in the agnostic learning and empirical risk minimization models (Kearns et al, 1992; Vapnik, 2000), can provide learning methods—such as support vector machines—with the required properties. The key additional step is to analyze the way in which slight imperfection in each component carries through to slight imperfection in the whole agent.
- *Reinforcement learning* can be used to learn value information such as utility functions, and several kinds of ε-δ convergence guarantees have been established for such algorithms. Applied in the right way to the metalevel decision problem, a reinforcement learning process can be shown to converge to a bounded-optimal configuration of the overall agent.
- *Compilation* methods such as explanation-based learning can be used to transform an agent's representations to allow faster decision making. Several agent architectures including SOAR (Laird et al, 1986) use compilation to speed up all

forms of problem solving. Some nontrivial results on convergence have been obtained by Tadepalli (1991), based on the observation that after a given amount of experience, novel problems for which no solution has been stored should be encountered only infrequently.

Presumably, an agent architecture can incorporate all these learning mechanisms. One of the issues to be faced by bounded optimality research is how to prove convergence results when several adaptation and optimization mechanisms are operating simultaneously.

7.4 Offline and online mechanisms

One can distinguish between *offline* and *online* mechanisms for constructing bounded-optimal agents. An offline construction mechanism is not itself part of the agent and is not the subject of bounded optimality constraints. Let C be an offline mechanism designed for a class of environments $\mathbf{E}$. Then a typical theorem will say that C operates in a specific environment $E \in \mathbf{E}$ and returns an agent design that is ABO (say) for E—that is, an environment-specific agent.

In the online case, the mechanism C is considered part of the agent. Then a typical theorem will say that the agent is ABO for all $E \in \mathbf{E}$. If the performance measure used is indifferent to the transient cost of the adaptation or optimization mechanism, the two types of theorems are essentially the same. On the other hand, if the cost cannot be ignored—for example, if an agent that learns quickly is to be preferred to an agent that reaches the same level of performance but learns more slowly—then the analysis becomes more difficult. It may become necessary to define asymptotic equivalence for "experience efficiency" in order to obtain robust results, as is done in computational learning theory.

It is worth noting that one can easily prove the value of "lifelong learning" in the ABO framework. An agent that devotes a constant fraction of its computational resources to learning-while-doing cannot do worse, in the ABO sense, than an agent that ceases learning after some point. If some improvement is still possible, the lifelong learning agent will always be preferred.

7.4.1 Fixed and variable computation costs

Another dimension of design space emerges when one considers the computational cost of the "variable part" of the agent design. The design problem is simplified considerably when the cost is fixed. Consider again the task of metalevel reinforcement learning, and to make things concrete let the metalevel decision be made by a Q function mapping from computational state and action to value. Suppose further that the Q function is to be represented by a neural net. If the topology of the neural net is fixed, then all Q functions in the space have the same execution time. Consequently, the optimality criterion used by the standard Q-learning process coincides

with bounded optimality, and the equilibrium reached will be a bounded-optimal configuration.[1] On the other hand, if the topology of the network is subject to alteration as the design space is explored, then the execution time of the different Q-functions varies. In this case, the standard Q-learning process will not necessarily converge to a bounded-optimal configuration; typically, it will tend to build larger and larger (and therefore more and more computationally expensive) networks to obtain a more accurate approximation to the true Q-function. A different adaptation mechanism must be found that takes into account the passage of time and its effect on utility.

Whatever the solution to this problem turns out to be, the important point is that the notion of bounded optimality helps to distinguish adaptation mechanisms that will result in good performance from those that will not. Adaptation mechanisms derived from calculative rationality will fail in the more realistic setting where an agent cannot afford to aim for perfection.

7.5 Looking further ahead

The discussion so far has been limited to fairly sedate forms of agent architecture in which the scope for adaptation is circumscribed to particular functional aspects such as metalevel Q functions. However, an agent must in general deal with an environment that is far more complex than itself and that exhibits variation over time at all levels of granularity. Limits on the size of the agent's memory may imply that almost complete revision of the agent's mental structure is needed to achieve high performance. For example, songbirds grow their brains substantially during the singing season and shrink them again when the season is over.) Such situations may engender a rethinking of some of our notions of agent architecture and optimality, and suggest a view of agent programs as dynamical systems with various amounts of compiled and uncompiled knowledge and internal processes of inductive learning, forgetting, and compilation.

If a true science of intelligent agent design is to emerge, it will have to operate in the framework of bounded optimality. One general approach—discernible in the examples given earlier—is to divide up the space of agent designs into "architectural classes" such that in each class the structural variation is sufficiently limited. Then ABO results can be obtained either by analytical optimization within the class or by showing that an empirical adaptation process results in an approximately ABO design. Once this is done, it should be possible to compare architecture classes directly, perhaps to establish asymptotic dominance of one class over another. For example, it might be the case that the inclusion of an appropriate "macro-operator formation" or "greedy metareasoning" capability in a given architecture will result in an improvement in behaviour in the limit of very complex environments—that is, one cannot compensate for the exclusion of the capability by increasing the machine

[1] A similar observation was made by Horvitz and Breese (1990) for cases where the object level is so restricted that the metalevel decision problem can be solved in constant time.

speed by a constant factor. Moreover, within any particular architectural class it is clear that faster processors and larger memories lead to dominance. A central tool in such work will be the use of "no-cost" results where, for example, the allocation of a constant fraction of computational resources to learning or metareasoning can do no harm to an agent's ABO prospects.

Getting all these architectural devices to work together smoothly is an important unsolved problem in AI and must be addressed before we can make progress on understanding bounded optimality within these more complex architectural classes. If the notion of "architectural device" can be made sufficiently concrete, then AI may eventually develop a *grammar* for agent designs, describing the devices and their interrelations. As the grammar develops, so should the accompanying ABO dominance results.

8 Summary

I have outlined some directions for formally grounded AI research based on bounded optimality as the desired property of AI systems. This perspective on AI seems to be a logical consequence of the inevitable philosophical "move" from optimization over actions or computations to optimization over programs. I have suggested that such an approach should allow synergy between theoretical and practical AI research of a kind not afforded by other formal frameworks. In the same vein, I believe it is a satisfactory formal counterpart of the informal goal of creating intelligence. In particular, it is entirely consistent with our intuitions about the need for complex structure in real intelligent agents, the importance of the resource limitations faced by relatively tiny minds in large worlds, and the operation of evolution as a design optimization process. One can also argue that bounded optimality research is likely to satisfy better the needs of those who wish to emulate human intelligence, because it takes into account the limitations on computational resources that are presumably an important factor in the way human minds are structured and in the behaviour that results.

Bounded optimality and its asymptotic version are, of course, nothing but formally defined properties that one may want systems to satisfy. It is too early to tell whether ABO will do the same kind of work for AI that asymptotic complexity has done for theoretical computer science. Creativity in design is still the prerogative of AI researchers. It may, however be possible to systematize the design process somewhat and to automate the process of adapting a system to its computational resources and the demands of the environment. The concept of bounded optimality provides a way to make sure the adaptation process is "correct."

My hope is that with these kinds of investigations, it will eventually be possible to develop the conceptual and mathematical tools to answer some basic questions about intelligence. For example, *why* do complex intelligent systems (appear to) have declarative knowledge structures over which they reason explicitly? This has been a fundamental assumption that distinguishes AI from other disciplines for

agent design, yet the answer is still unknown. Indeed, Rod Brooks, Hubert Dreyfus, and others flatly deny the assumption. What is clear is that it will need *something like* a theory of bounded optimal agent design to answer this question.

Most of the agent design features that I have discussed here, including the use of declarative knowledge, have been conceived within the standard methodology of "first build calculatively rational agents and then speed them up." Yet one can legitimately doubt that this methodology will enable the AI community to discover all the design features needed for general intelligence. The reason is that no conceivable computer will ever be remotely close to approximating perfect rationality for even moderately complex environments. It may well be the case, therefore, that agents based on approximations to calculatively rational designs are *not even close* to achieving the level of performance that is potentially achievable given the underlying computational resources. For this reason, I believe it is imperative not to dismiss ideas for agent designs that do not seem at first glance to fit into the "classical" calculatively rational framework.

Acknowledgements An earlier version of this paper appeared in the journal *Artificial Intelligence*, published by Elsevier. That paper drew on previous work with Eric Wefald and Devika Subramanian. More recent results were obtained with Nick Hay. Thanks also to Michael Wellman, Michael Fehling, Michael Genesereth, Russ Greiner, Eric Horvitz, Henry Kautz, Daphne Koller, Bart Selman, and Daishi Harada for many stimulating discussions topic of bounded rationality. The research was supported by NSF grants IRI-8903146, IRI-9211512 and IRI-9058427, by a UK SERC Visiting Fellowship. The author is supported by the *Chaire Blaise Pascal*, funded by the l'État et la Région Île de France and administered by the Fondation de l'École Normale Supérieure.

References

Agre PE, Chapman D (1987) Pengi: an implementation of a theory of activity. In: Proceedings of the Tenth International Joint Conference on Artificial Intelligence (IJCAI-87), Morgan Kaufmann, Milan, pp 268–272

Andre D, Russell SJ (2002) State abstraction for programmable reinforcement learning agents. In: Proceedings of the Eighteenth National Conference on Artificial Intelligence (AAAI-02), AAAI Press, Edmonton, Alberta, pp 119–125

Bellman RE (1957) Dynamic Programming. Princeton University Press, Princeton, New Jersey

Berry DA, Fristedt B (1985) Bandit Problems: Sequential Allocation of Experiments. Chapman and Hall, London

Breese JS, Fehling MR (1990) Control of problem-solving: Principles and architecture. In: Shachter RD, Levitt T, Kanal L, Lemmer J (eds) Uncertainty in Artificial Intelligence 4, Elsevier/North-Holland, Amsterdam, London, New York

Brooks RA (1989) Engineering approach to building complete, intelligent beings. Proceedings of the SPIE—the International Society for Optical Engineering 1002:618–625

Carnap R (1950) Logical Foundations of Probability. University of Chicago Press, Chicago

Cherniak C (1986) Minimal Rationality. MIT Press, Cambridge, Massachusetts

Dean T, Boddy M (1988) An analysis of time-dependent planning. In: Proceedings of the Seventh National Conference on Artificial Intelligence (AAAI-88), Morgan Kaufmann, St. Paul, Minnesota, pp 49–54

Dean T, Aloimonos J, Allen JF (1995) Artificial Intelligence: Theory and Practice. Benjamin/Cummings, Redwood City; California

Dennett DC (1986) The moral first aid manual. Tanner lectures on human values, University of Michigan

Doyle J, Patil R (1991) Two theses of knowledge representation: Language restrictions, taxonomic classification, and the utility of representation services. Artificial Intelligence 48(3):261–297

Good IJ (1971) Twenty-seven principles of rationality. In: Godambe VP, Sprott DA (eds) Foundations of Statistical Inference, Holt, Rinehart, Winston, Toronto, pp 108–141

Goodman ND, Mansinghka VK, Roy DM, Bonawitz K, Tenenbaum JB (2008) Church: a language for generative models. In: Proc. UAI-08, pp 220–229

Harman GH (1983) Change in View: Principles of Reasoning. MIT Press, Cambridge, Massachusetts

Hay N, Russell S, Shimony SE, Tolpin D (2012) Selecting computations: Theory and applications. In: Proc. UAI-12

Horvitz EJ (1987) Problem-solving design: Reasoning about computational value, trade-offs, and resources. In: Proceedings of the Second Annual NASA Research Forum, NASA Ames Research Center, Moffett Field, California, pp 26–43

Horvitz EJ (1989) Reasoning about beliefs and actions under computational resource constraints. In: Kanal LN, Levitt TS, Lemmer JF (eds) Uncertainty in Artificial Intelligence 3, Elsevier/North-Holland, Amsterdam, London, New York, pp 301–324

Horvitz EJ, Breese JS (1990) Ideal partition of resources for metareasoning. Technical Report KSL-90-26, Knowledge Systems Laboratory, Stanford University, Stanford, California

Howard RA (1966) Information value theory. IEEE Transactions on Systems Science and Cybernetics SSC-2:22–26

Hutter M (2005) Universal artificial intelligence: sequential decisions based on algorithmic probability. Springer

Kearns M, Schapire RE, Sellie L (1992) Toward efficient agnostic learning. In: Proceedings of the Fifth Annual ACM Workshop on Computational Learning Theory (COLT-92), ACM Press, Pittsburgh, Pennsylvania

Keeney RL, Raiffa H (1976) Decisions with Multiple Objectives: Preferences and Value Tradeoffs. Wiley, New York

Kocsis L, Szepesvari C (2006) Bandit-based Monte-Carlo planning. In: Proc. ECML-06

Kolmogorov AN (1965) Three approaches to the quantitative definition of information. Problems in Information Transmission 1(1):1–7

Koopmans TC (1972) Representation of preference orderings over time. In: McGuire CB, Radner R (eds) Decision and Organization, Elsevier/North-Holland, Amsterdam, London, New York

Kumar PR, Varaiya P (1986) Stochastic systems: Estimation, identification, and adaptive control. Prentice-Hall, Upper Saddle River, New Jersey

Laird JE, Rosenbloom PS, Newell A (1986) Chunking in Soar: The anatomy of a general learning mechanism. Machine Learning 1:11–46

Levesque HJ (1986) Making believers out of computers. Artificial Intelligence 30(1):81–108

Livnat A, Pippenger N (2006) An optimal brain can be composed of conflicting agents. Proceedings of the National Academy of Sciences of the United States of America 103(9):3198–3202

Marthi B, Russell S, Latham D, Guestrin C (2005) Concurrent hierarchical reinforcement learning. In: Proc. IJCAI-05

Marthi B, Russell SJ, Wolfe J (2008) Angelic hierarchical planning: Optimal and online algorithms. In: Proc. ICAPS-08

Matheson JE (1968) The economic value of analysis and computation. IEEE Transactions on Systems Science and Cybernetics SSC-4(3):325–332

Megiddo N, Wigderson A (1986) On play by means of computing machines. In: Halpern JY (ed) Theoretical Aspects of Reasoning about Knowledge: Proceedings of the 1986 Conference (TARK-86), IBM and AAAI, Morgan Kaufmann, Monterey, California, pp 259–274

Milch B, Marthi B, Sontag D, Russell SJ, Ong D, Kolobov A (2005) BLOG: Probabilistic models with unknown objects. In: Proc. IJCAI-05

von Neumann J, Morgenstern O (1944) Theory of Games and Economic Behavior, 1st edn. Princeton University Press, Princeton, New Jersey
Newell A (1982) The knowledge level. Artificial Intelligence 18(1):82–127
Nilsson NJ (1991) Logic and artificial intelligence. Artificial Intelligence 47(1–3):31–56
Papadimitriou CH, Yannakakis M (1994) On complexity as bounded rationality. In: Symposium on Theory of Computation (STOC-94)
Parr R, Russell SJ (1998) Reinforcement learning with hierarchies of machines. In: Jordan MI, Kearns M, Solla SA (eds) Advances in Neural Information Processing Systems 10, MIT Press, Cambridge, Massachusetts
Pfeffer A (2001) IBAL: A probabilistic rational programming language. In: Proc. IJCAI-01, pp 733–740
Russell SJ (1997) Rationality and intelligence. Artificial Intelligence 94:57–77
Russell SJ (1998) Learning agents for uncertain environments (extended abstract). In: Proceedings of the Eleventh Annual ACM Workshop on Computational Learning Theory (COLT-98), ACM Press, Madison, Wisconsin, pp 101–103
Russell SJ, Norvig P (1995) Artificial Intelligence: A Modern Approach. Prentice-Hall, Upper Saddle River, New Jersey
Russell SJ, Subramanian D (1995) Provably bounded-optimal agents. Journal of Artificial Intelligence Research 3:575–609
Russell SJ, Wefald EH (1989) On optimal game-tree search using rational meta-reasoning. In: Proceedings of the Eleventh International Joint Conference on Artificial Intelligence (IJCAI-89), Morgan Kaufmann, Detroit, pp 334–340
Russell SJ, Wefald EH (1991a) Do the Right Thing: Studies in Limited Rationality. MIT Press, Cambridge, Massachusetts
Russell SJ, Wefald EH (1991b) Principles of metareasoning. Artificial Intelligence 49(1–3):361–395
Russell SJ, Zilberstein S (1991) Composing real-time systems. In: Proceedings of the Twelfth International Joint Conference on Artificial Intelligence (IJCAI-91), Morgan Kaufmann, Sydney
Shoham Y, Leyton-Brown K (2009) Multiagent Systems: Algorithmic, Game-Theoretic, and Logical Foundations. Cambridge Univ. Press
Simon HA (1955) A behavioral model of rational choice. Quarterly Journal of Economics 69:99–118
Simon HA (1958) Rational choice and the structure of the environment. In: Models of Bounded Rationality, vol 2, MIT Press, Cambridge, Massachusetts
Solomonoff RJ (1964) A formal theory of inductive inference. Information and Control 7:1–22, 224–254
Srivastava S, Russell S, Ruan P, Cheng X (2014) First-order open-universe pomdps. In: Proc. UAI-14
Sutton R, Precup D, Singh SP (1999) Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. Artificial Intelligence 112:181–211
Tadepalli P (1991) A formalization of explanation-based macro-operator learning. In: Proceedings of the Twelfth International Joint Conference on Artificial Intelligence (IJCAI-91), Morgan Kaufmann, Sydney, pp 616–622
Tennenholtz M (2004) Program equilibrium. Games and Economic Behavior 49(2):363–373
Vapnik V (2000) The Nature of Statistical Learning Theory. Springer Verlag
Wellman MP (1994) A market-oriented programming environment and its application to distributed multicommodity flow problems. Journal of Artificial Intelligence Research 1(1):1–23
Wellman MP, Doyle J (1991) Preferential semantics for goals. In: Proceedings of the Ninth National Conference on Artificial Intelligence (AAAI-91), AAAI Press, Anaheim, California, vol 2, pp 698–703
Zilberstein S, Russell SJ (1996) Optimal composition of real-time systems. Artificial Intelligence 83:181–213