# Seven Days in the Life of a Robotic Agent

Waiyian Chong[1], Mike O'Donovan-Anderson[2], Yoshi Okamoto[3], and Don Perlis[1 2]
{yuan,mikeoda,perlis}@cs.umd.edu,yoshio@microsoft.com

[1] Department of Computer Science, University of Maryland, College Park, MD 20742
[2] Institute for Advanced Computer Studies, University of Maryland
[3] Microsoft Corporation, Redmond, WA 94536

**Abstract.** Bootstrapping is a widely employed technique in the process of building highly complex systems such as microprocessors, language compilers, and computer operating systems. It could play an even more prominent role in the creation of computation systems capable of supporting intelligent agent behaviors because of the even higher level of complexity. The prospect of a self-bootstrapping, self-improving intelligent system has motivated various fields of research in machine learning. However, a robust, generalizable methodology of machine learning is yet to be found; there are still a lot of learning behaviors that no existing learning technique can adequately account for. We believe a uniform, logic-based system such as active logic [1, 2], will be more successful in the realization of this ideal. The overall architecture that we envision is as follows: a central commonsense reasoner module attends to novel situations where the system does not already have expertise, and to its own failures; it then reasons its way to solutions or repairs, and puts these into action while at the same time causing "expert" modules to be either created or retrained so as to more quickly enact those solutions on future occasions. Thus what we propose is a kind of meld between declarative and procedural techniques where the former has great expressive power and flexibility (but is slow) and the latter is very fast but hard to adapt to new situations. We will explore the possibilities of using reflection and continual computation toward this end.

## 1 Introduction

A unifying theme of AI research is the design of an architecture for allowing an intelligent agent to operate in a *common sense informatic situation* [3], where the agent's perception (hence its knowledge about the world) is incomplete, uncertain and subjected to change; and the effect of its actions indeterministic and unreliable. There are many reasons (e.g., scientific, philosophical, practical) to study intelligent agent architecture; for our purposes, we will define our goal as to improve the performance of the agent, where performance is in turn defined as resources (time, energy, etc) spent in completing given tasks. We are interested in the question "What is the best strategy to build an agent which can perform competently in a common sense informatic situation?" It is clear that it will be impractical for the designer of the agent to anticipate everything it may encounter in such a situation; hence it is essential that some routes of self-improvement be provided for the agent if it is to attain reasonable level of autonomy. What should we provide to the architecture to open these routes?

For an agent to function competently in commonsense world, we can expect the underlying architecture to be highly complex. Careful attention should be paid to the designing process, as well as the designed artifact to ensure success. We identified the following requirements to guide our design: (i) In addition to fine-tuning of specialized modules the agent might have, more fundamental aspects of the architecture should be open to self-improvement. For example, an agent designed to interact with people may have a face recognition module; a learning algorithm to improve its face recognition accuracy is of course desirable, but it is not likely to be helpful for the agent to cope with unexpected changes in the world. (ii) Improvements need to be made reasonably efficiently. It's said that a roomful of monkeys typing away diligently at their keyboards will eventually produce the complete works of Shakespeare; in the same vein, we can imagine a genetic algorithm, given enough time and input, can evolve a sentient being, but the time it takes will likely be too long for us to withstand. (iii) Somewhat related to the previous two points, it is important to stress that the improvements made be transparent to us so that we can incrementally provide more detailed knowledge and guidance when necessary to speed up the improvements.

Toward building an intelligent agent, we can borrow a few lessons from builders of other sophisticated systems: in particular, the technique of bootstrapping is of relevance. Bootstrapping technique has been widely employed in the process of building highly complex systems such as microprocessors, language compilers, and computer operating systems. It could play an even more prominent role in the creation of computation systems capable of supporting intelligent agent behaviors, because of the even higher level of complexity. Typically in a bootstrapping process, a lower-level infrastructural system is first built "by hand"; the complete system is then built, within the system itself, utilizing the more powerful constructs provided by the infrastructure. Hence, it provides benefits in the ways of saving effort as well as managing complexity.

Ideally, as designers of the agent, we'd like to push as much work as possible to be automated and carried out by computer. There is no doubt that the study of specialized algorithms has been making great contributions to the realization of intelligent agency; however, we think that the study of bootstrapping behavior may be a more economical way to achieve that goal. Instead of designing the specialized modules ourselves, we should instead look for way to provide the infrastructure on which agents can discover and devise the modules themselves.

Once we accept bootstrapping as a reasonable way to proceed, a few natural questions arise: What constructs are needed in the infrastructure to support the bootstrapping of intelligence? How should they be combined? How should they operate? More generally, if we leave alone a robot agent in a reasonably rich environment for a long period of time, what will enable the robot to evolve itself into a more competitive agent? How do we provide a path for the agent to improve itself? We think an example will help us to answer the questions! In the next section, we will tell the story of Al the office robot, to show the importance and desirability of self-improving capability in an artificial agent. In light of the typical problems that a robot may encounter in the real world, the following two sections (Sec 3 and Sec 4) present a more detailed account of two key ideas: reflection and continual computation, which we think are essential to the success of the robot, and argue that the uniformity and expressiveness of a logic-based system can facilitate the implementation of complex agency. In section 5, we will give a brief introduction to Active Logic, the theoretical base of our implementation, and discuss why we think this approach is promising. It is nonetheless clear that there are still very many problems to solve before Al can be more than science fiction.

## 2 Seven Days in the Life of Al

Let us consider Al, a robot powered by active logic. Al is an "office robot", who roams the CS office building delivering documents, coffee, etc. Al was endowed at birth with the desire to make people happy. We will see how Al developed into an excellent office robot of great efficiency through its first week of work.
**1st day**: Al was first given a tour of the building. Among other things, it was shown the power outlets scattered around the building so that it could recharge itself.
**2nd day**: The morning went well: Al delivered everything on target. But during the afternoon Al ran into a problem: it found itself unable to move! The problem was soon diagnosed — it was simply a low battery. (Since thinking draws less energy than moving, Al could still think.) It turned out that although Al knew it needed power to operate and it could recharge itself to restore its battery, it had never occurred to Al that, "it would need to reach an outlet before the power went too low for it to move!" [1] The movement failure triggered Al to derive the above conclusion, but it was too late; Al was stuck, and could not deliver coffee on request. Caffeine deprived computer scientists are not happy human beings; Al had a bad day.
**3rd day**: Al was bailed out of the predicament by its supervisor in the morning. Having learned its lesson, Al decided to find an outlet a few minutes before the battery got too low. Unfortunately for Al, optimal route planning for robot navigation is an NP-complete problem. When Al finally found an optimal path to the nearest power outlet, its battery level was well below what it needed to move, and Al was stuck again. Since there was nothing else it could do, Al decided to surf the web (through the wireless network!), and came upon an interesting article titled "Deadline-Coupled Real-time Planning" [4].
**4th day**: After reading the paper, Al understood that planning takes time, and that it couldn't afford to find an optimal plan when its action is time critical. Al decided to quickly pick the outlet in sight when its battery was low. Unfortunately, the outlet happened to be too far away, and Al ran out of power again before reaching it. In fact, there was a closer outlet just around the corner; but since a non-optimal algorithm was used, Al missed it. Again, stuck with nothing else to do, Al kicked into the "meditation" mode where it called the Automated Discovery (AD) module to draw new conclusions based on the facts it accumulated these few days. Al made some interesting discoveries: upon inspecting the history of its observations and reasonings, Al found that there were only a few places it frequented; it could actually precompute the optimal routes from those places to the nearest outlets. Al spent all night computing those routes.

Meanwhile, Al also built a special-purpose (procedural) navigator module NM to navigate to those routes, so that (i) the navigation would be faster and (ii) it could spend

[1] Counter to traditional supposition that all derivable formulas are already present in the system.

more time attending to other matters such as sorting mail for delivery (this being a more error-prone task requiring commonsense analysis).

**5th day**: This morning, Al's AD module derived an interesting theorem: "if the battery power level is above 97% of capacity when Al starts (and nothing bad happened along the way), it can reach an outlet before the power is exhausted." Al didn't get stuck that day. But people found Al to be not very responsive. Later, it was found that Al spent most of its time around the outlets recharging itself — since Al's power level dropped 3% for every 10 minutes, the theorem above led it to conclude that it needed to go to the outlet every 10 minutes.

It also turned out that two of the power outlets it used for recharging became inoperative and AL had to deliberately (reason its way to) override its navigator module NM, and retrain it to avoid those outlets.

**6th day**: After Al's routine introspection before work, it was revealed that the knowledge base was populated with millions of theorems similar to the one it found the day before, but with the power level at 11%, 12%, ..., and so on. In fact, the theorem is true when the power level is above 10% of capacity. Luckily, there was a meta-rule in Al's knowledge base saying that "a theorem subsumed by another is less interesting;" thus all the theorems with parameter above 10% were discarded. Equipped with this newer, more accurate information, Al concluded that it could get away with recharging itself every 5 hours.

**7th day**: That happened to be Sunday. Nobody was coming to the office. Al spent its day contemplating the meaning of life.

Analyzing the behavior of Al, we can see a few mechanisms at play: in addition to the basic deductive reasoning, goal directed behavior, etc., Al also demonstrates capabilities such as abductive reasoning (diagnoses of failures), explanation-based learning (compilation of navigation rules, derivation of recharging rules), reflection (examining and reasoning about its power reading, revision of recharging rule), and time-sensitivity (understanding that deliberations take time, people don't like waiting, etc). Of course, none of these is new in itself; however, the interactions among them has enabled Al to demonstrate remarkable flexibility and adaptivity in a ill-anticipated (by the designer of Al) and changing world. Below, we will elaborate on the reflective capability and the continual aspect of the agent's operations.

## 3 Reflection

A computational system is said to be reflective when it is itself part of its own domain (and in a causally connected way). More precisely, this implies that (i) the system has an internal representation of itself, and (ii) the system can engage in both "normal" computation about the external domain and "reflective" computation about itself [5]. Hence, reflection can provide a principled mechanism for the system to modify itself in a profound way.

We suggest that a useful strategy for a self-improving system is to use reflection in the service of self-training. Just as a human agent might deliberately practice a useful task, increasing her efficiency until (as we say) it can be done "unconsciously" or "automatically", without explicit reasoning, we think that once a reflective system identifies an algorithm or other method for solving a frequently encountered problem, it should be able to create procedural modules to implement the chosen strategy, so as to be able in the future to accomplish its task(s) more efficiently, without fully engaging its (slow and expensive) common-sense reasoning abilities.

Although reflection sounds attractive, it has largely been ignored by researchers of agent architecture, mainly because of the high computation complexity involved in doing reflective reasoning. However, we think the solution to the problem is not by avoiding reflection, but looking at the larger picture and considering the environment and extent in which an agent operates, and finding way to reap the benefits of reflection without being bogged down by its cost. We think the notion of continual computation is a promising venue for reflection to become useful.

## 4 Continual Computation

Any newcomer to the field of AI will soon find out that, almost without exception, all "interesting" problems are NP-hard. When a computer scientist is confronted with a hard problem, there are several options to deal with it. For example, one can simplify the problem by assuming it occurs only under certain conditions (which are not always realistic) and hoping bad cases don't happen frequently. One can also identify a simpler subproblem so that it can be solved algorithmically and automated, and leave the hard part for the human. Another option is for the scientist to study the problem carefully, derive some heuristics, and hope that they will be adequate most of the time. But none of these is quite satisfying: ideally, we would like the computer to do as much work for us as possible, and hopefully, be able to derive the heuristics by itself. A promising approach toward realizing this ideal is the notion of *continual computation* [6].

The main motivation behind continual computation is to exploit the *idle time* of a computation system. As exemplified by usage patterns of desktop computers, workstations, web-servers, etc. of today, most computer systems are under utilized: in typical employments of these systems, relatively long spans of inactivity are interrupted with bursts of computation intensive tasks, where the systems are taxed to their limits. How can we make use of the idle time to help improve performance during critical time?

Continual computation generalizes the definition of a *problem* to encompass the uncertain stream of challenges faced over time. One way to analyze this problem is to put it into the framework of probability and utility, or more generally, rational decision making:

> Policies for guiding the precomputation and caching of complete or partial solutions of potential future problems are targeted at enhancing the expected value of future behavior. The policies can be harnessed to allocate periods of time traditionally viewed as idle time between problems, as well as to consider the value of redirecting resources that might typically be allocated to solving a definite, current problem to the precomputation of responses to potential future challenges under uncertainty[7].

An implicit assumption of the utility-based work in continual computation is that the future is somehow predictable. But in many cases, this cannot be expected. For example, for long term planning, most statistics will probably lose their significance. Here is a place where logic-based systems with the capability to derive or discover theorems on its own (e.g., Lenat's AM system) can play a complementary role, similar to the way that mathematics plays a complementary role to engineering. Just as mathematicians usually do not rely on immediate reward to guide their research (yet discover theorems of utmost utility), AM can function in a way independent of the immediate utility of its work.

More precisely, if we adopt logic as our base for computation and look at problem solving as theorem proving [8], a system capable of discovering new theorems can become a very attractive model of a continual computation system. In such a system, every newly discovered theorem has the potential of simplifying the proof of future theorem; so in essence, theorems become our universal format for caching the results of precomputation and partial solutions to problems.

A simplistic embodiment of the model can just be a forward chaining system capable of combining facts in its database to produce new theorems using modus ponens, for instance. Such a system is not likely to be very useful, however, because it will spend most of its time deriving uninteresting theorems. So the success of this model of continual computation will hinge on whether we can find meaningful criteria for the "interestingness" of a theorem. In the classical AM [9–11], the system relies largely on human judgment determine interestingness. In a survey of several automated discovery programs, Colton and Bundy [12] identify several properties of concepts which seem to be relevant to their interestingness, such as novelty, surprisingness, understandability, existence of models and possibly true conjectures about them. Although these properties seem plausible, it is not obvious they are precise enough to be operational to guide automated discovery programs toward significant results.

## 5 Toward Implementation

Considering for a moment a few general requirements for a real-world agent like Al, it is obvious, first of all, that Al must perceive its environment. Further, if it is to reason about what he perceives, and use this reasoning to guide its actions, it must be capable of coming to have perceptually grounded empirical beliefs.[2] But the world is always changing, and if Al's beliefs are going to be useful, they would best reflect the world as it actually is — and that means that Al must be capable of forming new beliefs, and revising or getting rid of old beliefs, in response to real-time perceptual input from the world. Al's belief system, that is to say, must be appropriately reactive.

But it is clear that reactivity is not enough. For in any reasoning system, where new beliefs are constantly being derived from old ones, reactivity introduces a difficult problem: if a belief which is an antecedent condition in a sequence of reasoning itself changes, then the consequents of that sequence may be invalidated. Al must be able to recognize when this happens, and take appropriate steps to solve the problem. Thus,

[2] For a simple example of why perceptions are not enough, consider that when Al needs to charge itself, it may not be able, at that moment, to perceive a charging station. In this case it will need to *remember* where (the nearest) one is; thus even this basic action requires both perception (that his battery is low) and belief (that there is a station around the corner).

Al needs to be a perceiver, a believer, a reasoner, and a meta-reasoner, capable of reasoning not just *with* its beliefs, but *about* them. Further (and crucial to the possibility of boot-strapping), Al is concerned not just with meeting its various first-order goals (delivering coffee, recharging) in light of the changing environment in which it acts, but also in fulfilling second-order goals [13] — goals about what sort of agent to become — such as increased efficiency. Meta-reasoning in light of these second-order desires is a somewhat complicated and delicate task, for it can involve not just deciding how to do something, but deciding how to decide to do it, as when Al builds a special navigator module to compute paths to the various charging stations. Al, this is to say, must be both reactive and reflective, attending not just to the external world, but also to its own mind, finding ways to bring about its goals and desires in both spheres.

Al, therefore, must be able to recognize and react to changes in the environment, and to deal with the contradictions, irregularities, and invalidated conclusions that this will involve; further, in choosing when and how to act, it must be able to consider not just how best to achieve a given goal in light of the state of the world, but also which goals are the best to achieve, and which methods of achieving them are best given the sort of agent it wants to become.[3] Together, this suggests the need for extremely rich representations of Al's beliefs, goals, and desires, sufficient to support robust meta-reasoning. What is needed, then, for the reasoning engine of a real-world agent, is something reactive, flexible and expressive, a reasoning system in which beliefs can be added, changed, and removed in real-time without disrupting the reasoning process, and which can support introspection and meta-reasoning. Active Logic was designed, and is being continually developed, with these desiderata in mind.

As is detailed in, e.g., [18, 1, 19] active logic is one of a family of inference engines (step-logics) that explicitly reason in time, and incorporate a history of their reasoning as they run. Motivated in part by the thought that human reasoning takes place step-wise, in time — and that this feature supports human mental flexibility — Active Logic works by combining inference rules with a constantly evolving measure of time (a "Now") that can itself be referenced in those rules. As an example, from $Now(t)$ — the time is now "$t$" — one infers $Now(t+1)$, for the fact of an inference implies that time (at least one 'time-step') has passed. All the inference rules in Active Logic work temporally in this way: at each time-step all possible one step inferences are made, and only propositions derived at time $t$ are available for inferences at time $t+1$. There are special persistence rules so that every theorem $\alpha$ present at time $t$ implies itself at time $t+1$; likewise there are special rules so that if the knowledge base contains both a theorem $\alpha$ and its negation $\neg\alpha$, these theorems and their consequences are "distrusted"

[3] Interestingly, Aristotle's notion of character — roughly, a stable disposition for choosing which goals to achieve and which methods to employ in so doing — is more useful in understanding an agent like Al than are more modern notions of choice and agency, in which a radically free act of the will plays a larger role. For it is clear that the more decisions Al makes about what sort of agent to become — the more specialized modules and simple, reactive, procedural systems it builds for dealing with everyday, common situations — the more ossified its reactions to the world will become, and the more difficult it will be to change. Balancing the need for spontaneity against that for stability and efficiency is at the core of agency, both human and artificial. For more on the tensions of agency see, e.g. [14, 15]; for more on Aristotelian notions of character [16, 17].

so they are neither carried forward themselves nor used in further inference. These features, along with a quotation mechanism allowing theorems to refer to each other, give active logic the expressive and inferential power to monitor its own reasoning in a real-time fashion, as that very reasoning is going on, allowing it to watch for errors (such as mismatches between the environment and expectations), to note temporal aspects of actions or reasoning (an approaching deadline, or that progress is or is not occurring) which might dictate the adoption of a different goal or strategy, and to exert reasoned control over its past and upcoming inferential processes, including re-examination and alteration of beliefs and inferences. Active logic therefore supports flexible reasoning, and is well suited for complex and ever-changing real-world contexts like autonomous agency and human-computer dialog.

A simple example of active logic inference is shown, below.

$$\begin{aligned} i :& \mathit{Now}(i), A, A \rightarrow B \\ i+1 :& \mathit{Now}(i+1), A, A \rightarrow B, B \end{aligned}$$

Here $i$ and $i + 1$ in the left margin indicate time steps, and the propositions to the right are (some of) the beliefs in the KB at those times. Among the latter are beliefs of the form $\mathit{Now}(t)$, i.e., the logic "knows" what time it is. This time-stratified knowledge representation and reasoning is crucial to many applications of active logics, from deadline-coupled planning [4], to time-sensitive inference in general [1], to contradiction-detection [20], to discourse pragmatics [21, 22] (allowing the introduction of temporal subtleties into certain formal treatments of presupposition such as [23]).

In the example above, at step $i + 1$, $B$ has just been inferred from $A$ and $A \rightarrow B$ at step $i$. Also illustrated is the fact that beliefs at one step need not be "inherited" to the next, e.g., $\mathit{Now}(i)$ is not inherited to step $i + 1$. This disinheritance feature can also be applied to other beliefs, and is important in dealing with contradictory beliefs. When contradictions are encountered, active logic can, first of all, disinherit the contradictands so they do not cause further untrustworthy beliefs, and second, retrace its history of inferences to examine what led to the contradiction, performing metareasoning concerning which of these warrants continued belief [20, 21]. (For certain domains in which the automated system is a helper or advice-taker, it can also simply pass along the contradictory situation to a human user and await advice.) Indeed, a primary aim of our research into active logics has been to explore the extent to which contradictions may be categorized and generic strategies found that successfully resolve particular contradiction types. The central point for now, however, is that our use of active logic is based on its expressive power, and its ability to support the kinds of metareasoning we appear to need for a fully situated agency.

In addition to providing support for the desiderata mentioned above — flexible, non-monotonic, real-time reasoning — active logic's time-sensitivity helped Al in other ways. For instance, the fact that Al recognized that its reasoning itself took time allowed it to realize that optimality is not necessarily desirable, especially when there is a deadline approaching. More fundamentally, the active logic treatment of time allowed Al to keep track of changes: that the battery level is $X$ now does not imply that this is still true 5 hours later; active logic's $\mathit{Now}(i)$ predicate provides a natural and efficient way to deal with both reasoning *in* time, and also reasoning *about* time. Further, the history mechanism in active logic gave Al an opportunity to spot certain pattern in its past behavior, which helped it improve its future behavior (or, speaking more generally, the time-situatedness of Al's reasoning provides a natural framework to capture the future-orientation of agency, and the past-orientation required by learning).

Finally, because of the uniformity of a logic-based system, precomputation is seamlessly integrated into goal based problem solving through forward- and backward-chaining reasoning mechanisms. And the expressiveness of active logic made it possible to store the meta-rules about such things as interestingness of theorems, or the value and meaning of efficiency, which gave Al the basis for certain kinds of bootstrapping. These features allow for learning behavior without traditional learning mechanisms (explanation base learning, inductive learning, etc.) being explicitly programmed.

## 6 Conclusions

The so called *No Free Lunch Theorem* [24] states that "all algorithms that search for an extremum of a cost function perform exactly the same, when averaged over all possible cost functions." In other words, without domain specific structural assumptions of the problem, no algorithm can be expected to perform better on average than simple blind search. This result appears to be a cause for pessimism for researchers hoping to devise domain-independent methods to improve problem solving performance. But on the other hand, this theorem also provides compelling reason for embracing the notion of continual computation, which can be seen as a way to exploit domain dependent information in a domain independent way.

However, to take advantage of continual computation, we need to be able to express the concept of interestingness in a way sufficient to guide computation to profitable direction. Interestingness touches on the ultimate uncertainty: what to do next? Although utility theory has its place, we argued that there are aspects of interestingness not susceptible to utility based analysis. We believe that a forward and backward chaining capable logic system such as active logic, with its expressiveness, time sensitivity, and reflective ability to reason about theorems, proofs and derivations, is well-positioned to take advantage of the opportunity offered by continual computation, and can serve as a solid basis for the realization of intelligent agency.

## References

1. Elgot-Drapkin, J., Perlis, D.: Reasoning situated in time I: Basic concepts. Journal of Experimental and Theoretical Artificial Intelligence **2** (1990) 75–98
2. Elgot-Drapkin, J., Kraus, S., Miller, M., Nirkhe, M., Perlis, D.: Active logics: A unified formal approach to episodic reasoning. Technical report, Computer Science Department, University of Maryland (1996)
3. McCarthy, J.: Artificial intelligence, logic and formalizing common sense. In Thomason, R., ed.: Philosophical Logic and Artificial Intelligence. Klüver Academic (1989)
4. Nirkhe, M., Kraus, S., Miller, M., Perlis, D.: How to (plan to) meet a deadline between *now* and *then*. Journal of logic computation **7** (1997) 109–156
5. Maes, P.: Issues in computational reflection. In Maes, D.N.P., ed.: Meta-Level Architectures and Reflection. Elsevier Science Publishers B.V. (North-Holland) (1988) 21–35

6. Horvitz, E.: Models of continual computation. In: Proceedings of the 14th National Conference on Artificial Intelligence and 9th Innovative Applications of Artificial Intelligence Conference (AAAI-97/IAAI-97), Menlo Park, AAAI Press (1997) 286–293
7. Horvitz, E.: Principles and applications of continual computation. Artificial Intelligence **126** (2001) 159–196
8. Bibel, W.: Let's plan it deductively! In: IJCAI. (1997) 1549–1562
9. Lenat, D.B. In: AM: Discovery in Mathematics as Heuristic Search. McGraw-Hill, New York, NY (1982) 1–225
10. Lenat, D.B.: Theory Formation by Heuristic Search. Artificial Intelligence **21** (1983) 31–59
11. Lenat, D.B., Brown, J.S.: Why AM and EURISKO appear to work. Artificial Intelligence **23** (1984) 269–294
12. Colton, S., Bundy, A.: On the notion of interestingness in automated mathematical discovery. In: AISB Symposium on AI and Scientific Discovery. (1999)
13. Frankfurt, H.: The Importance of What We Care About. Cambridge University Press, Cambridge, UK (1988)
14. Bratman, M.: Faces of Intention. Cambridge University Press, Cambridge, UK (1999)
15. Bratman, M.E.: Intention, Plans, and Practical Reason. Harvard University Press (1987)
16. Aristotle: Nicomachean Ethics. N/A (350BC)
17. Broadie, S.: Ethics with Aristotle. Oxford University Press, Oxford, UK (1995)
18. Elgot-Drapkin, J., Kraus, S., Miller, M., Nirkhe, M., Perlis, D.: Active logics: A unified formal approach to episodic reasoning. Technical report, Computer Science Department, University of Maryland (1996)
19. Purang, K., Purushothaman, D., Traum, D., Andersen, C., Traum, D., Perlis, D.: Practical reasoning and plan execution with active logic. In: Proceedings of the IJCAI'99 Workshop on Practical Reasoning and Rationality. (1999)
20. Miller, M., Perlis, D.: Presentations and this and that: logic in action. In: Proceedings of the 15th Annual Conference of the Cognitive Science Society, Boulder, Colorado (1993)
21. Gurney, J., Perlis, D., Purang, K.: Interpreting presuppositions using active logic: From contexts to utterances. Computational Intelligence **13** (1997) 391–413
22. Traum, D., Andersen, C., Chong, Y., Josyula, D., O'Donovan-Anderson, M., Okamoto, Y., Purang, K., Perlis, D.: Representations of dialogue state for domain and task independent meta-dialogue. Electronic Transactions on Artificial Intelligence (forthcoming)
23. Heim, I.: Presupposition projection and the semantics of attitude verbs. Journal of Semantics **9** (1992) 183–221
24. Wolpert, D.H., Macready, W.G.: No free lunch theorems for optimization. IEEE Transactions on Evolutionary Computation **1** (1997) 67–82