Appears in *AI Magazine, Dec. '93.*

Intelligence without Robots (A Reply to Brooks)

Oren Etzioni
Department of Computer Science and Engineering
University of Washington
Seattle, WA 98195
(206) 685-3035
etzioni@cs.washington.edu

December 18, 1994

Abstract

In his recent papers, entitled "Intelligence without Representation and "Intelligence without Reason," Brooks argues for studying complete agents in real-world environments and for mobile robots as the foundation for AI research. This article argues that, even if we seek to investigate complete agents in real-world environments, robotics is neither necessary nor sufficient as a basis for AI research. The article proposes real-world *software* environments, such as operating systems or databases, as a complementary substrate for intelligent-agents research, and considers the relative advantages of software environments as testbeds for AI. First, the cost, effort, and expertise necessary to develop and systematically experiment with software artifacts are relatively low. Second, software environments circumvent many thorny, but peripheral, research issues that are inescapable in physical environments.

Brooks's mobile robots tug AI towards a bottom-up focus in which the mechanics of perception and mobility mingle inextricably with, or even supersede, core AI research. In contrast, the *softbots* (software robots) we are advocating facilitate the study of classical AI problems in real-world (albeit, software) domains. For example, our UNIX[1] softbot has led us to investigate planning with incomplete information, interleaving planning and execution, and a host of related high-level issues.

[1] UNIX is a trademark of AT&T Bell Labs.

Introduction

In his recent papers, entitled "Intelligence without Representation" [4] and "Intelligence without Reason" [3], Brooks propounds a number of positions including:

- *Complete agents in real-world environments:* "At each step we should build complete intelligent systems that we let loose in the real world with real sensing and real action" [4, page 140].
- *Robotics as the foundation for AI:* "The agents should be embodied as mobile robots... the new approach can be extended to cover the whole story, both with regards to building intelligent systems and to understanding human intelligence." [3, page 585].

This article argues that, even if we accept Brooks's first position and seek to build complete agents in real-world environments, we need not accept robotics as *the* foundation for AI. Clearly, robotics is an important and challenging enterprise, with much to contribute to AI. However, the article challenges Brooks's position that the primary path to progress in AI is "to study intelligence from the bottom up, concentrating on physical systems (e.g., mobile robots), situated in the world, autonomously carrying out tasks of various sorts" [3, page 569].

The article proposes real-world *software* environments, such as operating systems or databases, as a substrate for intelligent-agents research. Over the years, software environments have been explored as domains for machine learning [5, 6], intelligent user interfaces [21], planning [2], distributed AI [17, 19], and more. We argue for a unified conception: complete, intelligent agents that interact with real-world software environments by issuing commands and interpreting the environments' feedback. We refer to such agents as *softbots* (*soft*ware ro*bots*) [9]:

- A softbot's effectors are commands transmitted to the external environment in order to change its state (e.g., UNIX shell commands such as `mv` or `compress`).
- A softbot's sensors are commands that provide the softbot with information about its external world (e.g., `pwd` or `ls` in UNIX).

Softbots offer the methodological advantages of investigating complete agents in real-world environments without the overhead associated with robotic agents.

The remainder of this article is organized as follows. First, we show how softbots satisfy Brooks's desiderata for AI research vehicles. Second, we consider some advantages of softbots as a substrate for AI research.

Brooks's Arguments and Softbots

Brooks advances a number of arguments for his positions. Below, we consider his methodological arguments for building complete agents that operate in real-world environments, and

his argument for "embodiment" as a way to endow internal agent processing with meaning. In both cases, we show that these arguments apply equally well to softbots, a possibility that Brooks does not consider. Finally, we review and critique Brooks's evolutionary argument for robotics.

Engineering Methodology Argument

Brooks writes, "... I, and others, believe that human level intelligence is too complex and little understood to be correctly decomposed into the right subpieces at the moment and even if we knew the subpieces we still wouldn't know the right interfaces between them." [4, page 140]. As Mitchell *et al.* put it: "[the] reductionist research strategy has reached the point of diminishing returns." [16, page 352]. While both statements are quite strong, it seems clear that developing complete or integrated agent architectures has a distinct methodological advantage: the researcher is less likely to make unrealistic assumptions about the interfaces between different components of the architecture and about what each component will compute.

Given that one is committed to developing complete agents, Brooks argues that the agents should be tested in the real world: "with a simplified world... it is very easy to accidentally build a submodule of the systems which happens to rely on some of those simplified properties... the disease spreads and the complete system depends in a subtle way on the simplified world" [4, page 150]. Thus, Brooks is opposed to simulated worlds. After all, the infamous Blocksworld is just yesterday's simulated world.

The softbot paradigm escapes these quandaries by committing to full realism at every step. Softbots operate in dynamic, real-world environments that are not engineered by the softbots' designers. In the UNIX environment, for example, other agents (particularly humans) are continually changing the world's state by logging in and out, creating and deleting files, etc. Softbots are forced to cope with changes in their environment (where did that file go?) in a timely fashion. To succeed, softbots have to make sense of the flow of information through their limited bandwidth sensors, and respond appropriately.

Brooks emphasizes that an agent ought to have some purpose; it ought to be useful (cf. [18]). The preponderance of problems such as *feature shock* (the paralysis a user feels when facing a bewildering array of complex, poorly-documented features [12]) and *information anxiety* (a user's emotional response to the increasing volume and diversity of electronic data [15, 22]) suggest that there is no shortage of useful tasks for a softbot. Some simple examples are:

- Filtering electronic mail, and sending routine messages such as meeting reminders, talk announcements, and so on.
- Scheduling meetings [5, 14].
- Performing system maintenance tasks (e.g., around-the-clock intrusion detection).

In short, softbots satisfy every facet of Brooks's engineering methodology.

Symbol Grounding Argument

Brooks claims that "only through a physical grounding can any internal symbolic or other system find a place to bottom out, and give 'meaning' to the processing going on within the system" [3, page 584]. Standard semantic accounts of representational languages define 'meaning' and 'truth' in terms of an underlying model or logical interpretation. But what do the symbols in the underlying model mean? Brooks argues that only the physical world can ground an agent's internal representation. This rather abstract observation actually has practical ramifications for intelligent agents.

As Agre and Chapman put it: in classical AI planners, the truth of a Blocksworld proposition such as `on(a,b)` is determined by checking whether a relation corresponding to `on` applies to objects corresponding to `a` and `b`. The check is performed in the planner's model, not in the external world. Similarly, an agent satisfies the goal `on(a,b)` by updating its internal model to include the effects of executing the action `stack(a,b)`, not by interacting with the external world.

This "practice of allowing primitive actions to traffic in constant symbols" hides an important problem [1]. Since physical entities do not have tags associated with them, saying "I correspond to internal symbol `a`," an agent operating in the physical world has to develop methods that reliably map from perceptual experiences *in the world* to internal representations and conversely. This process, and related problems of linking perception with internal representation, are ignored by classical AI planners, but have to be confronted by a robotic agent operating in a physical environment.

Again, this argument supports the softbot paradigm equally well. In contrast to a Blocksworld-style simulated world, there is no privileged relationship between a softbot's internal symbols and the entities in its external world. For instance, suppose the softbot is instructed to format and print the most recent draft of a particular AAAI paper represented internally by `file-object-35`. The softbot has to decide whether the file called `learning.tex`, which it perceives in the directory `/ai/papers/`, corresponds to its internal symbol `file-object-35` or not. The software objects in the softbot's external world give meaning to its internal symbols. Although the mechanics of software perception are more manageable, and the nuisance of sensory noise is eliminated, the fundamental problem of mapping perceptual experiences to internal symbols remains.

Evolutionary Time Argument

Brooks points out that biological evolution, "spent" most of its multi-billion year history developing insects, reptiles, and primates. Humans arrived a mere 2.5 million years ago, and invented writing only recently. Brooks writes "this suggests that problem solving behavior, language, expert knowledge and application, and reason, are all pretty simple once the essence of being and reacting are available" [4, page 141]. Based on this observation, Brooks advocates studying intelligence 'bottom up,' starting with insects, eventually moving up to reptiles, and so on.

Whatever the merits of Brooks's bottom-up research strategy, his evolutionary argument

has to be elaborated. Brooks argues that since higher cognitive functions appeared, quite recently on an evolutionary time scale (a mere 2.5 million years ago), they are "pretty simple" in some sense. This claim presupposes a direct relationship between evolutionary time and some undefined measure of complexity. However, evolution is not a smooth, gradual process. Many evolutionary theorists subscribe to the theory of *punctuated equilibria* which asserts that the rate of evolutionary change is highly variable. As Gould puts it in a popular account, "the fossil record with its abrupt transitions offers no support for gradual change, and the principle of natural selection does not require it—selection can operate rapidly" [10, page 188]. We should not underestimate the amount of evolutionary change underlying our higher cognitive functions.

The vagaries of evolutionary theory aside, Brooks does not explain why biological evolution is relevant to AI research methodology. Suppose natural selection constrained evolution to 'design' organisms whose chance to reproduce is maximal, preferring quick reflexes to higher cognitive functions. Are AI systems subject to the same constraints? Certainly, softbots are not. On the other hand, suppose we accept the relevance of evolution to AI. Shouldn't we be emulating evolution much more closely than Brooks suggests? Brooks does not justify 'skipping' the billions of years evolution spent developing multi-celled organisms. Isn't it equally plausible to argue that AI should focus on developing the appropriate hardware (i.e., designing and manufacturing simple organisms), and the rest will fall into place relatively quickly? On what basis does Brooks conclude that following evolution at a very coarse grain (i.e., robotics before higher cognitive functions) is appropriate?

The Argument for Softbots

The previous section showed that Brooks's methodological arguments actually support the softbot paradigm, and called into question his evolutionary argument. This section presents an independent argument for softbots. The argument has both weak and strong versions. The weak version is straight forward. Software environments (e.g., databases, computer networks, operating systems) are the subject of intense study in computer science; software agents are gaining prominence outside AI (e.g., Knowbots [11]), demonstrating their intrinsic interest. Software environments are not idealizations of physical environments; developing softbots is a difficult and exciting challenge in its own right. This challenge necessitates its own research programme; developing mobile robots as a basis for softbots is about as plausible developing softbots as a basis for mobile robots. Hence, robotics is not sufficient as a foundation for AI. Softbotics and robotics are complementary methodologies for investigating intelligent agents in real-world environments. Clearly, physically-oriented research issues (e.g., overcoming sensor noise, motion planning, representing liquids, shapes, etc.) are best studied in physical environments, and software issues (e.g., responding to error messages, cloning softbots on remote machines, modeling databases, users, etc.) are best studied in software.

The strong version of the argument is that the study of many core AI issues is facilitated

by the softbots framework and potentially hindered by robotic testbeds.[2] An agent testbed shapes and directs one's research, providing a source of intuitions, motivating examples, simplifying assumptions, stumbling blocks, test cases, etc. Robotic testbeds lead one to focus on robotics. Thus, many core AI issues such as planning with incomplete information, grounding of internal symbols, learning from experiments, and more, are better studied in software domains. Brooks's "complete agents in real-world environments" methodology is attractive, but building mobile robots is *not* necessary to implement it. In many ways, softbots are preferable. Below, we enumerate the pragmatic advantages of software environments over physical environments in support of this claim.

In principle, mobile robots offer excellent testbeds for AI research. In practice, building intelligent systems that successfully interact with an unpredictable physical environment is a rigorous challenge, given existing technology. The cost of such robots (including laser range finders, sonars, grippers, television cameras, etc.) is non-trivial, and the effort and expertise required to assemble and operate such an apparatus are considerable.

Conducting experiments using mobile robots is often time-consuming and difficult. Experiments are frequently hampered by a wide variety of hardware difficulties and malfunctions [4, 20]. Days and even weeks go by in which the robot is not operational. Even when the robot is operational, the mean time between failures can be short. As a result, carrying out empirical AI research using robots can be quite tedious and slow. Furthermore, while robotic task environments are much more realistic than the Blocksworld, introducing the problems of sensing, uncertainty, and noise, the environments often remain highly unrealistic due to hardware limitations. Realism is lost when the agent's external environment is manipulated to improve the agent's performance. For instance, Brooks describes how Shakey, SRI's mobile robot, operated in rooms where "the walls were of a uniform color, and carefully lighted, with dark rubber baseboards, making clear boundaries with the lighter colored floor ..." [3]. More recent AI robots operate in more realistic environments, but are restricted to simple tasks such as avoiding walls and fetching soda cans. Much more realistic robots have been built, of course, but they require orders of magnitude *more* investment of time, money, and expertise in robotics before core AI research can take place.

Brooks acknowledges the frustrations and pragmatic difficulties attendant on AI research utilizing mobile robotic agents. The mean time between failures, for one of his robots, was as short as fifteen minutes [3, page 587]. Brooks himself writes "experimental work with physical Creatures is a nontrivial and time consuming activity...as of mid-1987, our work in learning is held up by the need to build a new sort of video camera and high-speed low-power processing box to run specially developed vision algorithms at 10 frames per second." [4, page 158].

Thus, software task environments have a number of pragmatic advantages over physical ones. First, the mean time between hardware failures is much greater for a workstation supporting a software environment than for a mobile robot. Second, rebooting a workstation and restoring a softbot "from disk" is much easier than fixing a broken gripper in a physical

[2]Note that, in contrast to Brooks [3, page 578], we believe that classical approaches (e.g., current work on knowledge representation and on planning) still have much to contribute to AI.

robot or identifying and replacing a malfunctioning chip. As a result, software experiments are easier to perform, control, and repeat than robotic experiments, facilitating systematic experimental research of the sort advocated by [13] and others. In addition, software facilitates the dissemination and replication of research results. It is straight forward to distribute multiple copies of a softbot, whereas the distribution of research-prototype robots is difficult.

Software environments are particularly well-suited for agent research. Providing a softbot with basic execution and sensing mechanisms is easy. For instance, our UNIX softbots rely on a simple program that sends and receives strings from a UNIX shell. Once the low-level problems associated with vision (edge detection, stereoscopy, occlusion, sensory noise, etc.) and other physical sensing modalities are eliminated, fascinating high-level problems (e.g., how to plan sensory operations) emerge. Many difficult representation and reasoning problems (e.g., liquids, shapes, physical actions, and much more) are avoided. This is a disadvantage if one is interested in studying these problems, but an advantage if one wants to focus on agent research and finds the formalization of physical knowledge to be a distraction. Finally, many software environments are benign, giving a softbot an opportunity to survive and engage in useful activities over time.

To summarize, software environments have three main advantages over physical ones:

- **pragmatic convenience:** the cost, effort, and expertise necessary to develop and systematically experiment with physical artifacts far exceeds that associated with software artifacts.
- **research focus:** software environments circumvent many thorny, but peripheral, research issues that have to be addressed in physical environments.
- **easy embodiment:** as a consequence of the first two items, providing an agent with effective sensors and actuators is relatively easy in software environments.

Yet, in contrast to simulated physical worlds, software environments are readily available (sophisticated simulations can take years to develop and perfect) and intrinsically interesting. Furthermore, software environments are *real.*

Conclusion

This article argued that bottom-up research on mobile robots, though valuable, is neither necessary nor sufficient as a foundation for core AI research. Robotics is not sufficient for AI, because the challenge of developing intelligent software agents (or softbots) dictates its own research agenda; robotics is not necessary, because many AI issues can be studied profitably in real-world software environments such as operating systems or databases.

In fact, software environments are particularly well-suited for the study of complete intelligent agents. The pragmatic convenience of software environments facilitates rapid development of, and systematic experimentation with, software agents. Providing a softbot with effective sensors and actuators is relatively straight forward, enabling researchers to

focus on high-level issues, and circumventing many thorny, but peripheral, problems that are inescapable in physical environments.

A priori arguments only carry so much weight, though. The real test of the softbot paradigm is whether it will yield fundamental contributions to core AI. Our language for planning with incomplete information (UWL [7]) is a modest example, but the jury is still out. To paraphrase Brooks [4, page 158], only experiments with real softbots in real software worlds can answer the natural doubts about our approach. Time will tell.

A UNIX Softbot (INSERT SECTION)

To make the softbot paradigm concrete we briefly describe a general-purpose UNIX softbot (called Rodney[3]) under development at the University of Washington. See [8] for a comprehensive description of Rodney. Rodney accepts high-level user goals and dynamically synthesizes the appropriate sequence of UNIX commands. Rodney executes the sequence, recovering from errors and retrying commands if necessary. The following are examples of the types of requests that Rodney handles successfully:

1. **Notification requests:**

   - Notify me if my disk utilization exceeds eighty percent.
   - Let me know when Neal logs into his workstation.
   - Show me any posts containing the string "bicycle" that appear on the market bulletin board this week.

   The choice of notification medium (a beep, a message displayed on the screen, or an e-mail message.) is under the softbot's control, as is the means of monitoring the events in question.

2. **Enforcing constraints:**

   - Keep all files in the directory `/papers` group-readable.
   - Ensure that all my Postscript files are current (i.e., automatically generate a new postscript file whenever the corresponding TEX file is modified).

3. **Locating and manipulating objects:**

   - Print my file on any nearby printer that is not busy, and tell me where to find it when the print is done.
   - Locate Melanie Mitchell
     (utilizing `whois`, `netfind`, `staffdir`, `finger`, and more)

[3] Brooks's early robots were named Herbert, Allen, Seymour, etc.

These classes are neither exhaustive nor mutually exclusive, but illustrate our main point: Rodney enables a user to specify *what* to accomplish, leaving the decision of *how* to accomplish it to the softbot. In essence, Rodney raises the level of discourse between the user and the machine. This goal-oriented approach offers a number of advantages over conventional operating-system interfaces. Although an expert programmer could conceivably write a shell script to satisfy the individual goals we have listed above, the programmer could not create a shell script to accomplish every conceivable user goal or combination of goals. Furthermore, as new system facilities become available, the shell scripts would need to be continually updated and modified.

In contrast, the softbot represents UNIX commands (and applications such as `netfind`) as STRIPS-style operators, and utilizes general-purpose planning algorithms to dynamically generate a plan that satisfies the user's goals [7, 8]. Once the softbot "knows" about a new facility, that facility becomes immediately available to its planning process, and is automatically invoked to satisfy relevant user goals. Furthermore, unlike a shell script, the softbot is not locked into a rigid control flow. It fluidly backtracks from one option to the next, based on information collected at run time. If one printer is jammed, the softbot will try another; if `whois` fails, the softbot will try `netfind`, and so on. The nature and ordering of the softbot's options are subject to learning, which enables the softbot to improve its performance over time.

Acknowledgments

I thank Keith Golden, Neal Lesh and Richard Segal for helping to make Rodney real. Thanks are also due to Steve Hanks, Dan Weld, Denise Draper, and Mike Williamson for their collaboration in designing UWL, and to Hank Levy and Dan Weld for numerous discussions and contributions to the softbots project. Other contributors to the project include Greg Fichtenholtz, Terrance Goan, Rob Spiger, and David Simmons. This research was funded in part by Office of Naval Research Grant 92-J-1946, by National Science Foundation Grant IRI-9211045, and by an NSF National Young Investigator Award.

Biography

Oren Etzioni is an assistant professor at the University of Washington. He received his B.A. from Harvard University in 1986, and his Ph.D. in Computer Science from Carnegie Mellon University in 1990. In 1993, he received a National Young Investigator Award from the NSF. His research interests include machine learning, planning, and software agents.

References

[1] P. Agre and D. Chapman. What are Plans for? In *Designing Autonomous Agents*, pages 17–34. MIT/Elsevier, 1990.

[2] Y. Arens, C. Y. Chee, C.-N. Hsu, and C. A. Knoblock. Retrieving and integrating data from multiple information sources. *International Journal on Intelligent and Cooperative Information Systems*, 1993. In press.

[3] R. A. Brooks. Intelligence without reason. In *Proceedings of the Twelveth International Joint Conference on Artificial Intelligence*, pages 569–595, Sam Mateo, California, 1991. Morgan Kaufmann.

[4] R. A. Brooks. Intelligence without representation. *Artificial Intelligence*, 47:139–160, 1991.

[5] L. Dent, J. Boticario, J. McDermott, T. Mitchell, and D. Zabowski. A Personal Learning Apprentice. In *Proceedings of AAAI-92*, pages 96–103, Sam Mateo, California, July 1992. Morgan Kaufmann.

[6] T. G. Dietterich. *Constraint Propagation Techniques for theory-driven data interpretation.* PhD thesis, Stanford University, 1984.

[7] O. Etzioni, S. Hanks, D. Weld, D. Draper, N. Lesh, and M. Williamson. An approach to planning with incomplete information. In *Proceedings of the Third International Conference on Principles of Knowledge representation and reasoning*, pages 115–125, Sam Mateo, California, 1992. Morgan Kaufmann.

[8] O. Etzioni, N. Lesh, and R. Segal. Building softbots for unix (preliminary report). Unpublished Manuscript, 1992.

[9] O. Etzioni and R. Segal. Softbots as testbeds for machine learning. In *Working Notes of the AAAI Spring Symposium on Knowledge Assimilation*, Menlo Park, CA, 1992. AAAI Press.

[10] S. J. Gould. *The Panda's Thumb.* W. W. Norton, New York, NY, 1980.

[11] R. E. Kahn and V. G. Cerf. An open architecture for a digital library system and a plan for its development. Technical report, Corporation for National Research Initiatives, March 1988.

[12] L. Kleinrock. Distributed systems. *Computer*, 18:90–103, November 1985.

[13] P. Langley and M. Drummond. Toward an experimental science of planning. In K. P. Sycara, editor, *Proceedings of the workshop on innovative approaches to planning, scheduling, and control*, Sam Mateo, California, 1990. Morgran Kaufmann.

[14] P. Maes and R. Kozierok. Learning interface agents. In *Proceedings of INTERCHI-93*, 1993.

[15] E. Messinger, K. Shoens, J. Thomas, and A. Luniewski. Rufus: the information spunge. Technical Report RJ 8294, IBM Almaden research center, August 1991.

[16] T. M. Mitchell, J. Allen, P. Chalasani, J. Cheng, O. Etzioni, M. Ringuette, and J. C. Schlimmer. Theo: A framework for self-improving systems. In K. VanLehn, editor, *Architectures for Intelligence.* Lawrence Erlbaum, Hillsdale, NJ., 1991.

[17] J. Rosenschein. Synchronization of Multi-Agent Plans. In *Proceedings of AAAI-82*, 1982.

[18] R. C. Schank. Where's the AI? *AI Magazine*, 12(4):38–49, 1991.

[19] Y. Shoham. Agent-oriented programming. *Artificial Intelligence*, 60(1):51–92, March 1993.

[20] M. Tan. *Cost-sensitive Robot Learning.* PhD thesis, Carnegie Mellon University, 1991. Available as technical report CMU-CS-91-134.

[21] R. Wilensky, D. Chin, M. Luria, J. Martin, J. Mayfield, and D. Wu. The Berkeley UNIX Consultant project. *Computational Linguistics*, 14(4):35–84, 1988.

[22] R. S. Wurman. *Information Anxiety.* Doubleday, 1989.

Letter to the Editor, in reply to letters by Drs Miller, Stein, and Wellman. All letters appeared in AI Magazine (Summer '94 issue).

To readers who have just tuned in, here is the story to date. In a series of important papers, Rodney Brooks argued for a number of positions including Build Complete Agents in Real-world Environments, Robotics is the Foundation of AI, The World is Its Own Best Model, and many more. In a recent AI Magazine article (December '93), I argued that we can accept Brooks's compelling methodological idea (Build Complete Agents in Real-World Environments), without buying into the rest of the Brooksian agenda. In particular, we can build *softbots* — complete AI agents that interact with real software worlds such as the internet. Most people, the letter writers included, seem to accept this idea. (The softbot paradigm was more ground breaking when we first articulated it in 1991.)

My article did not consider other elements of Brooks's multi-faceted position. Instead, I made a second, more controversial, claim: "the study of many core AI issues is facilitated by the softbot framework and potentially hindered by robotic testbeds." Note that this claim only speaks to someone who is interested in core AI (i.e., the fields of knowledge representation, automated reasoning, planning, machine learning, etc.). I did not attempt to argue for core AI — only to explore the implication of different agent testbeds for its investigation.

I'd now like to clarify a number of points raised in the letters received.

Softbots $\neq$ UNIXbots: despite Miller's focus on UNIX, a softbot may interact with *any* real-world software environment. My own group is currently focusing on internet softbots, and other groups are investigating a wide range of generic tasks including e-mail management, information gathering, visitor scheduling, and more.

Softbots $\neq$ shell scripts: the representation of software commands as planning operators is a central aspect of our softbot's architecture. As a result, we are able to provide a human user with an expressive goal language that allows logical combinations (e.g., negation, disjunction, and universal quantification) of the predicates mentioned in the operators. When a new system facility becomes available, we need only write the appropriate operator models and search control rules to update the softbot. We are also investigating the use of learning techniques to help automate this task. This approach offers many advantages over trying to code and maintain a shell script for each goal the user might have in mind. As Dan Weld is fond of saying *"a softbot is worth a thousand shell scripts."*

Softbots leverage existing software: Miller points to the challenge of writing UNIX. But just as Brooks did not have to erect an office building to for his artificial insects, we did not need write UNIX (or other software environments) to build our softbots. Softbots interact with *existing* software environments. Contrary to Miller's letter, we did not pay for a UNIX source license, because we never look at the source code. Instead, our softbot relies on the "pre-fabricated" tools and utilities available to human computer users — tools for sending mail, printing files, internet utilities such as `netfind`, etc. Mobile robots have yet to achieve the physical analog — using vacuum cleaners and lawn mowers, reading maps, etc.

Softbots and humans share primitive tools: in many cases, the software commands used by our softbot are similar, if not identical, to those used by humans. Consider the task of printing a file. Both the human and the softbot use the UNIX command `lpr`. To monitor a print job, both use `lpq`. The challenge to the softbot is cognitive: which printer should it use? What if the closest printer is busy or jammed? Should the softbot use a color printer? a high-speed one? What if its human "boss" is late to a meeting in another building? What if the file to-be-printed is confidential?

Softbot perception and actuation are relatively easy: the structured and reliable nature of most software tools makes softbot perception and actuation relatively easy. For example, parsing the highly structured output of `wc` (and likewise `pwd`, `hostname`, `lpq`, `ls`, and many other commands) is straightforward. Exceptions include tools that require natural language understanding, on-line video and audio, etc. Note, though, that postscript files can be converted to ASCII and that information read by humans in a graphical format (e.g. a MOSAIC page) often has a softbot-accessible format as well (e.g. an HTML file).

Stein is correct that building a softbot has some overhead, but that overhead is relatively small. She speculates that we are forced to deal with arcane and "bizarre" details of UNIX. In fact, less than 10% of the softbot code is devoted to the low level of mechanics of perception and actuation, and much of that code was written by undergraduate programmers. The bulk of the softbot (and of our research!) involves domain independent algorithms for planning, reasoning, and learning.[4]

In contrast, robotic vision, hearing, and fine motor coordination are far inferior to their human correlates. Consider fetching a printout. For a robot, this task is fraught with motor and sensory pitfalls. How does the robot open the door to the printer room? How does it find the appropriate printout? How does it separate the pages of different printouts? To borrow Tom Dean's phrase, robots are "perceptually challenged." From a core AI point of view, the largest overhead associated with robotics is the *research* required to achieve adequate perception and actuation.

Is Robotics Necessary? there is no question that robotics is a valuable, necessary, and exciting field of endeavor. In the article I wrote "Clearly, robotics is an important and challenging enterprise, with much to contribute to AI." Nevertheless, a core AI researcher need not build robots to satisfy Brooks's Complete Agents methodology. Softbots are an attractive alternative. This is the sense in which "bottom-up research on mobile robots, though valuable, is neither necessary nor sufficient as a foundation for core AI research." I regret that this carefully qualified point was misinterpreted as the "outrageous conclusion that robotics research is utterly unnecessary."

General lessons from softbots: Stein raises a key question. Should we "expect lessons learned in the decidedly artificial world of the computer to carry over to the extremely different world in which we live?" Stein is dubious. Indeed, although parallels between physical and software worlds abound, there are also many differences. A softbot can easily

[4]In a recent project, we linked the softbot to the telephone network by using the Macintosh's telephony suite, and writing operators to model dialing a phone number, listening for a busy signal, etc. No change was necessary to the softbot's basic algorithms.

clone itself, store snapshots of itself on disk, even revert to a previous "self." Thus, as we delve more and more deeply into the investigation of software agency, we might find that softbots and humans diverge. Still, could we create a software intelligence, but fail to understand human intelligence? Shouldn't our theory of intelligence be general enough to encompass both software and physical agents?

In fact, we have been surprised by the extent to which ideas discovered and explored in software domains have natural physical analogues. For example, we have found that while the softbot cannot obtain complete information about all the files and people accessible on the internet, it can obtain complete information about a particular *locale*, such as a UNIX directory, a file archive, or an `FTP` site. We have devised a polynomial-time algorithm for inferring and updating logical sentences of this sort (Etzioni, Golden, & Weld, KR'94). Clearly, a locale need not be a software construct. A person (or a robot) can obtain information about the contents of a drawer, the occupants of a room, etc. Our algorithm is domain independent. I do not mean to suggest that softbots are a way of pursuing robotics — they are not. However, I want to emphasize that while we rely on UNIX tasks for illustration and validation, the theoretical results we report are domain independent.

Finally, as suggested by Wellman, the internet and the emerging national information infrastructure provide us with an opportunity to contribute not only to a deeper understanding of intelligence, but also to the U.S. economy and to society at large. The operating systems, information technology, and user interface communities are all struggling to make "cyberspace" palatable and accessible to millions of new computer users with tools such as `Mosaic`, `gopher`, `WAIS`, etc. In the next few years, the opportunity (and implicit challenge) we face is to demonstrate that AI technology can add value and power to this rapidly evolving set of services and tools.