Intelligence without Robots: A Reply to Brooks

Oren Etzioni

In his recent papers, entitled “Intelligence without Representation” and “Intelligence without Reason,” Brooks argues for mobile robots as the foundation of 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-agent research and considers the relative advantages of software environments as test beds 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 toward 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) I advocate facilitate the study of classical AI problems in real-world (albeit, software) domains. For example, the UNIX softbot under development at the University of Washington has led us to investigate planning with incomplete information, interleaving planning and execution, and a host of related high-level issues.

I challenge Brooks’s position that the primary path to progress in AI is “to study intelligence from the bottom up. . .” (Brooks 1991a, p. 569).

In this article, I propose real-world software environments, such as operating systems or databases, as a substrate for intelligent-agent research. Over the years, software environments have been explored as domains for machine learning (Dent et al. 1992; Dietterich 1984), intelligent user interfaces (Wilensky et al. 1988), planning (Arens et al. 1993), distributed AI (Rosenschein 1982; Shoham 1993), and more. I argue for a unified conception: complete, intelligent agents that interact with real-world software environments by issuing commands and interpreting the environments’ feedback. I refer to such agents as softbots (software robots): A softbot’s effectors are commands transmitted to the external environment to change its state (for example, UNIX shell commands such as mv or compress). A softbot’s sensors are commands that provide the softbot with information about its external world (for example, 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, I show how softbots satisfy Brooks’s desiderata for AI research vehicles. Second, I 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. Here, I 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, I show that these arguments apply equally well to softbots, a possibility that Brooks does not consider. Finally, I review and critique Brooks’s evolutionary argument for robotics.

Engineering Methodology Argument

Brooks (1991b, p. 140) 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.” As Mitchell et al. (1991, p. 352) put it, “[The] reductionist research strategy has reached the point of diminishing returns.” Although 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 unreal-istic assumptions about what the interfaces are between different com-
ponents of the architecture and what each component will compute.

Given that one is committed to developing complete agents, Brooks (1991b, p. 150) argues that the agents
should be tested in the real world: "[W]ith a simplified world...it is very
easy to accidentally build a submod-
ule 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."
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 soft-
bots' designers. In the UNIX environ-
ment, for example, other agents (par-
ticularly humans) are continually
changing the world's state by logging
in and out, creating and deleting
files, and so on. Softbots are forced to
cope with changes in their environ-
ment (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 (see Schank [1991]). The
preponderance of problems such as
feature shock (the paralysis a user feels
when facing a bewildering array of
complex, poorly documented features
[Kleinrock 1985]) and information
anxiety (a user's emotional response
to the increasing volume and diversity
of electronic data [Messinger et al.
1991; Wurman 1989]) 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 and talk announcements,
scheduling meetings (Dent et al.
1992; Maes and Kozierek 1993), and
performing system maintenance tasks
(for example, around-the-clock intru-
sion detection). In short, softbots sat-
sify every facet of Brooks's engineer-
ing methodology.

Symbol Grounding Argument

Brooks (1991a, p. 584) claims that
"only through a physical grounding
that these internal symbolic or other
system find a place to bottom out, and
give 'meaning' to the processing
going on within the system." Stan-
andard semantic accounts of representa-
tional languages define 'meaning '
and 'truth' in terms of an underlying
model or logical interpretation. How-
ever, what do the symbols in the
underlying model mean? Brooks argues
that only the physical world
can ground an agent's internal rep-
resentation. This rather abstract observa-
tion actually has practical ramifica-
tions for intelligent agents.

As Agre and Chapman put it, in
classical AI planners, the truth of a
blocks world 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 exter-
nal world. Similarly, an agent satisfies
the goal \(on(a,b)\) by updating its inter-
ernal 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 (Agre
and Chapman 1990). Because physi-
cal entities do not have tags associat-
ed with them saying "I correspond to
internal symbol a," an agent operat-
ing in the physical world has to devel-
ulp methods that reliably map from
perceptual experiences in the world to
internal representations, and con-
versely. This process and related prob-
lems of linking perception with inter-
nal representation are ignored by
classical AI planners but have to be
confronted by a robotic agent operat-
ing in a physical environment.

Again, this argument supports the
softbot paradigm equally well. In
contrast to a blocks world-style simu-
lated world, there is no privileged
relationship between a softbot's inter-
nal symbols and the entities in its
external world. For example, suppose
the softbot is instructed to format
and print the most recent draft of a
particular AI conference paper pre-
represented internally by the phrase
file-object-35. The softbot has to decide
whether the file called learning.tex,
which it perceives in the directory
\(\text{/ai/papers/}\), corresponds to its inter-
nal symbol file-object-35. 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 elim-
inated, the fundamental problem of
mapping perceptual experiences to
internal symbols remains.

Evolutionary Time Argument

Brooks points out that biological evo-
lution spent most of its multibillion-
year history developing insects, rep-
tiles, and primates. Humans arrived
a mere 2.5 million years ago and
invented writing only recently.
Brooks (1991b, p. 141) writes, "This
suggests that problem solving behav-
ior, language, expert knowledge and
application, and reason are all pretty
simple once the essence of being and
reacting are available." Based on this
observation, Brooks advocates study-
ing intelligence from the bottom up,
starting with insects, eventually mov-
ing up to reptiles, and so on.

Whatever the merits of Brooks's
bottom-up research strategy, his evo-
lutionary argument has to be elabo-
rated. Given that higher cognitive
functions appeared quite recently on
the evolutionary time scale (a mere
2.5 million years ago), Brooks argues
that higher cognitive functions are
"pretty simple" in a 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
188) puts it in a popular account,
"[T]he fossil record with its abrupt
transitions offers no support for grad-
ual change, and the principle of natu-
ral selection does not require it—selection can operate rapidly."
We should not underestimate the amount of evolutionary change underlying our higher cognitive functions.

The vagaries of evolutionary theo-
ry 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. However, 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 of evolution spent developing multicelled organisms. Isn’t it equally plausible to argue that AI should focus on developing the appropriate hardware (that is, 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 coarse grain (that is, 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 straightforward. Software environments (for example, databases, computer networks, operating systems) are the subject of intense study in computer science; software agents are gaining prominence outside AI (for example, KNOWBOTS [Kahn and Cerf 1988]), 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 program; developing mobile robots as a basis for softbots is about as plausible as 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 ori-

A UNIX Softbot

To make the softbot paradigm concrete, I briefly describe a general-purpose UNIX softbot, called ROODNEY (Brooks’s early robots were named Herbert, Allen, Seymour, and so on), under development at the University of Washington. (See Etzioni, Lesh, and Segal [1993] for a comprehensive description of ROODNEY). ROODNEY accepts high-level user goals and dynamically synthesizes sequences of UNIX commands that satisfy the goals. ROODNEY executes the sequences, recovering from errors and retrying commands if necessary. The following are examples of the types of user requests that ROODNEY handles successfully:

Notification Requests
- Notify me if my disk utilization exceeds 80 percent.
- Let me know when Neal logs in.
- 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.

Enforcing Constraints
- Keep all files in the directory /papers group-readable.
- Ensure that all my postscript files are current (that is, automatically generate a new postscript file whenever the corresponding \TeX file is modified).

Locating and Manipulating Objects
- Print my file on any nearby printer that is not busy, and tell me where to find it.
- Locate Melanie Mitchell (using whois, netfind, staffdir, finger, and more).

These classes of requests are neither exhaustive nor mutually exclusive but illustrate my main point: ROODNEY enables a user to specify what to accomplish, leaving the decision of how to accomplish it to the softbot. In essence, ROODNEY 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 I listed, 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 continually be 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 (Etzioni et al. 1992; Etzioni, Lesh, and Segal 1993). Once the softbot knows about a new facility, the facility becomes 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 tries another; if whois fails, the softbot tries 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.
ent research issues (for example, overcoming sensor noise; motion planning; representing liquids, shapes) are best studied in physical environments, and software issues (for example, responding to error messages; cloning softbots on remote machines; modeling databases, users) are best studied in software.

The strong version of the argument is that the study of many core AI issues is facilitated by the softbot framework and potentially hindered by robotic test beds. An agent test bed shapes and directs one’s research, providing a source of intuitions, motivating examples, simplifying assumptions, stumbling blocks, test cases, and so on. Robotic test beds lead one to focus on robotics. Thus, many core AI issues, such as planning with incomplete information, grounding internal symbols, and learning from experiments, 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.

Here, I enumerate some of the difficulties associated with mobile robotics research. In principle, mobile robots offer excellent test beds 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) is nontrivial, 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 (Brooks 1991b; Tan 1991). 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 tedious and slow. Furthermore, although robotic task environments are much
more realistic than the blocks world, introducing the problems of sensing, uncertainty, and noise, the environments often remain highly unrealistic because of hardware limitations. Realism is lost when the agent’s external environment is manipulated to improve the agent’s performance. For example, Brooks describes how SHAKEY, the SRI International 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” (Brooks 1991a, p. 577). 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 using mobile robotic agents. The mean time between failures for one of his robots was as short as 15 minutes (Brooks 1991a, p. 587). Brooks (1991b, p. 158) himself writes, “[E]xperimental 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.”

In contrast, software task environments have a number of pragmatic advantages. 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 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 Langley and Drummond (1990) and others. In addition, software facilitates the dissemination and replication of research results. The distribution of multiple copies of a softbot is straightforward, 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 example, 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, stereoscoppy, occlusion, sensory noise, and so on) and other physical sensing modalities are eliminated, fascinating high-level problems (for example, how to plan sensory operations) emerge. Many difficult representation and reasoning problems (for example, liquids, shapes, physical actions) are avoided, which is a disadvantage if you want to study these problems but an advantage if you want to focus on agent research and find the formalization of physical knowledge to

WINTER 1993    11
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, although 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 straightforward, 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. The University of Washington language for planning with incomplete information (UWL) (Etzioni et al. 1992) is a modest example, but the jury is still out. To paraphrase Brooks (1991b, p. 158), only experiments with real softbots in real software worlds can answer the natural doubts about our approach. Time will tell.

Acknowledgments

I thank Keith Golden, Neal Lesh, and Richard Segal for helping to make the university’s softbot, RODNEY, real. Thanks are also owed 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 softbot project. Other contributors to the project include Greg Fichtenholtz, Terrance Goan, Rob Spiger, and David Simmons. This research was funded in part by the Office of Naval Research, grant 92-J-1946; in part by the National Science Foundation (NSF), grant IRI-9211045; and in part by an NSF Young Investigator Award.
Notes
1. See the paper by myself and Richard Segal in the working notes for the 1992 AAAI Spring Symposium on Knowledge Assimilation, “Softbots as Test Beds for Machine Learning.”
2. Note that in contrast to Brooks (1991a, p. 578), I believe that classical approaches (for example, current work on knowledge representation and planning) still have much to contribute to AI.

References

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 January 1991. In 1993, he received a National Science Foundation Young Investigator Award. His research interests include machine learning, planning, and software agents.