Benchmarks, Testbeds, Controlled Experimentation, and the Design of Agent Architectures

Steve Hanks, Martha Pollack[1], Paul Cohen[2]

Department of Computer Science and Engineering
University of Washington
Seattle, WA 98195

Technical Report 93–06–05
June 17, 1993

[1] Department of Computer Science and Intelligent Systems Program, University of Pittsburgh
[2] Department of Computer Science, University of Massachusetts

Benchmarks, Testbeds, Controlled Experimentation, and the Design of Agent Architectures

Steve Hanks
Department of Computer Science & Engineering
University of Washington*

Martha E. Pollack
Department of Computer Science
and Intelligent Systems Program
University of Pittsburgh †

Paul R. Cohen
Department of Computer Science
University of Massachusetts ‡

June 18, 1993

Abstract

The methodological underpinnings of AI are slowly changing. Benchmarks, testbeds, and controlled experimentation are becoming more common. While we are optimistic that this change can solidify the science of AI, we also recognize a set of difficult issues concerning the appropriate use of this methodology. We discuss these issues as they relate to research on agent design. We survey existing testbeds for agents, and argue for appropriate caution in their use. We end with a debate on the proper role of experimental methodology in the design and validation of planning agents.

*Hanks was supported in part by NSF grants IRI-9008670 and IRI-9206733.

†Pollack was supported by the Air Force Office of Scientific Research, Contracts F49620-91-C-0005 and F49620-92-J-0422, by the Rome Laboratory (RL) and the Defense Advanced Research Projects Agency, Contract F30602-93-C-0038, and by an NSF Young Investigator's Award, IRI-9258392

‡Cohen was supported in part by the Defense Advanced Research Projects Agency, Contract F30602-91-C-0076

Contents

1 Introduction

In recent years, increasing numbers of AI research projects have involved *controlled experimentation*, in which a researcher varies the features of a system or the environment in which it is embedded, and measures the effects of those variations on aspects of system performance. At the same time, two research tools have gained currency: *benchmarks*, precisely defined, standardized tasks; and *testbeds*, challenging environments in which AI programs can be studied. In our view, the move toward more principled experimental methods is uncontroversially a good thing; indeed, we are optimistic that it will solidify the science of AI. However, we also recognize some issues concerning the appropriate use of these methods. First, benchmarks and testbeds no more guarantee important results than, say, microscopes and bunsen burners. They are simply part of the apparatus of empirical AI. It is up to the researcher to discriminate between uninteresting and important phenomena, and to follow up reports of experiments with thorough explanations of their results. Second, there is little agreement about what is a *representative* benchmark or testbed problem. A third and related concern is that results obtained with benchmarks and testbeds often are not general. Fourth, because benchmarks and testbeds are attractive to program managers and others who provide funding, there is a real danger that researchers will aim for the prescribed benchmark target when funding is perceived to be the reward. In sum, we are concerned that benchmarks and testbeds, if not carefully used, will provide only a comfortable illusion of scientific progress—controlled experimentation with reproducible problems and environments, and objective performance measures—but no generalizable, significant results.

Benchmarks and testbeds serve at least two different purposes. One is to provide metrics for comparing competing systems. Comparison metrics are valuable for some purposes, but performance comparisons do not constitute scientific progress unless they suggest or provide evidence for explanatory theories of performance differences. The scientific value of well-crafted benchmarks and testbeds is their power to highlight interesting aspects of system performance, but this value is realized only if the researcher can adequately explain why his or her system behaves the way it does.

The experimental control that can be achieved with testbeds can help us explain why systems behave as they do. AI systems are intended to be deployed in large, extremely complex environments, and testbeds serve as simplified, simulated versions of those environments, in which the experimenter has access to particular aspects of the environment, and other

aspects are allowed to vary randomly. The experimental process consists in the researcher varying the features of the testbed environment, the benchmark task, or the embedded system, and measuring the resulting effects on system performance. A fundamental question exists, however, about the viability of this approach. The concern grows out of the tension between realism and the possibility of experimental control. On the one hand, controlled experiments seem at least currently to be feasible only for simplified systems operating in highly idealized environments. On the other hand, our ultimate interest is not simplified systems and environments but, rather, "real-world" systems deployed in complex environments. It is not always obvious whether the lessons learned from the simplified systems are generally applicable, but neither is it obvious how to perform systematic experiments without the simplifications.

Researchers disagree about how best to proceed in light of this tension. One approach is to maintain systematicity in experiments while looking for ways to translate the results of the experiments into general principles that apply to more complex systems and environments. The alternative is to focus on more realistic systems and environments and try to conduct systematic experiments on them directly. Much of this paper will focus on a comparison of these approaches.

Although benchmarks, testbeds, and controlled experimentation are increasingly important in a number of subareas of AI, including, for example, natural-language understanding and machine learning, we will focus our discussion on its role in agent design. We begin, in the next section, by describing some of the criteria for good benchmarks and testbeds, and discussing some of the potential difficulties encountered in their design. In Section 3, we discuss the range of features that a testbed for agent design might have. In Section 4, we survey existing testbeds for agent design with these features in mind. Finally, in Section 5, we return to the general issue of experimental methodology in agent design, and discuss some unresolved questions concerning its use. Our points will become increasingly more controversial as the paper proceeds, and, indeed, by the end of the paper we will no longer speak with one voice.

2 Benchmarks and Testbeds

Benchmarks are a common tool in computer science. In the design of CPUs, for example, matrix multiplication is a good benchmark task because it is representative of an important class of numerical processing problems, which in turn is representative of a wider class of

computational problems—those that do not involve significant amounts of I/O. The matrix multiplication problem can be described precisely and rigorously. Moreover, matrix multiplication is illuminating: it tells the CPU designer something interesting about the CPU, namely, its processing speed. In other words, if we are interested in processing speed as a measure of performance, then matrix multiplication is a good benchmark: good performance on matrix multiplication problems predicts good performance on the large class of numerical tasks for which the processor is being designed.

An early benchmark task for AI planning programs was the Sussman anomaly (the "Three Block Problem") [Sussman 1975]. The Sussman anomaly helped many researchers elucidate how their planners worked. It was popular because, like matrix multiplication, it was representative of an important class of problems, those involving interactions among conjunctive subgoals, and it was very easy to describe.

A benchmark is illuminating to the degree that it tells us something we want to know about the behavior of a program. Our goals as scientists, engineers, and consumers dictate what we want to know. Sometimes we are most interested in the system's raw performance: in buying a workstation we may be impressed with the rate at which a particular machine performs matrix multiplication. Likewise, as the potential user of an AI search algorithm we may be impressed with the performance of the *min-conflicts heuristic* algorithm on the Million Queens problem [Minton *et al.* 1990]. As scientists and engineers, however, our interests are different. In these roles, we want to understand why a system behaves the way it does. What is it about the Cray architecture that allows high-performance matrix multiplication? Why does the min-conflicts heuristic algorithms solve increasingly difficult N Queens problems in roughly constant time?

Understanding a system's behavior on a benchmark task requires a model of that task, so our goals as scientists and engineers will often be served only by benchmark tasks that we understand well enough to model precisely. This is especially true for cases in which we expect a program to "pass" the benchmark test. Without a model of the task it is difficult to see what has been accomplished: we risk finding ourselves in the position of knowing simply that our system produced the successful behavior—passing the benchmark.

Models are also important when we design benchmarks to be failed, but in this case we need a model of the factors that make the benchmark difficult. For example, we learn more about articulation by asking a human to say "black back brake block" repeatedly than we do from the equally unpronounceable sentence "alckb bcak raebk lbcko." Both are extremely

difficult but the former is more illuminating because we have models of phonetics that explain why it is difficult. Experiments can tell us which design choices lead to good performance on benchmark tasks, but we need good models of these tasks to explain why this is so. On the other hand, building a good model tends to require a simple problem, and there is always the danger that a simple problem will not be especially illuminating.

Benchmarks ideally are problems that are both amenable to precise analysis and representative of a more complex and sophisticated reality. Unfortunately, the current state of the field often elevates these problems to a new status: they become interesting for their own sake rather than as an aid in understanding a system's behavior on larger, more interesting tasks. Cohen's [1991] survey of papers in the 1990 found that 63% of the papers focused on benchmark problems such as N Queens, the Yale Shooting Problem, and Sussman's Anomaly. Yet very few of these papers made explicit the connection between the benchmark problems and any other task. Without this additional analysis it is difficult to say whether these problems are representative of others we presumably care about, and therefore exactly why the reported solutions are themselves interesting.

As AI begins to focus less on component technologies and more on complete, integrated systems, these traditional benchmarks may reveal their limitations. For example, although we might use N Queens to test the capability and speed of a constraint-satisfaction algorithm embedded in, say, a factory scheduler, this benchmark will not tell us whether the quality of a schedule is appropriate given time constraints and other goals of the program. However, it is far from obvious that *any* benchmark can be devised for such a case. Benchmarks are problems that everyone can try to solve with their own system, so the definition of a benchmark cannot depend on any system-specific details, nor can the scoring criteria. What a researcher learns about a system from performance on a benchmark is liable to be inversely proportional to the size, complexity and specificity of the system.

Thus the conscientious researcher, intent on evaluating her system, faces an uncomfortable choice. The behaviors of the system's components can be evaluated individually on benchmark tasks, or the system's behaviors—not necessarily those of individual components—can be evaluated by task-specific criteria. On the one hand, the researcher learns, say, that her constraint-satisfaction algorithm is extremely slow and won't scale up; on the other, she learns that her system nonetheless produces robust, timely schedules for the particular job-shop she has modeled. Neither result is likely to evoke interest outside the researcher's own laboratory. Why should the rest of us care that an inefficient algorithm

suffices to solve an applied problem that doesn't concern us? The difficulty is that as our attention turns to integrated programs, benchmark scores for component processes might be at variance with, or predict poorly, task-specific measures.

The potential mismatch between benchmark scores and performance on real tasks is also a concern for researchers who are developing testbeds. While some testbeds are no more than an interface to specify parameters of a benchmark problem and instrumentation to measure performance, those described in this article provide rich environments that present a wide range of challenges to planners and related AI programs. You can design a lot of tasks for your planning system in Tileworld, Phoenix, and the other testbeds discussed below. You can study a lot of phenomena—real-time satisficing, graceful degradation under resource restrictions, path planning and navigation, sensor fusion, various kinds of learning, and so on. But each of these general behaviors will be implemented in a particular way depending on the specific testbed and system being developed. "Graceful degradation" in a simplified Tileworld agent might have little in common with what we call graceful degradation in a complex system deployed to perform a real task; just as "aggressive behavior" in seagulls has little in common with aggressive behavior in teenage boys. McDermott's [1981] wishful mnemonic problem has not gone away: two testbed researchers might each claim to have achieved graceful degradation under resource restrictions, but it is more accurate to say that each has achieved something that he or she *calls* graceful degradation. Testbeds make it easier to build programs that exhibit diverse behaviors, but researchers have to face the problem of understanding what like-named behaviors have in common.

Benchmarks and testbeds do not currently bridge the gap between general and specific problems and solutions. A gap exists between the benchmark N Queens problem and another, domain-specific problem that you care about. A gap exists between the testbed problem of having too few bulldozers to fight fires in the Phoenix simulation and a general resource-limited planning problem. Those of us who build and work with testbeds appreciate the opportunities they provide to study many phenomena, but we also recognize the difficulties involved in finding testbed-specific problems that satisfy the criteria of benchmarks: that are simultaneously representative of larger, more interesting problems, easy to describe, and illuminating.

3 Current Issues in Agent Design

Despite the difficulties in designing testbeds, and perhaps because of the promise associated with testbed-based experimentation, a number of testbed systems for studying agent design have been developed to date. Section 4 surveys some of them. This section motivates the survey by describing some significant research issues in agent design and noting corresponding features that testbeds should exhibit. Much current research in agent design builds on the the classical planning paradigm that characterized the field for several years, so our section begins with a short explanation of that paradigm.

The classical planning paradigm assumes an environment that is both *controlled* and *simple*. The planning agent is generally assumed to have complete control over the environment, which means that its intended actions are the only events that can change the world's state and furthermore that the effects of its actions are fully known, both to the agent and to the system designer. The agent is usually assumed to possess complete and error-free information about the state of the world when it begins planning. Since it knows the initial state of the world, knows what actions it intends to carry out, and knows what the effects of those actions will be, it can at least in principle predict exactly what the state of the world will be when it finishes acting. In other words, it knows ahead of time whether a particular plan will or will not achieve its goal.

Classical planners embody strong simplifying assumptions both in the sense that their capabilities (the class of problems they can solve) tend to be quite limited, and also in the sense that the worlds in which they operate tend to be small—exhibiting few features and a limited physics. Planners are generally tested in domains with few planning operators, on goals with few conjuncts, and on models of the world in which few features are explicitly modeled. Performance tends to degrade when the number of operators, goal conjuncts or environmental features increase. Just as control means that the planner can *in principle* prove that its plan will work, the simplifying assumptions mean that the planner can *as a practical matter* generate that proof. Control and simplifying assumptions therefore allow the planner the luxury of generating provably correct plans prior to execution time.

Most current work on agent architectures aims toward relaxing these assumptions. Reactive systems, for example, deal with the problem that the world can change unpredictably between plan time and execution time by deciding what to do at execution time instead of generating a plan prior to execution. Case-based planners confront the simplicity problem

by storing only the essential details of a solution, allowing the planner to concentrate on the relevant features of a new problem.

Below we describe some specific issues that have recently attracted the attention of planning researchers, and therefore guide decisions about what features a planning testbed might exhibit.

Exogenous Events Perhaps the most limiting assumption of the classical planning worlds (most notably the Blocksworld) is that no exogenous, or unplanned, events can occur. Relaxing this assumption makes the process of predicting the effects of plans more difficult [Hanks 1990b] and also introduces the need to react to unplanned events as they occur at execution time [Agre and Chapman 1987], [Firby 1989]. The *time cost* of planning becomes important in a world that allows unplanned changes: the longer the agent takes to plan, the more likely it is that the world has changed significantly between the time the plan was generated and the time it is executed [Bratman *et al.* 1988], [Russell and Wefald 1991], [Dean and Boddy 1988].

Complexity of the World A realistic world has many features. Even a simple block has color, mass, texture, smell, and so on, although many of these features will be irrelevant to many tasks. A realistic world also has a complex causal structure: changes in one aspect of the world may change many other aspects, even though most of those changes may again be irrelevant to any particular problem. Reasoning about more realistic models of the world requires the ability to represent and make predictions about complex mechanisms [Weld and de Kleer 1989], as well as the ability to recognize and focus attention on those aspects of the world relevant to the problem at hand [Hanks 1990a]. A testbed for exploring realistically complex planning problems should itself provide a complexity and diversity of features.

Quality and Cost of Sensing and Effecting Sensing and effecting, generally ignored by the classical planners, are neither perfect nor cost free. An agent must therefore incorporate incorrect and noisy sensor reports into its predictive model of the world [Hanks and McDermott 1993] and must plan sensing actions to improve its state of information, taking into account both the benefit of that information and the cost of acquiring it [Chrisman and Simmons 1991]. Thus a testbed for studying agent design might be populated with agents having imperfect sensors and effectors. The testbed needs

to make a clean distinction between the agent and the simulated world, the agent's sensing and effecting capabilities defining the interface.

Measures of Plan Quality Classical planners are provided with a goal state to achieve, and they stop when their plans can achieve that state. But simple achievement of a goal state is an inadequate measure of success: it does not take into account the cost of achieving the goal, and it also does not admit the possibility of partial goal satisfaction. [Haddawy and Hanks 1993] and [Wellman and Doyle 1991] explore the relationship between goal expressions and utility functions. A testbed for exploring richer notions of success and failure should allow the designer to pose problems involving partial satisfaction of desired states, forcing the planner to trade the benefits of achieving the goal against the cost of achieving it. The problem of balancing cost against solution quality becomes more difficult when the agent is actually planning for a *sequence* of problems over time, some of which may not even have been made explicit when it begins to plan.

Multiple Agents Allowing multiple agents to act in the world introduces new problems: how to coordinate behaviors, how the agents should communicate, how the effects of simultaneous actions differ from the effects of those actions performed serially. Multiple-agent planning is an active research area [Bond and Gasser 1988] and a testbed for exploring these research issues must allow coordinated behavior and communication among the agents that inhabit it.

In addition to the functionality required to make the testbed challenging, we will also identify some design issues that will tend to make a testbed more useful to prospective users:

A Clean Interface It is important to maintain a clear distinction between the agent and the world in which the agent is operating. The natural separation is through the agent's sensors and effectors, so that interface should be clean, well defined, and well documented. A designer must be able to determine easily what actions are available to the agent, how the actions are executed by the testbed, and how information about the world is communicated back to the agent.

A Well Defined Model of Time Testbeds must present a reasonable model of passing time in order to simulate exogenous events and simultaneous action, and to define

clearly the time cost of reasoning and acting. (This is a general problem in simulation and modeling. See [Law and Kelton 1981], for example.) On the other hand the testbed must *somehow* be able to communicate how much "simulated time" has elapsed. Making sense of experimental results requires a way to reconcile the testbed's measure of time with that used by the agent.

Supporting experimentation Testing an agent architecture amounts to assessing its performance over a variety of sample problems and conditions. Controlled experiments require problems and environmental conditions be varied in a controlled fashion. A testbed should therefore provide a convenient way for the experimenter to vary the behavior of the worlds in which the agent is to be tested.

The experimenter must also be able to monitor the agent's behavior in the testbed world [Langley and Drummond 1990]. While it is far from clear at this point what statistics *should* be used in such an assessment, the testbed must allow performance statistics to be gathered. It is also useful for the data to be formatted automatically for analysis using statistical software packages.

4 Testbed Implementations

Previous sections provided the motivations for simulated testbed worlds and discussed some of kinds of problems that might be explored in them. This section surveys several of the simulated worlds available to the community. Our survey is not exhaustive nor is our selection of testbeds meant to imply that they are the "best" available. For each testbed, we describe the sort of world the testbed is supposed to simulate and the research problems it was designed to test; we discuss the interface between the agent and the world and that between the researcher and the system (agent plus world); and we summarize the main methodological commitments associated with the testbed.

4.1 Grid worlds

Several testbed worlds have been organized around the theme that the agent is situated in a rectangular two-dimensional grid and its main task is to push tiles around the grid. We will first discuss the Tileworld of Pollack and Ringuette [1990], then the independently developed NASA Tileworld (NTW) [Philips and Bresina 1991], and the MICE simulator [Montgomery *et al.* 1992]

[Pollack and Ringuette 1990] reports on the Tileworld testbed—a system designed to support controlled experiments with agent architectures situated in dynamic and unpredictable environments. The world consists of a rectangular grid, on which can be placed the *agent*, some *tiles*, some *obstacles*, and some *holes*. Each object occupies one cell of the grid. The agent can move up, down, left, and right, unless doing so would cause it to run into the world's boundaries or an obstacle. When a tile is in a cell adjacent to the agent, the agent can push the tile by moving in its direction. The agent's goal is to fill holes with tiles. Each hole has a *capacity* C and a *score* S. When the agent pushes C tiles into a hole, that hole disappears and the trial's score increases by S. Each trial has a time limit, and the agent's performance is measured by the trial's score at its completion.[1]

The Tileworld environment includes exogenous events: objects in the world can appear and disappear during a simulation. The experimenter can control the rate at which these objects appear and disappear, as well as certain characteristics (capacity and score) of the newly created objects. The ability to control these parameters is an important feature of the Tileworld, because it allows systematic exploration of worlds with various characteristics (e.g., worlds that change relatively quickly or slowly). The goal of such exploration is to find systematic relationships between world characteristics and corresponding characteristics of the embedded agent. The Tileworld system is distributed with a basic agent design, which is also parameterized to allow manipulation by the experimenter (see the discussion below).

The interface between the agent and the world allows the agent, at any time, to take one of four primitive actions: move left, right, up, and down. Some or all of the primitive actions may be infeasible at a given time; for example, if an obstacle is blocking the way. The effects of each action are predetermined and deterministic: the agent will always move to the appropriate adjacent cell if it chooses to do so and the move is feasible. It will never end up in a different cell "by accident." Tiles and obstacles are characterized by their types and by their location on the grid. Each takes up exactly one cell. Holes, which may occupy one or more cells, are characterized by location, capacity, and score.

Holes, obstacles, and tiles appear and disappear probabilistically, according to parameter settings established by the researcher prior to any trial. The probabilities are independent of one another: a single probability governs the appearance of tiles, and it is the same regardless

[1]The scoring metric in Tileworld was later revised to make it easier to compare trials of varying length: raw score is replaced with a normalized value called efficiency [Kinny and Georgeff 1991]. A number of changes have been made to the Tileworld system since 1990, some of which are discussed in Section 5.2; see also [Pollack *et al.* 1993]. Code and documenation for the Tileworld is available by sending mail to `tileworld-request@cs.pitt.edu`.

of the time, of the location, or of any other parameter in the game.

The Tileworld has no explicit sensing operators. The agent is provided with a data structure that describes the world's state in complete detail and with complete accuracy. The use of this information is left up to the designer of the embedded agent; for example, he or she can design mechanisms that distort the information to introduce inaccuracy.

The researcher describes a world by specifying the size of the grid, the duration of the game, and the probability parameters governing the appearance and disappearance rates of tiles, obstacles, and holes, and the distribution of hole scores and capacities. The experimenter can control additional environmental characteristics: for example, the experimenter can decide whether hole scores remain constant until the hole disappears or whether the score decreases over time. To facilitate experimentation, the system provides mechanisms for specifying suites of experiments, which can then be run without intervention, and for recording performance data.

Three related qualities characterize the Tileworld: its abstract nature, its simplicity, and its "parameterizability." The Tileworld is not an attempt to model any particular planning domain; instead the world might be used to pose paradigmatic planning problems in the abstract. It is a very simple world which presents the agent with only a few possibilities for action; objects have few attributes, and the occurrence and effects of exogenous events are not complex. The world's simplicity means that a few parameters define a world instance completely, and these parameters can be varied as experiments are performed.

The Tileworld was originally developed to investigate a particular agent architecture (IRMA [Bratman *et al.* 1988]), and, in fact, is distributed to the research community with an embedded IRMA agent. IRMA actually specifies a space of agent architectures; in other words, there is a range of agent architectures within the IRMA framework. The embedded Tileworld agent is parameterized to allow exploration of the design choices consistent with the IRMA specifications.

The interface between a Tileworld agent and its environment works as follows: when the agent wants to perform some action, it calls the simulator as a subroutine, specifying the action it wants to perform, along with an indication of the amount of time that elapsed since its last call (representing the amount of time it spent reasoning about what to do). The simulator then updates the world, both to reflect exogenous events that took place during that period and to reflect the agent's new actions. The resulting world is then passed back to the agent (in a data structure called the *world*).

This approach to agent/environment interface places the responsibility for specifying sensing and effecting conditions on the agent designer. If the agent uses the *world* data structure directly, it will always have a complete and correct model. Incomplete or noisy sensing can be achieved by manipulating this data before the agent is allowed to use it. Similarly, imprecision in effecting change has to be specified within the agent itself.

The NASA Tileworld NTW [Philips and Bresina 1991], [Philips *et al.* 1991] is an independently developed testbed also organized around the theme of a two-dimensional grid with tiles. Exogenous events in the NTW consist of *winds* that can blow tiles across the grid. NTW has no obstacles or holes.

Two features distinguish the two simulators. First, the NTW simulator has no built-in measure of success analogous to the notion of a score. What the agent is supposed to do, and what constitutes success, is left entirely to the experimenter. The second is the nature of the interface between the agent and its environment: the Tileworld agent called the simulator as a subroutine and passed information back and forth using a shared data structure. The NTW agent and the world simulator run asynchronously: the agent posts commands to the world, which are put on a queue and eventually executed. Operators can be programmed to "fail" probabilistically: a grasp operation might not result in the agent holding the tile, a move might result in the agent being displaced to an adjacent location other than the one intended. The agent is given no indication of whether an operator has succeeded or failed, and must explicitly sense the world in order to ascertain the effects of its actions.

MICE [Montgomery and Durfee 1990], [Montgomery *et al.* 1992] is another grid-oriented simulator, designed to support research into coordinating the problem-solving behavior of multiple autonomous agents. The basic layout of MICE consists only of a grid and various agents, though agents can be used to simulate objects like tiles and forest fires.

The basic MICE operator is `MOVE`, moving the agent from one grid cell to an adjacent cell. The `LINK` command is an abstract version of a "grasp" operator: the agent uses it to pick up objects. The world is populated only with agents, but they can be quite diverse. MICE has no explicit provision for exogenous events, though they can be simulated to some extent by implementing agents that have the desired effects on the world (making a grid cell wet and slippery to simulate rain, for example)

The main difference between MICE and the Tileworlds is that MICE makes even less of a commitment to a "world physics"— the experimenter defines an agent's sensing and effecting capabilities, and also the effect of actions taken simultaneously by the agents. MICE might

be viewed more as a *framework* for building testbeds rather than a simulator in and of itself. (The MICE designers have built versions of Tileworld and of Phoenix using this platform. See [Montgomery and Durfee 1990], for example.)

4.2 The Phoenix testbed

Phoenix [Hart and Cohen 1990] [Greenberg and Westbrook 1990] is a framework for implementing and testing multiple autonomous agents in a complex environment. The scenario is fire fighting: the world consists of a map with varying terrain, elevations, and weather. Fires can start at any location and will spread depending on the surrounding terrain. Agents are fire-fighting units (commonly bulldozers) that change the terrain to control the fires.

It is helpful to distinguish the Phoenix simulator from the Phoenix environment and Phoenix agents. The simulator has three main functions: to maintain and update the map, to synchronize the activities of the environment and the agents, which are implemented as independent *tasks*, and to gather data. The Phoenix environment includes a representation of Yellowstone National Park (from Defense Mapping Agency data) and the tasks that implement fires. Phoenix agents generate tasks that simulate a fireboss, several bulldozers, watchtowers, helicopters, fuel tankers and so on. Agent tasks include moving across the map, cutting fireline, predicting the course of fires, planning how several bulldozers should attack fires, monitoring progress and detecting failures in expectations, and failure recovery. Tasks insert themselves (by sending messages) onto a *timeline* maintained by the Phoenix simulation. Tasks run intermittently and sometimes periodically.

Phoenix agents sense and change the Phoenix environment by sending messages to the object managing the map, but the simulator makes no attempt to control the form of the messages. Thus Phoenix agents have no predetermined set of operators. The Phoenix environment contains only two kinds of objects, agents and fires. However, each cell of the map of the environment contains information that agents and fires use to determine their behavior. For example, bulldozers travel more quickly on cells that are designated "blacktop road" and fires burn more quickly in the direction designated "uphill." Exogenous events are also implemented as tasks, and influence other tasks indirectly. For example, wind causes fires to burn more quickly.

Tasks make their effects known by sending messages to the simulator. The form of these messages is not restricted: any task may in principle find out anything about the world and effect any change. The simulator enforces no model of sensing. It provides information about

the world (the characteristics of a cell in the map, for example) by responding to messages, but does not restrict its answers. However, the Phoenix agents have limited sensory and physical abilities; for example, bulldozers have a 200 meter radius of view (although the view is not affected by elevation) and they move and cut fireline at rates codified by the U.S. Forestry Service.

Defining an environment consists of defining a map—the topographical features for a land area, including ground cover, elevation, roads, rivers, and buildings—and processes within the environment such as fires and wind. Defining an agent is generally more complicated because it involves designing sensors, effectors, a planner, a reactive component, internal maps of the environment, and so on.

Phoenix includes an experiment-running facility that includes a language for specifying scripts for changes in weather, fires starting, and other events. It also allows for agents' behavior to be monitored, producing data files that can be read by data-manipulation and statistical packages. The design of the Phoenix system is modular and other testbeds have been developed rapidly by swapping out the Yellowstone map and the Phoenix agent definitions, and swapping in, for instance, a world of shipping lanes, ports, docks, ships and roads.

Phoenix differs from the previous simulators in that tries to provide a realistic simulation of a single domain rather than implement an abstract domain-independent task environment. Apart from that difference, however, it is quite similar to the MICE simulator in that it enforces few constraints on how agents and exogenous events can sense or change the world. The simulator maintains the map and schedules activities, but, like MICE, much of the domain's physics lies in definitions of the individual tasks.

4.3 The Truckworld

The Truckworld [Firby and Hanks 1987] [Hanks *et al.* 1992] is a multiagent testbed designed to test theories of reactive execution [Firby 1989] and to provide motivating examples for a theory of reasoning about dynamic and uncertain worlds [Hanks and McDermott 1993], [Hanks and McDermott 1992]. The main commitment is to provide a realistic world for its agents, but without physical sensors or effectors[2].

An agent is a truck consisting of two arms, two cargo bays, several sensors, and various

[2]Truckworld code and documentation is available by sending mail to `truckworld-users-request@cs.washington.edu`

other components like a fuel tank, a set of tires, and direction and speed controllers. It operates in a world consisting of *roads* and *locations*. Roads connect the locations, which are populated with *objects*. The simulator itself places few restrictions on the behavior of objects, which can be quite complex. The Truckworld can model objects like fuel drums, which the truck can use to increase its fuel level, tire chains, which help it drive safely down slippery roads, vending machines, which require money and produce a product, and bombs, which tend to break unprotected objects in their immediate vicinity.

Exogenous events like rainstorms occur periodically in the world. A rainstorm makes all roads in its vicinity wet, and dirt roads become muddy for a while. The truck runs the risk of getting stuck in the mud if it travels on a muddy road without proper tires. Objects in the vicinity of a rainstorm get wet too, and that might affect their behavior (a match might not ignite any more, a plant might start growing). The occurrence of events can depend both on random chance and on characteristics of the world (rainstorms might be more likely at certain locations or at certain times of day).

The Truckworld provides a wide variety of (simulated) sensors: cameras report visual features of objects, sonars report whether there is an object at a location, scales report an object's weight, and X-ray machines report on objects within a closed container. Sensors typically have noise parameters: a camera sometimes reports an incorrect but "close" color for an object, and this is more likely at night than during the day. A scale reports the object's true weight distorted according to a user-supplied noise distribution; a sonar occasionally incorrectly reports that an object is present.

A variety of communication devices are available: radios allow connection among agents; loudspeakers produce sounds that can be detected by microphones in the vicinity. Motion detectors notice when objects appear or disappear from their immediate vicinity. Tape recorders are activated when a sound is produced, and an agent can retrieve the recorded message later.

Communication between an agent and the simulator is tightly controlled: each agent and the simulator itself run as separate processes, communicating over two channels. The agent performs actions and gets sensor reports over the *command channel*, and uses the *control channel* to manipulate the simulator's internal state (e.g. to connect or disconnect from the simulator, to advance the simulator's clock, or to collect statistics about the world). Multiple agents communicate using only the communication devices the world provides for them.

We had two main goals in designing the Truckworld: (1) to provide a testbed that gen-

erates interesting problems both in deliberative and reactive reasoning, but without committing to a particular problem domain, and (2) to provide significant constraints on the agent's effecting and sensing capabilities and on the causal structure of the world, but still allow the system to be extended to meet the designer's needs.

The Truckworld occupies a position between simple abstract simulators like the Tileworlds, a domain-specific simulator like Phoenix, and a testbed-building platform like MICE. The Truckworld implements a specific set of operators for the agent (unlike MICE), but provides fewer constraints than do Tileworld or Phoenix on the nature of the other objects in the world and on the interaction between the agent and those objects.

4.4 Summary

We have looked at five systems for implementing planning testbeds: the parameterizable Tileworlds, the multi-agent MICE platform, the Phoenix firefighting simulation, and the Truckworld simulator. Although there are many differences as to what features each system offers and what design decisions each makes, we can identify three main areas in which the systems differ:

- Domain dependence
  Phoenix strives for a realistic depiction of a single domain, the Tileworlds and MICE try to describe abstract worlds and operators that affect the world. There is an obvious tradeoff in this decision: a researcher using a domain-dependent simulator may be able to demonstrate that her program is an effective problem solver in that domain, but may have difficulty going on to conclude that the architecture is effective for dealing with other domains. A researcher using an abstract simulator may be able to build a system based on what he or she judges to be "general problem-solving principles," but then the difficulty is in establishing that those principles apply to any realistic domain.

- Definition of sensors and effectors
  The question arises as to whether or to what extent the simulator should define the agent's sensing and effecting capabilities. On one extreme we have the Phoenix simulation, which does not itself impose any constraints on environment dynamics or what agents can find out about their environment. All such constraints are specified in the agent definitions and are merely enforced by the simulator. MICE and NTW represent the other extreme: the simulator defines an agent as well as a world physics, supplying

a set of sensing and effecting operations as part of the world. Truckworld partially defines the truck's effecting capabilities: it defines a set of primitive commands, but the exact effect of these commands depends on the objects being manipulated. Objects and their interactions are defined by the experimenter. The Truckworld does not, on the other hand, define a set of sensing operations. Sensors are objects defined by the experimenter that happen to send sensory information back over the command channel.

- Parameterizability
  The Tileworlds have a built-in set of parameters that characterize the behavior of a world. These parameters facilitate experimentation: by varying the world's parameters systematically and matching them against various agent designs one may be able to come up with "agent types" that perform well for particular "world types." The price one pays for this ability to perform experiments is in simplicity and in control. A world that is fully characterized by a small number of parameters must be simple, and furthermore the parameters must characterize completely the nature of the agent's behavior in that world. Phoenix allows the experimenter to specify values for parameters such as wind speed, and also to write scripts for how parameters change over time during an experiment. Phoenix also provides a mechanism called "alligator clips" for recording the values of parameters during experiments.

  So we are again faced with a tradeoff: in the Tileworlds and Phoenix one may be able to demonstrate a systematic relationship between a world's characteristics and an agent's performance. Such demonstrations, however, must be supplemented with convincing arguments that these will be valid in a more realistic world—and it is far from easy to make such arguments. In the Truckworld one can demonstrate that the agent performs well on more complex problems, but it may be difficult to demonstrate precisely the reasons for that success and to apply those reasons to other domains.

5 Discussion

The discussion to this point has touched on mainly uncontroversial points: the need for introducing more rigorous empirical methods into planning research and the roles that testbed environments and benchmark tasks might play. The question of what the ultimate goal of these research efforts is, and how it might best be pursued, is the subject of some dis-

agreement among the authors. The following three sections reflect this disagreement, and represent the authors' personal opinions. In the first Hanks argues against a program of controlled experimentation in small, artificially simple worlds. Pollack defends such a program in the second section. In the third Cohen addresses the problem of generalizing results from testbed experiments.

5.1 The danger of "experimentation in the small" (Steve Hanks)

The planning community has been pushed (or has pushed itself) in two directions recently, and these directions seem at odds. We see pressure to apply our representations and algorithms to more realistic domains, and at the same time we feel the need to evaluate our systems more rigorously than by announcing a system's ability to solve a few small, carefully chosen problems. The problem is that programs that operate in more realistic domains tend to be bigger and more complicated, and big complicated programs are more difficult to understand and evaluate.

In writing this paper we agreed on the following two points: (1) that our primary objectives as researchers in planning are to build systems that extend the functionality of existing systems—that solve larger or more complicated problems or solve existing problems better; and, (2) to understand how and why these systems work, and further that running experiments is a good way (though not the only way) to accomplish the goal of understanding the systems we build. We tended to disagree, however, on the best way to achieve these objectives, and in particular on the issue of what form an experimental methodology should take and what role it should play in the system-building process.

This section will discuss a particular methodological approach, which I will call "experimentation in the small." [Langley and Drummond 1990] advocate this position in the abstract. [Pollack and Ringuette 1990] and [Kinny and Georgeff 1991] explore it concretely using an implemented testbed and suite of experiments. I take the methodological commitments of this approach to be the following:

1. The researcher conducts her experiments in a testbed world significantly simpler than the world in which the agent is ultimately to be deployed. In particular the world is supposed to exhibit *particular* interesting characteristics, but will be artificially simple in other aspects.

2. The testbed provides a set of parameters that govern the world's behavior. Experimen-

tation is a process of matching characteristics of the agent's problem-solving methods with the world's parameter values; the goal of experimentation is to discover relationships between these two sets of characteristics that predict good (or bad) performance.

The main point of this section will be that experimentation in small, controlled worlds is not, *in and of itself*, an effective way to establish meaningful relationships between agents and their environments. We will see below that the nature of the relationships established by these experiments is inherently connected with the implementation details both of the agent and of the testbed worlds. The hard part remains: generalizing beyond the particulars of the world or even arguing that a particular testbed world is *appropriate* for studying a particular agent architecture. The experiments themselves do not provide guidance in this task, and may even tend to hinder it.

I will use the Tileworld testbed and experiments from [Pollack and Ringuette 1990] and [Kinny and Georgeff 1991] to make these points. My goal in doing so is not to single out this particular work for criticism. I do so first because it's important to discuss the *concrete* results that can be expected from these experimental endeavors, and these two pieces of work are rare examples of systematic experimentation with agent architectures in small, controlled worlds.

5.1.1 The original tileworld experiments

The planning agent studied in [Pollack and Ringuette 1990] is an implementation of the IRMA architecture [Bratman *et al.* 1988]. One of the key ideas advanced in that paper is that one way for agents to cope with a changing environment is to filter out (avoid considering) options that conflict with their current intentions (this is the "filtering" mechanism) unless the option is especially promising (this is the "filter override" mechanism). IRMA also suggests that the agent separate what to do ("deliberation") from how to do it ("planning"). The Tileworld agent thus chooses its actions in two phases: the deliberation phase chooses a hole to fill (we will call this the "current hole"), then the planning phase plans a sequence of moves that will fill the current hole. The agent's filter/override mechanisms govern when the agent will abandon the current hole in favor of a new alternative.

The Tileworld agent has three components:

1. The *filter-override* mechanism—a test applied to a newly appeared hole that determines whether the task of filling the current hole should be reconsidered in light of the new

option(s)[3]. Only one filter-override mechanism was implemented: a threshold v such that a new hole would be considered as an alternative to the current hole just in case its score exceeded the score of the current hole by at least v points.

2. The *deliberator*—a procedure that chooses the next hole to work on. Two alternatives were implemented. The simpler (highest score or HS) deliberator always chooses the hole with the highest score. The more complicated (likely value or LV) deliberator divides the hole's score by an estimate of the cost of filling it: the sum of the distances of the n closest tiles, where n is the hole's capacity.

3. The *path planner*—given a hole to fill, the path planner uses breadth-first search to generate the *optimal* sequence of moves to fill it with tiles. The choice of a path planner was not among the agent parameters varied experimentally—only the optimal path planner was implemented.

The experiments show the following results:

1. An agent that acts "in parallel with reasoning" performs slightly better than an agent that "acts and reasons serially."

2. The more sophisticated LV deliberator (the one that ranks a hole by dividing its score by the sum of the hole's distance to the n closest tiles) performs somewhat better than the simpler HS deliberator (that ranks a hole on the basis of its score alone).

3. The filter-override mechanism at best has no effect on the agent's performance, and in some cases makes it perform worse.

[Hanks and Badr 1991] analyzes these experiments in detail; here I want to discuss some issues relevant to the question of what this experimental paradigm can be expected to accomplish. In particular I want to stress the need for caution in interpreting these results, since there is a large gap between the effort's larger goal, of establishing general relationships between agent designs and environmental conditions, and what is actually presented in the paper. I don't see this as a fault of the paper—which presents preliminary work—but it is important to keep the results in perspective.

[3]The filtering mechanism in the original Tileworld agent is trivial: when the agent is working on filling a hole, the filter rejects all other holes; when the agent does not have a current hole the filter accepts all holes.

The connection between a general architecture for problem solving (in this case IRMA) and the particular results reported must be interpreted taking into account many design and implementation decisions, among them:

1. the way in which the IRMA architecture was realized in the Tileworld agent (e.g. equating "deliberation" with choosing which hole to fill, "planning" with generating a sequence of moves to fill that hole)

2. the implementation of these modules in the Tileworld agent (e.g. the specific algorithms for deliberation and path planning, and how they interact)

3. the implementation of the Tileworld simulator (e.g. the choice of what enviornmental parameters can be varied, the interaction among the different parameters and between the agent and the simulator, and the simplifying assumptions built into the world itself).

Consider the first result, for example, and what broader conclusions we might be able to draw from it. The Tileworld uses a simulated notion of "serial" and "parallel" reasoning. In fact the act cycle and reasoning cycle run sequentially, but they are constrained to take the same amount of time. Is this implementation detail important to assess "the benefit of acting in parallel with reasoning?" I'm not sure. In the current implementation the agent cannot be interrupted in the middle of its reasoning cycle by changes to the world that occur during the concurrent act cycle, which strikes me as a significant deviation from truly parallel reasoning and acting. In any event the speedup result must be interpreted with an understanding of the particular implementation, and cannot be interpreted more broadly without further analysis.

The second result, suggesting that the LV deliberator performs better than the HS deliberator, must also be interpreted in the context of the particular implementation. We note in [Hanks and Badr 1991] that one part of the Tileworld agent is the path-planning algorithm that (1) solves the problem optimally, (2) is not subject to experimental variation, and (3) is written in C, presumably for efficiency reasons. To what extent do the experimental results depend on the ability to solve the path-planning subproblem quickly and optimally? [Hanks and Badr 1991] shows that the fast path planner has *much* more of an effect on the system's performance than does variation in the deliberator (which was one of the parameters varied experimentally). Given this fact we should be cautious about interpreting the experimental

result too broadly: would an agent actually benefit from a more sophisticated deliberator if it were unable to solve the path-planning subproblem quickly and optimally? That question would have to be answered in order to apply the result beyond the specific implementation and experimental setting examined.

The final result, that the filter-override mechanism does not generally improve the agent's performance, strikes me as the one most closely related to the specific agent and environment implementations. The authors recognize the problem that the environment did not "challenge" the deliberator, thus rendering a fast preliminary filtering mechanism unnecessary. They propose making the environment more challenging, specifically by making the world change more quickly (i.e. by changing the parameters that govern the world's behavior).

Another interpretation of the same result is that the Tileworld is *inherently* not a good test of an IRMA-like filtering mechanism. The justification for a filtering and override mechanism is that the filter/override benefits the problem solver when deliberation is complex and difficult, but at the same time when deliberation at least potentially benefits the planner significantly.

Put another way, deliberation is really a matter of predicting the future state of the world and choosing one's actions so as to maximize utility given the predicted future. The problem with Tileworld is that there is very little to predict. Tiles appear and disappear randomly, and with no pattern. The effects of the agent's actions are localized. On balance there is little to be gained from thinking hard about the world, which we demonstrate in [Hanks and Badr 1991] by demonstrating that there is little benefit to be had even by implementing a deliberator that computes the agent's *optimal* course of action given present information. If deliberation is either easy to do or doesn't benefit the agent significantly, then there is no need for a surrogate for deliberation like the filter/override. The authors mention the possibility of making the deliberation process more expensive, but not the possibility of changing the world (e.g. by giving it more causal structure or by making the agent's reward structure more complex) so as to give more potential payoff to the deliberation process.

The point of this discussion is to demonstrate the difficulty of interpreting experimental results such as those reported in [Pollack and Ringuette 1990], or more specifically the difficulty associated with applying the results to *any* circumstances other than those under which the experiments were conducted. Below I will discuss the implications to the general paradigm of experimentation in the small, but first I want to discuss some follow-up experiments in the Tileworld environment.

5.1.2 Subsequent tileworld experiments

The experiments in [Pollack and Ringuette 1990] tried to establish a relationship between the agent's commitment to its current plan—its willingness to abandon its current goal to consider a new option, or "boldness" as it was called—and the rate at which the world changes. [Kinny and Georgeff 1991] try to make this relationship precise and provide additional empirical support. They begin their experimental inquiry by further simplifying the testbed world:

> [The Tileworld] was considered too rich for the investigative experiments we had planned. Therefore, to reduce the complexity of the object-level reasoning required of our agent, we employed a simplified Tileworld with no tiles. [Kinny and Georgeff 1991, p. 83]

The agent's task in this simplified Tileworld is to move itself to a hole on the board, at which point it is awarded the hole's score. The agent is provided with perfect, immediate, and cost-free information about the world's present state.

Once again the planning agent is configured around the tasks of deciding which hole to pursue, decide which path to take to the chosen path, and deciding whether or not to pursue a new hole that appears during execution.

The agent always chooses the hole with the highest ratio of score to distance. It will adopt a *new* hole according to its "filter override" policy, also called its "degree of commitment" or "degree of boldness." Degree of boldness is a number b—the agent will automatically reconsider its choice of hole after executing b steps of its path toward the hole it is currently pursuing. A "bold" agent will therefore tend to make a choice and stick with it. A "cautious" agent will tend to reconsider more often, and will be more likely to abandon its old choice of hole in favor of a new one.

Another agent parameter is its "planning time," a number p set by the experimenter. The path planner produces an *optimal* path to the current hole, and the "planning time" parameter dictates that it took p time units to do so. It is important to point out two things. First of all, the number p bears no necessary relationship to the amount of time that it *actually* took to generate the plan. "Planning time" is a constant set by the experimenter and does not depend on the time it takes to build a plan for the current path. Second, increasing or decreasing p has no effect on solution quality. The authors are *not* exploring the tradeoff between planning time and plan quality. The path planner always returns an

optimal path; the planning-time parameter makes it *seem* like it took p time units to do so.

A single parameter causes variation in the world: γ, which is the ratio of the agent's "clock rate" to the rate at which the world changes. Large values of γ indicate that the world changes frequently relative to the amount of time it takes the agent to act. The agent's "effectiveness" was measured by dividing the number of points the agent actually scored by the sum of the scores for all the holes that appeared during the game.

The experiments showed various relationships between effectiveness, rate of world change, commitment, and planning time:

- Effectiveness decreased as the rate of world change (γ) increased.
- As planning time approached 0, an agent that reconsidered its options (choice of hole) after every step performed better than an agent that never reconsidered.
- As γ increased, an agent that reconsidered often tended to perform better than an agent that reconsidered infrequently, planning time held constant.
- When the cost of planning is high, an agent that reconsidered infrequently tended to perform better than one that did so frequently, rate of world change held constant.

The experiments above used an agent that reconsidered its current hole after a fixed number of steps b. If the agent instead reconsidered its choice of path either after b steps *or* when the target hole disappeared then the bold agent outperformed the cautious agent, regardless of the values of p and γ. Performance was improved further by reconsidering the choice of target when a hole appeared closer to the agent than the current target.[4]

Once again I want to point out the difficulty in applying these results to situations other than the specific experimental environment. Doing so requires evaluating the simplifications to the Tileworld and how they affect the complexity of the deliberation task, evaluating how well the definitions of "boldness" and "planning time" apply to different domains, and so on. To what extent does the last result, for example, depend on the fact that the agent was provided with complete, instantaneous, correct, and cost-free information about changes to the world?

[4]In both experiments the agent was automatically and immediately notified of the appearance and disappearance of holes.

5.1.3 Analysis

How do these experiments advance the cause of building intelligent agents? I think it's clear that the agents presented in these papers do not in and of themselves constitute significant progress. Both operate in extremely simple domains, and the actual planning algorithm consists of using a shallow estimate of a hole's value to focus the agent's attention, then applying an optimal algorithm to plan a path to the chosen hole. This strategy is feasible only because the testbed is so simple: the agent has at most four possible primitive actions, it doesn't have to reason about the indirect effects of its actions, it has complete, perfect, and cost-free information about the world, its goals are all of the same form and do not interact strongly, and so on.

The argument must therefore be advanced that these experimental results will somehow inform or constrain the design of a more interesting agent. Such an argument will ultimately require translating these results into general relationships that apply to significantly different domains and agents, and I pointed out how tricky it will be to establish *any* applicability beyond the experimental testbed itself. A crucial part of this extensibility argument will be that certain aspects of the world—those that the testbed was designed to simulate more or less realistically—can be considered in isolation, that is, that studying certain aspects of the world in isolation can lead to constraints and principles that still apply when the architecture is deployed in a world in which the testbed's simplifying assumptions are relaxed.

The nature of experimental relationships Finding a general and useful interpretation for experimental results is a crucial part of the process of controlled experimentation. One immediately faces the tradeoff between stating the relationships in such a way that is not so general as to be uninformative, but at the same time not so specific that they don't generalize outside the particular agent and world in which the experiments were conducted.

Both Tileworld papers discuss the difference between a "bold" and a "cautious" agent, for example. These general terms are supposed to suggest an agent's willingness to reassess its plan commitments as it executes its plans: a "bold" agent rarely reconsiders its plans, a "cautious" agent does so frequently.[5]

The two main results from [Kinny and Georgeff 1991] can be stated as follows: First, that it's a good policy for an agent to be more cautious as the world changes more rapidly. Or in

[5]The terms can be defined precisely within the IRMA framework—they describe the sensitivity of the agent's filter-override mechanism—but presumably the terms and the associated relationships are intended to be applied to agents other than implementations of IRMA.

other words, planning ahead doesn't do as much good when the world changes a lot before or while the plan is being executed. Second, it's a good policy for an agent to rethink its commitment to a goal when the goal disappears, or when a goal appears that superficially looks more promising than its current goal. Both results turns out to be quite robust, holding as various other parameters both of the agent and of the world are varied. Stated this way the relationships seem pretty straightforward—I would be surprised to hear about an agent that did *not* adopt these policies, either explicitly or implicitly. The question is, therefore, whether the relationships stated in these very general terms provide significant guidance to those that build other agents.

Of course the first relationship can be restated much more specifically, mentioning the agent's goals (to move to holes), its problem-solving strategy (to choose a hole using a heuristic, then plan a path optimally to that hole, using exactly p units of time to do so), its definition of "boldness" (the number of operators it executes before replanning), and the nature of "world change" parameter (the rate at which holes randomly appear). Interpreted in this light the result is much less obvious—it does provides significant guidance to somebody who wants to design an agent using the architecture so described, to act effectively in a world so described, given problems of the sort so described. But nobody really wants to do that. So the problem is how to interpret this more specific relationship in a broader context. What if the agent doesn't have immediate, perfect, and cost-free information about the appearance of holes? What if the designer does not have an optimal and efficient path planner at his disposal? What if the appearance of holes is not truly random, but operates according to some richer causal structure? Do the same relationships still hold; for that matter, are the same relationships even meaningful?

The main point here is that experimentation will not provide us automatically with meaningful relationships between agents and environments. Claiming that a specific experimental relationship establishes a connection between "boldness" and "rate of world change" constitutes a form of wishful thinking[6]. It translates a very specific relationship between a particular implemented agent and a particular simulated world into terms that are very intuitive, very broad, and very imprecise. Giving intuitive names to these characteristics and to their relationship does not make them meaningful or broadly applicable. The real contribution of such an analysis would be to come up with the right way of characterizing the agent, the world, and their relationship—in terms that are not so specific as to be ap-

[6]Cf. [McDermott 1981]

plicable only to the experimental domain but not so vague as to be vacuously true. So far the experimental work has not focused on this question; in fact it's worth asking whether running experiments in artificially small simple worlds is the right place to start looking for these relationships at all.

5.1.4 Examining environmental features in isolation

We turn now to the second assumption underlying experimentation in the small: that a particular characteristic of a realistic world can be studied in isolation, and good solutions to the restricted problem will lead to good solutions in the realistic world. The Tileworld, for example, focuses on unplanned change in the form of random appearance and disappearance of tiles and holes (or just holes in the case of the simplified Tileworld), but simplifies away other aspects of the world.

This "scaling" assumption is absolutely crucial to the whole experimental paradigm, and I have not seen it defended in the literature. In fact the only explicit mention of the assumption I have found appears in [Philips *et al.* 1991]:

> We are not suggesting that studies of these attributes in isolation are sufficient to guarantee the obvious goals of good methodology, brilliant architectures, or first-class results; however, we *are* suggesting that such isolation *facilitates* the achievement of such goals. Working on a real-world problem has obvious benefits, but to understand the systems that we build we must isolate attributes and carry out systematic experimentation.

My own work leads me to believe that it will be quite difficult to isolate particular aspects of a large planning problem. [Hanks 1990b], for example, confronts the problem of reasoning about plans in an uncertain world. Unplanned, random change—like tiles and holes appearing and disappearing—is one source of uncertainty, but there are others: the agent can have incomplete or incorrect information about the world's initial state, an incomplete model of its own actions, and may not have enough to time to consider explicitly every outcome of its plan. I see no way to separate one of these factors from the others in any principled way, therefore I see no way that studying the simplified problem of a world in which all uncertainty is due to unplanned, random change will shed light on the larger problem of reasoning about plans in an uncertain world.

It's not even clear whether the problem that the Tileworld papers claim to be investigating—the decision of when it is advantageous to act as opposed to deliberate—can be considered in a context in which all exogenous change is random. The decision about whether to plan or act depends both on the world and on the agent's ability to predict the world: the better it is at reasoning about the effects of its actions, the more benefit can be derived from thinking ahead.

The Tileworld trivializes the prediction process by making the world essentially unpredictable: tiles and holes appear and disappear randomly, without a regular pattern. The agent therefore has no incentive to reason about what tiles *might* appear or disappear or where they might appear, which greatly simplifies the question of whether it should deliberate or act. Can we therefore apply the experimental results established in the Tileworld to worlds in which prediction is a difficult problem?[7]

Experimentation in the small depends on the ability to study particular aspects of a realistic world in isolation and apply solutions to the small problems to a more realistic world. We have seen no indication that this can in fact be done; in fact neither Tileworld paper argues that random unplanned change is a reasonable feature for isolated study. An experimenter using these worlds therefore runs the risk of solving problems in a way that cannot be extended to more realistic worlds, and at the same time making his job artificially difficult for having studied the problem in isolation. [Kinny and Georgeff 1991, p. 82] states that

> "[Simulated worlds] should ideally capture the essential features of real-world domains while permitting flexible, accurate, and reproducible control of the world's characteristics."

An appealing proposition, but the fact is we don't know what it means to "capture the essential features of real-world domains," much less whether it is possible to do so in a system that allows "reproducible control of the world's characteristics." Conducting experiments in small, controlled worlds carries with it the responsibility of considering the implications of the simplifications that were made to allow the experimentation in the first place.

But at this point we must remind ourselves of our ultimate goals: to build systems that solve interesting problems and to understand why they do so. Research decisions must be oriented toward solving *problems*, not toward satisfying methodological goals. The ultimate

[7] [Chapman 1990] advances an even stronger view: that randomness without structure actually makes planning more difficult.

danger of experimentation in the small is that it entices us into solving problems that we understand rather than ones that are interesting. At best it gives the mistaken impression that we are making progress toward our real goal. At worst, over time it confounds us to the point that we believe that our real goal *is* the solution of the small, controlled problems.

5.1.5 Conclusion

In no way should this section taken to be an argument against using experimental methods to validate theories or programs. In fact I think the need for experimentation is manifest: we need to understand why and how well our ideas and our architectures work, and we will not always be able to do so using analytic methods. Neither am I opposed to conducting these experiments in controlled, overly simplified worlds. I can imagine, for example, a researcher implementing some idea in a system then building a small world that isolates the essence of that idea, then using the small world to explore the idea further. I object, however, when attention turns to the experimentation process itself instead of to the ideas that are to be tested, and when the assumptions inherent in the small world are adopted without regard to the relationships the world is supposed to demonstrate.

The ultimate value—arguably the *only* value—of experimentation is to constrain or otherwise inform the designer of a system that solves interesting problems. In order to do so the experimenter must demonstrate three things:

1. that her results—the relationships she demonstrates between agent characteristics and world characteristics—extend beyond the particular agent, world, and problem specification she studied,

2. that the solution to the problem area she studied in isolation will be applicable when that same problem area is encountered in a larger, more complex world, and

3. that the relationship demonstrated experimentally actually constrains or somehow guides the design of a larger more realistic agent.

The experimental work I have seen has addressed *none* of these questions.

I originally stated our two objectives as researchers as (1) building interesting systems and (2) understanding why they work. It seems to me that experimentation in the small adopts the position that these goals should be tackled in reverse order—that you can understand how an interesting system *must* be built without actually building one. I don't believe that is

the case; rather we should be building systems and *then* applying analytic and experimental tools to understand why they did (or did not) work.

5.2 The promise of experimentation (Martha Pollack)

My co-author (SH) believes that experimentation "in the small" is a dangerous enterprise. I believe that, to the contrary, controlled experimentation—"small", "medium", or "large"—promises to help AI achieve the scientific maturity it has so long sought. In these comments I shall try to defend that belief.

SH begins by stating that the primary objectives of those studying agent design are "(1) to build systems that extend the functionality of existing systems ...and (2) to understand how and why *these* systems work." (p. 18, emphasis mine.) I would put the second point somewhat differently and claim that we aim to understand "how and why such systems *can* work." This is not a minor matter of wording, but rather a fundamental disagreement about research methodology. SH believes that complex-system building must precede experimental analysis, while I believe that these two activities can and should proceed in parallel. SH does not object to all experimentation, but only to experimentation "in the small," i.e., experimentation using simplified systems and environments. I claim not only that such experimentation can be informative, but that, given our current state of knowledge about system design, controlled experimentation often requires such simplifications. Thus, in my view, SH's position is tantamount to an injunction against all experimentation in AI—in other words, it is a call for the maintenance of the *status quo* in AI methodology.

It is important to be clear about what would constitute an understanding of how and why certain autonomous agents work. In my view, this will consist in a theory that explains how alternative design choices affect agent behavior in alternative environments, i.e., it will largely be made up of claims having the form: "A system with some identifiable properties S, when situated in an environment with identifiable properties E, will exhibit behavior with identifiable properties B."[8]

The goal of experimentation in AI (and arguably, a primary goal of the science of AI taken as a whole) is to elucidate the relationships between sets of properties S, E, and B, as defined above. I will argue that for experimentation to succeed in meeting this goal, two

[8]For similar statements of this research paradigm, see [Chrisman *et al.* 1991,Cohen *et al.* 1990,Rosenschein *et al.* 1990,Pollack and Ringuette 1990,Langley and Drummond 1990]. Some researchers also split out the properties of the agent's task; in these comments, I will consider the task specification to be part of the environment, but my argument does not depend on this.

types of simplification must be made. The first is inherent in the very notion of experimental design. Experimentation necessarily involves selective attention to and manipulation of certain characteristics of the phenomena being investigated. Such selectivity and control constitutes a type of "simplification" of the phenomena. The second type of simplification that is currently needed arises from our existing abilities to build complex AI systems. Large, complex systems that tackle interesting problems are generally not principled enough to allow the experimenter to meaningfully probe the design choices underlying them. Moreover, they are designed for environments in which it may be difficult or impossible to isolate, manipulate, and measure particular characteristics. Finally, they do not generally include instrumentation to measure their performance, although it is conceivable that in many cases this could be added in a fairly straightforward way. Thus these systems do not allow the experimenter sufficient access at least to S and E, and possibly also to B; they are, in short, ill-suited for controlled experimentation. In contrast, the kinds of simplified systems we described in Section 4, i.e., testbeds such as the Tileworlds, Truckworld, and Phoenix, along with their embedded agents, are designed specifically to provide the control needed by the experimenter.

SH correctly notes that the simplifications required for experimentation introduce methodological challenges. In particular, he points out the issue of generalizability: how can a researcher guarantee that the simplifications made in the design of an experiment do not invalidate the generality of the results obtained? I believe that this is a serious issue, one that poses a significant challenge to AI researchers. Moreover, I agree with SH that by and large the controlled experimentation that has been performed to date in agent design— including my own work—has not adequately met this challenge. That, however, is due to the fact that so far there has been painfully little controlled experimentation conducted in AI: as SH notes, the Tileworld experiments represent relatively "rare examples of systematic experimentation with agent architectures" (p. 19).[9] It is extremely difficult, and often impossible, to have confidence in the generality of the results obtained from a very few experiments. The desire for robust, generalizable results should lead us to do more, not less, experimentation.

The problem of generalizability is not unique to AI: it is inherent in the experimental methodology, a methodology that has been tremendously successful in, and, indeed is the cornerstone of many other sciences. I see nothing in AI's research agenda that would

[9] Although I believe the situation is changing; recent conference proceedings appear to include an increasing number of experimental papers on agent design, and, in some other subfields of AI, notably machine learning and text understanding, there are many such papers.

preclude its also benefiting from controlled experimentation. Of course, adopting the experimental method entails adapting it to the particulars of the AI research program. In my comments below, I will give some necessarily sketchy suggestions about how we might adapt the methodology, and, in particular, how the challenge of generalizability can be met in AI. Following SH, I will use the Tileworld as an example.

5.2.1 Simplification in experimentation

> "Simplification, paring back the variables, far from invalidating results, is indeed required by the foundations of empirical design. The success of reductionism depends on measuring and reporting only that bit of cloth that can be understood and tested piecemeal. [Powers 1991, p.355]."

Experimentation mandates simplification. In investigating a complex phenomenon, the experimenter selectively attends to some aspects of it, namely, those that she believes are relevant to her hypotheses. She exerts control over those aspects of the phenomenon, manipulating them as necessary to test her hypotheses. At the same time, she holds constant those influences she believes are extraneous to her hypotheses, and allows or even forces random variation in those influences that she believes are "noise." This selective attention to and intentional manipulation of certain aspects of the phenomenon is the "paring back [of] the variables" noted in the quotation above.

Does SH object to simplification *per se*, i.e., does he believe that, to be useful, a hypothesis about agent design cannot make reference only to some aspects of an agent's architecture or environment? Although he appears to be inclined toward this conclusion when he asserts his belief that "it will be quite difficult to isolate particular aspects of a large planning problem," (p. 27), this is not his primary objection. Rather, what he views as dangerous is a particular way of achieving simplification in research on agent design, namely, by conducting experiments using highly simplified agents operating in highly simplified environments. This is what he terms "experimentation in the small." SH's introductory comments mention only objections to the use of simplified environments, but his criticisms of the Tileworld experiments show that also objects to the use of highly simplified agents.

I alluded earlier to my belief that it is necessary to make significant simplifications in the agents and environments we use in conducting experimentation. Large, realistic systems have generally been built without the benefit of a principled understanding of agent design—precisely what experimentation (supplemented with theorizing) aims at. As a result, it

is extraordinarily difficult to determine which mechanisms of these complex systems are responsible for which aspects of their behavior—in other words, to isolate the key properties of S and B. It is difficult to determine what in the system is essential to the observed behavior, and what is instead an artifact of the way the system happened to be implemented. In addition, when these systems are deployed in real environments, there is not a ready way to isolate and control key feature of those environments, i.e., to get a handle on E.

The testbeds that we surveyed in Section 4 are designed specifically to provide the researcher with the control needed to conduct experimentation—to enable her to control and monitor the conditions of the environment and to measure the behavior of a system embedded in that environment. In other words, a useful testbed will give the researcher a handle on B and on E.

To give the researcher a way to measure B, the testbed designer specifies what counts as successful behavior in the testbed environment, and provides instrumentation that measures success. To give the researcher a way to control and monitor E, the testbed designer selects some set of environmental features, and provides instrumentation that allows the researcher to control these. One potential objection is that the testbed designer thereby influences the experiments that can be conducted using the testbed; researchers may want to study other characteristics of B and E than those identified by the testbed designer. This, however, is only a problem if researchers are mandated to use particular testbeds. The problem disappears if we leave the decision about which testbed to use to individual researchers. A testbed is just a tool, and it is up to the researcher to determine the best tool for her current task. Indeed, in some cases, researchers may need to build their own tools to pursue the questions of interest to them. It is worth noting, though, that some testbeds may be more flexible than others—i.e., may more readily suggest ways to model a variety of environmental features and/or behavioral aspects, and thus be more amenable to modification by the testbed users. I will suggest below that flexibility is one of the strengths of the Tileworld system.

So far, I have focused on how a testbed allows control of B and E. It is, of course, also necessary for the researcher to have control of the system features, S. One way to achieve this is to use the same kind of parameterization in an agent embedded in a testbed environment as is used in the environment itself. One of the more useful features of the Tileworld system is precisely that it provides the experimenter with control over the embedded system as well as over the environment.

SH does not dispute the claim that simplification of the kind provided by testbed envi-

ronments and agents provides experimental control. What worries him is that the price we may pay for this control is too high. His main argument is that the very simplifications that provide the needed control also make it impossible to produce results that are in any sense "real" or generalizable, i.e., that can be shown to be applicable to larger AI applications. All three of the authors of this paper agree that that this problem, often called "realism," is the most difficult challenge facing researchers on agent design who adopt the experimental methodology we have discussed in this paper. However, we disagree about whether this difficulty in insurmountable.

5.2.2 Towards realism

The problem of realism is a challenge for experimentalists—for all experimentalists, not just those in AI. To achieve the experimental control they need, scientists in many disciplines have made use of simplified systems, and have thus had to address the question of how the lessons they learn using those systems may be applied to more complex phenomena. Yet the history of science is full of examples in which this challenge has been successfully met. To give just a few examples:

- Biologists have used the very simple organisms drosophila and *e.coli* in numerous experiments aimed at understanding the fundamental mechanisms of genetics. The results of these experiments have had tremendous significance for the theory of inheritance in all organisms, including humans.

- Neurobiologists have used aplysia, animals with only a few neurons, to conduct experiments investigating neuroplasticity. Again, the results have been generalized to theories about the ways in which human brains function.

- Engineers have built systems to simulate natural phenomena—wind tunnels and wave machines, for example. These simulations abstract away from much of the complexity of "real" environments. Nonetheless, experiments conducted using them have provided many valuable lessons about the effects of the modeled phenomena on engineered artifacts such as airplanes.

Of course, merely pointing out that many other sciences have been able to meet the challenge of realism is not, in and of itself, enough to demonstrate that AI researchers concerned with agent design will be able to do so. What is needed is a closer look at *how*

this challenge has been met. A widely used, introductory textbook on statistics describes the process of achieving realism as follows:

> "Most experimenters want to generalize their conclusions to some setting wider than that of the actual experiment. Statistical analysis of the original experiment cannot tell us how far the results will generalize. Rather the experimenter must argue based on an understanding of psychology or chemical engineering or education that the experimental results do describe the wider world. Other psychologists or engineers or educators may disagree. This is one reason why a single experiment is rarely completely convincing, despite the compelling logic of experimental design. The true scope of a new finding must usually be explored by a number of experiments in various settings.
>
> A convincing case that an experiment is sufficiently realistic to produce useful information is based not on statistics, but on the experimenter's knowledge of the subject-matter of the experiment.[Moore and McCabe 1989, p.270].

The key to achieving realism lies in the researcher's "knowledge of the subject-matter"; the researcher must provide an argument, based on her understanding of the subject matter, that in fact the results of her experiments do "describe the wider world." For such arguments to be satisfying, they must be informed by a rich theory of the phenomena in question. For the experimental program to succeed in AI, AI researchers will need to be more scrupulous about careful theory development; as I have claimed elsewhere [Pollack 1992], our field has not always valued theory development as an integral part of our work.

Research into agent design *begins* with a theory. Of course, the theory, in whole or in part, may be informed by the theorist's previous experiences building large, interesting systems. An experimental research program on agent design includes the following components (*cf.* Cohen's "MAD" methodology [Cohen 1991]).

- A theory T, describing some aspect(s) of agent design and the purported effect of those design aspects on agent behavior in certain environments, particularly describing the the agent's architecture, the environment, and the agent's behavior.[10]
- An implemented testbed environment E, and a description of the characteristics of that environment.

[10] A general question exists about the appropriate language for the researcher to use in articulating her theory: sometimes it will be the language of mathematics, other times a natural language, clearly used, may suffice.

- An implemented agent A, who will operate in the testbed environment, and a description of that agent's architecture, as implemented.

- Mappings describing the relationship between the "real" phenomena described by the theory to their intended analogues in the testbed environment, the relationship between the agent architecture described in the theory and its realization in the implemented agent, and the relationship between the agent's design and its performance in the testbed world.

A typical set of experiments will then evolve from some hypothesis, typically asserting that, under the some given environment conditions, some specified behavior will be observed in agents having some given architectural characteristics. Experiments can then be designed using the implemented (or, operationalized) analogue of this hypothesis, relating conditions in the testbed to observed behavior in the implemented agent. Such experiments can have several different types of results. They may confirm, deny, or suggest modifications to the hypotheses in the underlying theory. They may suggest needed changes to the testbed system and/or to the mappings between the actual and simulated environment. Experimentation may reveal flaws in the way the environment was modeled. They may suggest needed changes to the simplified agent and/or the mappings between the actual and simulated agent. Experimentation may reveal flaws in the way the agent was modeled, or its behavior measured. Perhaps most importantly, they may suggest additional experimentation that should be performed, either using the same testbed and agent or some other testbed and agent.

This last type of result is critical: experimentation is an iterative process. Part of the experimental program is to refine the mapping between a theory and its realization in implemented systems. And part of the experimental program is to iteratively refine the experiments themselves. As Moore and McCabe put it, "a single experiment is rarely completely convincing ... The true scope of a new finding must usually be explored by a number of experiments in various settings." For this to happen in research on agent design, great care must be given to the way in which theories are stated, and to the way in which those theories are operationalized in experimental settings. Testbeds and simplified agents make it possible to meet this latter requirement.

5.2.3 The Tileworld experience

To make this discussion more concrete, I want to describe briefly some of the experiences we have had in conducting experiments using the Tileworld system. I will focus on the Tileworld because it is the experimental work with which I am most familiar, and because SH addresses it in his comments. I do not mean to suggest that the Tileworld is the ultimate testbed or one that all researchers should use in their work. On the contrary, for reasons I have already discussed, it is essential that AI researchers use a variety of testbed systems in their experimentation. Moreover, the Tileworld is an early, prototype testbed system, and, in using it, we have not only learned about agent design, but have also learned a great deal about the testbed design. These lessons have led us to make a number of changes and extensions to the oritinal system reported on in [Pollack and Ringuette 1990], some of which I will mention below.

Our initial goal in building the Tileworld was to study a particular, well-developed theory of resource-limited reasoning, called IRMA (the Intelligent Resource-Limited Machine Architecture), that we had previously developed [Bratman *et al.* 1988,Bratman 1987,Pollack 1991]. This theory built on a detailed philosophical analysis of the role of intention in managing reasoning; our aim was to investigate certain underspecified aspects of this model. In particular, we began with a theoretically motivated strategy for coping with changing environments—the strategy of commitment-based *filtering*. Roughly speaking, this strategy involves committing to certain plans and tending to ignore options for action that are deemed incompatible with those plans. Filtering can be more or less strict, and we wanted to determine the environmental conditions under which stricter filtering was more advantageous. In addition, there are various ways to realize the notion of "strictness," and we wanted to explore the effects of these alternatives on agent behavior in different environmental conditions. The environmental condition that we suspected would be most important was average rate of change in the environment. Details are be found in [Pollack and Ringuette 1990]; this brief sketch is meant to highlight the fact that, underlying our attempt to relate S (in this case, conditions on filtering), E (average rate of change), and B (the agent's overall performance), was a larger theory about the role of intentions in resource-limited reasoning.

The experiments that we conducted, as well as those performed by others using the Tileworld [Kinny 1990,Kinny and Georgeff 1991,Oh 1991,Hendler and Kinny 1992], led to each of the kinds of results I described above:

- They provided preliminary confirmation of some parts of the theory. Experimentation showed that strict filtering of incompatible options, coupled with an appropriate overriding mechanism, is viable at least under some circumstances [Kinny and Georgeff 1991,Kinny 1990]. In other words, commitment to one's plans can be a valuable strategy for managing a changing environment. Experimentation also suggested needed modifications to the theory. For example, Oh observed that the agent's performance is hindered by its inability to immediately adopt certain extremely promising options without deliberation [Oh 1991]. The original theory included a mechanism for short-circuiting deliberation to eliminate a new option, but it lacked a mechanism for short-circuiting deliberation to immediately adopt a new option. The theory thus needed to be modified to include a new mechanism of the latter type.

- The experiments suggested needed changes to the testbed environment. As SH correctly points out, the original Tileworld testbed was extremely homogeneous—essentially, the world only presented one type of top-level goal (hole-filling). This fact limited the range of experiments we could conduct: there was no way to explore the behavior of agents who had to perform complex (and thus, computationally costly) plan generation. We have, since the publication of [Pollack and Ringuette 1990], increased the complexity of the Tileworld environment, so that we can study situations in which a wider range of options are presented to the agent.

- The experiments also suggested needed changes to the agent embedded in the Tileworld environment. Early experiments showed that the simplifications we made in the deliberation and plan generation component of the system were too extreme. Both processes were uniformly inexpensive, and we were thus unable adequately to explore the advantages of the filtering process, whose intent is to cut down on the amount of deliberation and planning needed [Pollack and Ringuette 1990]. This limitation led us subsequently to increase the complexity of the deliberation process. Note the interaction between this change and the previous one described; the added complexity in the agent depended on added complexity in the environment.

- Finally, the experiments suggested a large number of additional experiments that need to be conducted to expand and strengthen our original theory. SH, in fact, gives many examples of such experiments. He wonders about the significance of the agent's ability to perform some planning problems optimally (p. 21). He suggests that the degree of

(un)predictability in an environment may be an important influence on the value of commiting to one's plans (p. 22). He asks, "What if the agent doesn't have immediate, perfect, cost-free information about the appearance of holes? What if the designer does not have an optimal and efficient planner at his disposal?" (p. 26). Questions such as these are precisely what a theory of agent design should answer, and directly suggest experiments that could be performed, using Tileworld and/or other testbed systems. We count as a success of our experience with the Tileworld that it has led a number of researchers to ask just these kinds of questions. Moreover, the Tileworld has proven to be flexible, in the sense that it can readily be modified to support experiments investigating environmental and agent-design issues other than those for which it was originally designed.

One error that we made in the initial Tileworld experiments was a failure to be precise enough in the terminology we used to describe our theory and its realization in the testbed and simplified agent.[11] Instead of using qualitative terms, we should perhaps have developed quantitative analyses. For example, instead of describing environments as "fast" or "slow" relative to some arbitrary baseline, we might have defined the rate of environmental changes as the ratio between the average period of time between changes in the environment and the average amount of time it takes an agent to form an arbitrary plan. Qualitative definitions like this would certainly have facilitated the specification of the mapping functions between "real" phenomena and the Tileworld operationalization of them.

It is clear that significant effort must be put into the development of vocabularies for describing agents and environments and their realizations in implemented systems. I agree completely with SH that the real contribution of this line of research will be "to come up with the right way or characterizing the agent, the world, and their relationship" (p. 26). This is the primary goal of our ongoing work. However, I disagree strongly with SH when he goes on to claim that so far the terms used in the Tileworld studies (and in all other experimentation "in the small") are "so specific as to be applicable only to the experimental domain [or] so vague as to be vacuously true".

Consider the Tileworld results that he describes as vacuously true. He states these in terms of the circumstances under which it is advantageous to reconsider the plans to which

[11] Another error was our failure to to provide a clean enough interface between the agent and the environment; it is more difficult than we had hoped to excise the IRMA-based embedded agent and replace it with an alternative. Also, as SH points out (p. 21), we employed an awkward mechanism, which has since been modified, for simulating concurrent acting and reasoning on a sequential machine.

one has already committed (e.g., be more inclined to reconsideration when the world is changing more rapidly; reconsider when your goal becomes impossible). But what is most important about the early Tileworld results is that they support the idea that commitment is a good idea in the first place: the results, as described by SH, have to do with refinements to that basic idea. Kinny and Georgeff found that commitment led to the most effective behavior under all the conditions they studied, provided the agent was given a minimal override policy that allows for reconsideration of goals that have become unachievable.

The key idea of the IRMA theory is that it pays for an agent in a dynamic environment to commit to certain courses of action, even though the environment may change so that some of those courses of action cease to be optimal. Local optimality—always doing what is best at a given time—must be sacrificed in the interest of doing well enough overall; commitment to one's plans generally rules out local optimality, but can help lead to overall satisficing, i.e., "good enough," behavior. Although I cannot restate the entire argument here (again, see [Bratman *et al.* 1988,Bratman 1987,Pollack 1991]), it should be said that this is far from being a claim that is so obvious that all reasonable people would assent to it.[12] SH says that he would be "surprised to hear about an agent that did *not* adopt these policies" (p. 25), but in fact the recent literature in agent design has been filled with examples of agents, specifically the so-called reactive agents, that are notable precisely because they do not commit to any plans; instead, they decide at each point in time what action is appropriate ([Agre and Chapman 1987,Brooks 1991,Schoppers 1987]). A standard attempt to resolve the debate between the reactivists and the deliberativists has been to suggest a middle road: rational agents sometimes should deliberate about and commit to plans, and other times should react more immediately to their environment. The Tileworld experiments conducted to date can be seen, at least in part, as an attempt to clarify the conditions under which each alternative is desirable.

5.2.4 Conclusion

In these comments, I have distinguished between two kinds of simplification in experimentation: (a) investigating hypotheses that focus on particular characteristics of a system, its behavior, and its environment, and (b) using simplified systems, operating in simplified environments, to conduct the experiments. I have claimed that the former is essential to all

[12] If you don't believe me, I invite you to listen to the objections that are raised when I give talks describing IRMA.

experimentation, and that, while in principle the latter is not necessary, *de facto* it is, given the current state of our science.

Although SH focuses, in his comments, on the difficulties involved in using testbeds and simplified agents in experimentation, in his conclusion he supports their use, provided that the hypotheses towards which they are directed were inspired by experiences with particular large-scale systems. Thus, he says that he is not "opposed to conducting ... experiments in controlled, overly simplified worlds [and can] imagine, for example, a researcher implementing some idea in a system, then building a small world that isolates the essence of that idea, then using the small world to explore the idea further" (p. 29). So apparently SH feels that the problem is *not* in the use of simplified systems and agents *per se*, but rather in the fact that researchers who have so far employed simplified systems and agents have been willing to investigate hypotheses that have been developed apart from the implementation of any particular system. Given this, it appears that the primary dispute between SH and me has little to do with the use of testbeds and simplified systems. We both agree that unprincipled fiddling with any systems (large or small) is just that. Experimentation must build on theorizing.[13] But SH demands that any theory worth investigating must derive directly from a large, implemented system, while I see no need for this restriction. Sometimes, hypotheses about agent design may result from other avenues of inquiry—such as the philosophical theorizing that led to IRMA—and it may be may be more effective to explore these theories experimentally before investing in large, complex systems that embody them.

5.3 Generalization of testbed results (Paul Cohen)

Much of the preceding discussion touches on the problem of generalizing results from research with testbeds. I will do the reader no service by recounting my coauthors' arguments. Instead I will try to clarify what testbeds are for, focusing on their role in the search for general rules of behavior.[14]

I was struck by Steve Hanks' repeated assertion that results from the Tileworld studies are difficult to interpret, so this will serve as the launching point for my own comments. All empirical results are open to interpretation. Interpretation is our job. When we read

[13]There is an exploratory phase of experimentation that may occur after initial attempts at verifying a particular theory and may sometimes look like "fiddling," but that is another matter.

[14]Much of what I will say arises from conversations with Professor Bruce Porter, of the University of Texas. Although I owe my current understanding of the issues to our discussions, I do not mean to imply that he agrees with everything here.

the results of a study we have to ask ourselves, what does this mean? We can answer the question several ways. First, we might say, Goodness gracious, this result deals a deadly blow to the prevailing theory of, say, "agent curiosity." Let's agree that this is unlikely for two reasons: we don't have a theory of agent curiosity, or a theory of any other agent behavior; and death-dealing empirical results are in any case rare. Second, we might interpret a study as a chink in the armor of a prevailing theory, as results from astronomy sometimes are interpreted as troublesome for the Big Bang theory. This too is unlikely because we don't have any theories that make predictions for results to contradict. Third, a study might be interpreted as supporting a prevailing theory, if we had any theories to support. Fourth, a result might suggest a theory or just a tentative explanation of an aspect of agent behavior. I interpret Kinney and Georgeff's paper in this way, as weak evidence for the theory that agents sometimes do better in unpredictable domains if they are "bold." And I have no sympathy for the complaint that the paper is difficult to interpret. Interpretation is our job, especially now, when we have no theories to do the job for us. In short, we ought to ask what our few empirical results mean—what theories they suggest, because we currently have no theories to provide interpretations—instead of asserting strenuously that they mean nothing.

Let us recognize that empirical results are rarely general. Interpretations of results might be general, but results are invariably tied to an experimental set-up. It is wrong to assert that because Kinney and Georgeff worked with a trivial testbed, their results have no general interpretation. I have already recounted one general interpretation: bold agents sometimes do better in unpredictable domains. Moreover, every substantive word in this interpretation has a precise meaning in the Tileworld. Thus, Kinney and Georgeff can say, "Bold agents sometimes do better in an unpredictable environment, and here is what we mean by bold, agent, sometimes, better and unpredictable. If you are interested in our theory, tell us what *you* mean by these terms and let us see if the theory generalizes."

Nothing prevents us inventing general theories as interpretations of results of testbed studies, and nothing prevents us designing additional studies to test predictions of these theories in several testbeds. For example, two students in my research group explored whether bold Phoenix agents do better as the Phoenix environment becomes more unpredictable. The experiment proved technically difficult because Phoenix agents rely heavily on failure-recovery strategies, so it is difficult to get them to commit unswervingly to any plan for very long. Their natural state is "bold," and they fail catastrophically when we make them less so,

so our results were inconclusive. But imagine the experiments had succeeded and we had accrued evidence that boldness really does help Phoenix agents when the environment becomes more unpredictable. Then two research groups—mine, and Kinney and Georgeff's—would have demonstrated the same result, right? Whether you agree depends on what you mean by "the same result." I mean this: Kinney and Georgeff offered a mapping from terms in their theory (bold, agent, better, sometimes, unpredictable) to mechanisms in the Tileworld, and I offered a mapping from the same terms to mechanisms in Phoenix, and we both found that a sentence composed from these terms—bold agents sometimes do better in an unpredictable environment—was empirically true. In reality, as I noted, we were unable to replicate Kinney and Georgeff's result. We failed for technical reasons; there was no easy way to create a Phoenix agent that was not "bold." Differences in experimental apparatus always make replication difficult; for example, Tileworld has just one agent and a limited provision for exogenous events, so it would be difficult to use Tileworld to replicate results from Phoenix. Still, these are only technical problems and do not provide a strong argument against the possibility of generalizing results from testbed research.

Testbeds have a role in three phases of research. In an *exploratory* phase, they provide the environments in which agents will behave in interesting ways. During exploration we will characterize these behaviors loosely; for example, we will observe behaviors that appear "bold" or "inquisitive." In exploratory research the principal requirement of testbeds is that they support the manifestation and observation of interesting behaviors, which is why I favor complex agents and testbeds over simple ones. In a *confirmatory* phase we tighten up the characterizations of behaviors and test specific hypotheses. In particular we will provide an operational definition of, say, "boldness" so that a data-collecting computer program can observe our agent's behavior and decide whether it is bold. We will test hypotheses about the conditions in which boldness is a virtue, and when we are done we will have a set of results that describe precise, testbed-specific conditions in which a precise, agent-specific behavior is good or bad. In confirmatory research the primary requirement of a testbed is that it provide experimental control and makes running experiments and collecting data easy. This is why Phoenix has a script mechanism for automatically running experiments and integrated data-collection, data-manipulation and statistical packages. In the third phase, *generalization*, we attempt to replicate our results. As I described earlier, several research groups might attempt to replicate "bold" behavior under comparable conditions as in the original experiment. Each group will have to design their own agent-specific, testbed-

specific definitions of "bold" and "comparable conditions." For example, uncertainty about the environment might be induced in agents by rapidly changing wind speed in Phoenix and by erratically moving holes in Tileworld. This requires testbeds to be parameterizable, and researchers to work closely during the generalization phase.

The "boldness theory" is general to the extent that boldness and unpredictability in Tileworld are similar phenomena as boldness and unpredictability in Phoenix and other testbeds. Similar agents in similar testbeds are apt to manifest similar behaviors, but this will not convince us that the behaviors are general. Generality is achieved when different agents in different testbeds exhibit common behaviors in common conditions. The more the agents and testbeds differ, the more difficult it will be to show that behaviors and conditions are common. If we had theories of behavior, we could show how conditions and behaviors in different testbeds are specializations of terms in our theories. But we do not have theories; we must bootstrap theories from empirical studies. Our only hope is to rely on our imaginations and abilities to interpret behaviors and conditions in different testbed studies as similar.

In conclusion, I believe results of testbed research can be generalized. Some features of testbeds will make it easier to observe, explain and test hypotheses about agents' behaviors. Generalization is done by scientists, not apparatus, so I strongly disagree with any implication that particular kinds of testbeds preclude generalization. Testbeds offer researchers the opportunity to tell each other what they observed in particular conditions. When a researcher publishes an observation, other researchers are responsible for the hard work required to say, I observed the same thing!

References

[Agre and Chapman 1987] Philip E. Agre and David Chapman. Pengi: An Implementation of a Theory of Activity. In *Proceedings, AAAI*, pages 268–272, 1987.

[Bond and Gasser 1988] Alan H. Bond and Les Gasser, editors. *Readings in Distributed Artificial Intelligence*. Morgan Kaufmann, 1988.

[Bratman *et al.* 1988] Michael E. Bratman, David J. Israel, and Martha E. Pollack. Plans and Resource-Bounded Practical Reasoning. *Computational Intelligence*, 4:349–355, 1988.

[Bratman 1987] Michael E. Bratman. *Intention, Plans and Practical Reason*. Harvard University Press, Cambridge, MA, 1987.

[Brooks 1991] Rodney A. Brooks. Intelligence without Reasoning. In *Proceedings of the Twelth International Joint Conference on Artificial Intelligence*, pages 569–595, Sydney, Australia, 1991.

[Chapman 1990] David Chapman. On Choosing Domains for Agents, June 1990. Proceedings of the NASA/AMES Workshop on Benchmarks and Metrics.

[Chrisman and Simmons 1991] L. Chrisman and Reid Simmons. Senseful Planning: Focusing Perceptual Attention. In *Proceedings, AAAI*, 1991.

[Chrisman *et al.* 1991] Lonnie Chrisman, Rich Caruana, and Wayne Carriker. Intelligent Agent Design Issues: Internal Agent State and Incomplete Perception. In *AAAI Fall Symposium Series: Sensory Aspects of Robotic Intelligence*, 1991.

[Cohen *et al.* 1990] Paul R. Cohen, Adele E. Howe, and David M. Hart. Intelligent Real-Time Problem Solving: Issues and Examples. In L. Erman, editor, *Intelligent Real-Time Problem Solving: Workshop Report*, Palo Alto, CA, 1990. Cimflex Teknowledge Corp.

[Cohen 1991] Paul R. Cohen. A survey of the Eighth National Conference on Artificial Intelligence: Pulling together or pulling apart? *AI Magazine*, 12:16–41, 1991.

[Dean and Boddy 1988] Tom Dean and Mark Boddy. An Analysis of Time-Dependent Planning. In *Proceedings AAAI*. AAAI, August 1988.

[Firby and Hanks 1987] R. James Firby and Steve Hanks. A Simulator for Mobile Robot Planning. In *Proceedings of the DARPA Knowledge-Based Planning Workshop*, pages 23–1, December 1987.

[Firby 1989] R. James Firby. Adaptive Execution in Complex Dynamic Worlds. Technical report, Department of Computer Science, Yale University, January 1989.

[Greenberg and Westbrook 1990] Michael Greenberg and David L. Westbrook. The Phoenix Testbed. Technical Report COINS TR 90–19, Computer and Information Science, University of Massachusetts, 1990.

[Haddawy and Hanks 1993] P. Haddawy and S. Hanks. Utility Models for Goals-Directed Decision-Theoretic Planners. *Artificial Intelligence*, 1993. Submitted.

[Hanks and Badr 1991] Steve Hanks and Badr Al Badr. Critiquing the Tileworld: Agent Architectures, Planning Benchmarks, and Experimental Methodology. Technical Report 91–10–31, Department of Computer Science and Engineering, University of Washington, October 1991.

[Hanks and McDermott 1992] Steve Hanks and Drew McDermott. Modeling a Dynamic and Uncertain World II: Projecting Courses of Action. In Preparation, 1992.

[Hanks and McDermott 1993] Steve Hanks and Drew McDermott. Modeling a Dynamic and Uncertain World I: Symbolic and Probabilistic Reasoning about Change. *Artificial Intelligence*, 1993. To appear.

[Hanks *et al.* 1992] Steve Hanks, Dat Nguyen, and Chris Thomas. The New Truckworld Manual. Technical report, Department of Computer Science and Engineering, University of Washington, 1992. Forthcoming.

[Hanks 1990a] Steve Hanks. Practical Temporal Projection. In *Proceedings AAAI*, 1990.

[Hanks 1990b] Steve Hanks. Projecting Plans for Uncertain Worlds. Technical Report 756, Department of Computer Science, Yale University, January 1990.

[Hart and Cohen 1990] David M. Hart and Paul R. Cohen. Phoenix: A Testbed for Shared Planning Research, June 1990. Proceedings of the NASA/AMES Workshop on Benchmarks and Metrics.

[Hendler and Kinny 1992] James Hendler and David Kinny. Empirical Experiments in Selective Sensing with Non-Zero-Cost Sensors. Technical report, University of Maryland, 1992.

[Kinny and Georgeff 1991] David N. Kinny and Michael P. Georgeff. Commitment and Effectiveness of Situated Agents. In *Proceedings of the Twelfth International Joint Conference on Artificial Intelligence*, pages 82–88, Sydney, Australia, 1991.

[Kinny 1990] David N. Kinny. Measuring the Effectiveness of Situated Agents. Technical Report 11, Australian AI Institute, Carlton, Australia, 1990.

[Langley and Drummond 1990] Pat Langley and Mark Drummond. Toward an Experimental Science of Planning. In *Workshop on Innovative Approaches to Planning, Scheduling, and Control*. DARPA, November 1990.

[Law and Kelton 1981] Averill Law and W. David Kelton. *Simulation Modeling and Analysis*. McGraw-Hill, 1981.

[McDermott 1981] Drew McDermott. Artificial Intelligence Meets Natural Stupidity. In John Haugland, editor, *Mind Design: Essays in Philosophy, Psychology, and Artificial Intelligence*. MIT Press, 1981.

[Minton *et al.* 1990] Steven Minton, Mark D. Johnston, Andrew B. Philips, and Philip Laird. Solving Large-Scale Constraint Satisfaction and Scheduling Problems Using a Heuristic Repair Method. In *Proceedings of the Ninth National Conference on Artificial Intelligence*, pages 17–24, Boston, MA, 1990.

[Montgomery and Durfee 1990] Thomas A. Montgomery and Edmund H. Durfee. Using MICE to Study Intelligent Dynamic Coordination. In *Second International Conference on Tools for Artificial Intelligence*, pages 438–444. IEEE, 1990.

[Montgomery *et al.* 1992] Thomas A. Montgomery, Jaeho Lee, et al. MICE Users Guide. Technical report, Department of Electrical Engineering and Computer Science, University of Michigan, January 1992.

[Moore and McCabe 1989] David S. Moore and George P. McCabe. *Introduction to the Practice of Statistics*. W.H. Freeman and Company, New York, 1989.

[Oh 1991] John Oh. A Study of Filtering and Deliberation Strategies in Tileworld. Unpublished manuscript., 1991.

[Philips and Bresina 1991] Andrew B. Philips and John L. Bresina. NASA TileWorld Manual. Technical Report TR–FIA–91–04, NASA Ames Research Center, May 1991.

[Philips *et al.* 1991] Andrew Philips, Keith J. Swanson, Mark E. Drummond, and John L. Bresina. The NASA TileWorld Simulator (Instantiating Key Domain Attributes While Discarding Irrelevant Semantic Baggage), 1991. Position paper for the AAAI-91 Workshop on Real-Time AI.

[Pollack and Ringuette 1990] Martha E. Pollack and Marc Ringuette. Introducing the Tileworld: Experimentally Evaluating Agent Architectures. In *Proceedings of the Ninth National Conference on Artificial Intelligence*, pages 183–189, Boston, MA, 1990.

[Pollack *et al.* 1993] Martha E. Pollack, David Joslin, Arthur Nunes, and Sigalit Ur. Experimental investigation of an agent design strategy. In preparation., 1993.

[Pollack 1991] Martha E. Pollack. Overloading Intentions for Efficient Practical Reasoning. *Noûs*, 25(4):513–536, 1991.

[Pollack 1992] Martha E. Pollack. The Uses of Plans. *Artificial Intelligence*, 57(1):43–69, 1992.

[Powers 1991] Richard Powers. *The Gold Bug Variations.* William Morrow and Company, New York, 1991.

[Rosenschein *et al.* 1990] Stanley J. Rosenschein, Barbara Hayes-Roth, and Lee D. Erman. Notes on Methodologies for Evaluating IRTPS Systems. In L. Erman, editor, *Intelligent Real-Time Problem Solving: Workshop Report*, Palo Alto, CA, 1990. Cimflex Teknowledge Corp.

[Russell and Wefald 1991] Stuart J. Russell and Eric H. Wefald. *Do the Right Thing: Studies in Limited Rationality.* MIT Press, 1991.

[Schoppers 1987] Marcel J. Schoppers. Universal Plans for Reactive Robots in Unpredictable Environments. In *Proceedings of the Tenth International Joint Conference on Artificial Intelligence*, 1987.

[Sussman 1975] Gerald J. Sussman. *A Computer Model of Skill Acquisition.* American Elsevier, New York, 1975.

[Weld and de Kleer 1989] Daniel S. Weld and Johan de Kleer, editors. *Readings in Qualitative Reasoning about Physical Systems.* Morgan Kaufmann, 1989.

[Wellman and Doyle 1991] Michael P. Wellman and Jon Doyle. Preferential Semantics for Goals. In *Proceedings of the Ninth National Conference on Artificial Intelligence*, pages 698–703, Anaheim, CA, 1991.