# The Invisible Hand: How Evaluation Guides AI Research

## COINS Technical Report 88-21

**Paul R. Cohen** and **Adele E. Howe**

Experimental Knowledge Systems Laboratory
Department of Computer and Information Science
University of Massachusetts
Amherst, MA

**Abstract**

Evaluation should be a mechanism of progress both within and across AI research projects. For the individual, evaluation can tell us how and why our methods and programs work, and so tell us how our research should proceed. For the community, evaluation expedites understanding of available methods and so their integration into further research. In this paper, we present a five stage model of AI research and describe guidelines for evaluation that are appropriate for each stage. The guidelines, in the form of evaluation criteria and techniques, suggest how to perform evaluation. We conclude with a set of recommendations that suggest how to encourage evaluation of AI research.

---

We are indebted to Scott Anderson, Carole Beal, Carol Broverman, David Day, Mike Greenberg, Tom Gruber, Cynthia Loiselle, and Paul Utgoff for their comments on drafts of this paper and to Peter Patel-Scheider for sending us guidelines for reviewers.

This research was sponsored by DARPA-RADC contract F30602-85-C-0014, ONR University Research Initiative grant N00014-86-K-1764, and ONR Contract AFOSR 4331690-01.

> Every individual endeavors to employ his capital so that its produce may be of greatest value ... And he is in this led by an INVISIBLE HAND to promote an end which was no part of his intention by pursuing his own interests he frequently promotes that of society more effectually than when he really intends to promote it.
>
> Adam Smith, *Wealth of Nations*, 1776

Evaluation means making observations of all aspects of one's research. Often we think of evaluation only in terms of how well AI systems perform; yet, it is vital to all stages of research from early conceptualization to retrospective analyses of series of programs. Indeed, some of the most informative observations are not performance measures, but rather describe why we are doing the research, why our tasks are particularly illustrative, why our views and methods are a step forward, how completely they are implemented by our programs, how these programs work, whether their performance is likely to increase or has reached a limit (and why), and what problems we encounter at each stage of our research. To the research community, these observations are more important than performance measures because they tell us how research should proceed.

So ideally, evaluation should be a mechanism of progress both within and across individual AI research projects. Evaluation provides impetus to the research cycle. It opens new avenues of research, because experiments often raise new questions as others are answered, and because it identifies deficiencies and thus problems for further research. Evaluation provides a basis for the accumulation of knowledge. Without evaluation we cannot replicate results. Evaluation is how you convince the community that your ideas are worthwhile, that they work, and how they work. Because our colleagues rarely have access to run-time data and programs at various stages of development, we have an unprecedented responsibility to evaluate and communicate all aspects of our research. Unfortunately, we rarely publish performance evaluations, still less evaluations of other stages of research.

Evaluation is not standard practice in part because our methodology is vague. Where other sciences have standard experimental methods and analytic techniques, we have faith—often groundless and misleading—that building programs will somehow be informative. Where other sciences expect specific aspects of research to be presented (e.g., hypotheses, related research, experimental methods, analyses and results), empirical AI has no comparable standards. Now, we do not advocate blindly adopting the empirical methods of, say, the behavioral sciences; in fact, we have argued elsewhere that they are inappropriate to AI [4]. But we are saying that progress in AI would be amplified by a methodology that emphasizes evaluation, for the overarching reason that we can do our own research better if we know, in detail, the state of our colleagues' research.

Where will this methodology—and specifically the evaluation criteria, experiment

designs, and analytic tools—come from? The challenge is that we must develop it ourselves. Explorations along these lines have been reported in specific areas, for example, machine learning [9] and expert systems [7,6,16]. Discussions of broader scope focus on the role of experiments in AI [3,14]. Although these papers address different facets of evaluation (and even taken together, do not comprise a complete methodology for AI) a common theme is that we must develop *our own* evaluation methods, appropriate to AI practice. If we try to retrofit the methods of other fields they will probably become impediments. Thus, in this paper we have been motivated not by envy of "Scientific Method," but by the sense that we are wasting opportunities to understand, by empirical and analytic studies, the intelligent artifacts we build at great expense.

This paper adopts a simple multistage model of AI research and describes the kinds of evaluation that are appropriate at each stage. The evaluations are stated as questions (presented in Figures 1 – 5) and discussed in the text. Many are questions we ask whenever we hear about new research, questions we want researchers to answer when they publish their work. We also describe in schematic form some common kinds of experiments with AI systems. The tangible contribution of the paper is the organization of these questions and experiment schemas into five categories—corresponding with stages of research. More important, by far, is our intent that the paper should promote discussion and development of evaluation techniques and, more broadly, AI methodology. In the final section of the paper we introduce some methodological issues that we haven't space to address, hoping to provide an impetus for further discussion.

# 1 Evaluating Each Stage of Research

We present empirical AI in terms of a five-stage cycle: In the first stage a topic is refined to a task and a view of how to accomplish the task; in the second, the view is refined to a specific method; in the third, a program developed to implement the method; in the fourth, experiments are designed to test the program; in the fifth, the experiments are run; lastly, the cycle begins again, informed by the experiment results. (A similar model is described in [3].) You may object that this isn't how AI research works. First, this superficial description seems severely top-down. But we will show how evaluations can produce unexpected conclusions and thus "result-driven" iterations at each stage of the cycle. Second, the model seems idealized because it suggests that AI researchers always design, run, and analyze experiments, and we don't. But although it is idealized, it isn't unattainable and in fact won't require many modifications to standard AI practice.

## 1.1 Stage 1: Refine the topic to a task

Empirical AI begins when researchers find particular topics fascinating. The first stage of the research cycle involves simultaneously refining the research topic to a *task* and identifying a *view*. A task is something we want a computer to do, and a view is a "pre-design," a rough idea about how to do it. This stage takes a lot of effort; researchers don't simply say, "Ok, we are fascinated by discovery, so let's try mathematical discovery as a task and heuristic search as a view" [10]. The process is iterative because if the task and view don't fare well by the evaluation questions in Figure 1, you have to modify and re-evaluate them.

The questions in Figure 1 address two basic evaluation concerns: can you justify the research task to yourself and to the community, and is your view of how to solve the task viable?[1] Question 1 asks us to justify the task: Why is it interesting? If it has been studied before and you are advocating a novel view (a reformulation), why will this new perspective be an improvement? Question 2 asks whether one's view of the task will work. Perhaps the task itself is intractable, or impossible given the resources, or requires expertise or hardware that isn't available. Question 3 suggests re-evaluating whether, after narrowing the scope of a task, it still exemplifies a research topic. This is particularly important when the topic is general (e.g., mechanical design) but the task is specific (e.g., designing pulley systems). Question 4 asks what has been left out of research tasks; for example, if your problem is machine learning, you may decide to ignore the new term problem and focus instead on generalization techniques.

Questions 5 and 6 address whether we have a productive research plan. Most large projects require solutions to multiple problems—designing representations and control strategies, and building knowledge bases. Question 5 concerns which of these will be research contributions of the project. Question 6 asks whether you understand the problem well enough to recognize a solution.

Evaluations during this stage direct one's own research, and also provide the AI community with carefully justified tasks, views, and reformulations.

## 1.2 Stage 2: Design the method

At the next stage, one's view is refined to a method for solving the task. The word "method" implies a single algorithm, such as $A^*$ [1], candidate elimination [12], or Waltz filtering [18]. But frequently, the method for a task combines several algorithms and assorted knowledge structures. For example, the method in Hearsay-II involved many knowledge sources, data-driven and opportunistic control, and a novel communication structure [5]. Although this complexity strains the word "method", we will maintain it

[1] In the questions, we sometimes refer to the *problem*, by which we mean the task and the view combined.

1. Is the task significant? Why?
   (a) If the problem has been previously defined, how is your reformulation an improvement?
2. Is your research likely to contribute meaningfully to the problem? Is the task tractable?
3. As the task becomes more specifically defined for your research, is it still representative of a class of tasks?
4. Have any interesting aspects been abstracted away or simplified?
   (a) If the problem has been previously defined, have any aspects extant in the earlier definition been abstracted out or simplified?
5. What are the subgoals of the research? What key research tasks will be/have been addressed and solved as part of the project?
6. How do you know when you have successfully demonstrated a solution to the task? Is the task one in which a solution can be demonstrated?

Figure 1: Criteria for evaluating research problems

to remind us that we don't jump immediately into building programs, but first decide *how* we want to solve tasks.

Criteria for evaluating methods are presented in Figure 2. The first two questions suggest general criteria for comparing the method with previous work. Question 3 asks whether, and to what extent, these comparisons are qualified by the method's reliance on other factors. The remaining questions ask how well you understand how and why the method works: its underlying assumptions or limitations, its applicability or scope, and its relationship to the problem itself.

Many of these questions cannot be answered until the method is implemented in a program. In this ideal model of AI research, that's why we build programs. But let us consider two cases in which programming is not necessary for evaluation. First, some methods can be evaluated analytically without programming (e.g., $A^*$ search, candidate elimination). That is, the questions in Figure 2 can be answered by proofs or other analyses, or by reference to extant programs. Second, a method may already be so well understood that neither implementing it nor evaluating it will tell us anything we do not already know. Now, in both cases, exploratory programming may help us refine the method we want to solve the task, but we distinguish this role of programming—refining a method—from programming for the purpose of experimenting with and evaluating a method. Unfortunately, exploratory programming drifts easily into building systems, and we begin to focus on solving the task, and forget that, from the standpoint of empirical AI research, the purpose of building systems is to tell us something about

1. How is the method an improvement over existing technologies?
   (a) Does it account for more situations? (input)
   (b) Does it produce a wider variety of desired behaviors? (output)
   (c) Is the method expected to be more efficient? (space, solution time, development time, etc.)
   (d) Does it hold more promise for further development? (for example, due to the opening up of a new paradigm)
2. Is there a recognized metric for evaluating the performance of your method? (e.g., normative, cognitively valid, etc.)
3. Does it rely on other methods? (Does it require input in a particular form or preprocessed? Does it require access to a certain type of knowledge base or routines?)
4. What are the underlying assumptions? (known limitations, scope of expected input, scope of desired output, expected performance criteria, etc.)
5. What is the scope of the method?
   (a) How extendible is it? Will it easily scale up to a larger knowledge base?
   (b) Does it address exactly the task? portions of the task? a class of tasks?
   (c) Could it, or parts of it, be applied to other problems?
   (d) Does it transfer to more complicated problems? (perhaps more knowledge intensive or more/less constrained or with more complex interactions)
6. When it cannot provide a good solution, does it do nothing or does it provide bad solutions or does it provide the best solution given the available resources?
7. How well is the method understood?
   (a) Why does it work?
   (b) Under what circumstances, won't it work?
   (c) Are the limitations of the method inherent or simply not yet addressed?
   (d) Have the design decisions been justified?
8. What is the relationship between the problem and the method? Why does it work for this task?

Figure 2: Criteria for evaluating methods

our methods that we don't already know and can't learn by analysis. We build too many systems and evaluate too few.

## 1.3 Stage 3: Build a Program

But if our method requires programming not merely to implement it, but to understand whether and why it works, then we must build a program that supports experiments. Although in practice this stage may be indistinguishable from exploratory programming in the previous stage, its purpose is different, and so the evaluations we do at this stage are different.

The criteria in Figure 3 concern whether the program is *informative*. Question 1 asks whether its internal and external behavior clearly demonstrate the method. Question 2 implies that programs are rarely informative if they are designed to run on only a single example. Question 3 suggests specific ways to assess how well the program implements the method. Finally, because the purpose of building a program is to tell us something we didn't already know, question 4 asks whether we can predict performance in advance.

1. How demonstrative is the program?
   (a) Can we evaluate its external behavior?
   (b) How transparent is it? Can we evaluate its internal behavior?
   (c) Can the class of capabilities necessary for the task be demonstrated by a well-defined set of test cases?
   (d) How many test cases does it demonstrate?
2. Is it specially tuned for a particular example?
3. How well does the program implement the method?
   (a) Can you determine the program's limitations?
   (b) Have parts been left out or kludged? Why and to what effect?
   (c) Has implementation forced a more detailed definition or even re-evaluation of the method? How was that accomplished?
4. Is the program's performance predictable?

Figure 3: Criteria for evaluating method implementation

These questions (especially question 4) illustrate the iterative nature of evaluation at each stage: We don't build a program first and then throw it away if it is uninformative or utterly predictable; we develop informative, interesting programs by

continuing evaluation—just as we iteratively evaluate and develop tasks, views, methods, and experiments. However, unlike the earlier evaluations of tasks, views, and methods, evaluation at this stage is primarily for the individual researcher, not for the community at large.

## 1.4 Stage 4: Design Experiments

The fourth stage of the research cycle is to design experiments with the newly-implemented system. Just as we evaluate the design and construction of systems from the standpoint of whether they are informative, so we should evaluate whether experiments with those systems will be informative. Criteria for evaluating experiments (as opposed to their outcomes) are presented in Figure 4. The purpose of question 1 is to encourage us to test our programs with many, qualitatively different examples, that illustrate both the abilities and limitations of our programs. Questions 2 and 3 ask whether our programs should be compared with a standard (e.g., an expert), and what specific comparisons will be made. Question 4 asks whether our experiments provide enough evidence to claim that a program is general, and question 5 suggests criteria for comparing programs. Both acknowledge that programs (and their underlying methods) are evaluated in the context of other research projects, and so raise the methodological issues of how results accumulate and generalize over projects (see also Sec. 2 and [4]).

These questions address *what* may be evaluated in experiments but don't say *how* the experiments should be conducted. AI has been evolving rudimentary, informal experiment schemas that address some of the questions in Figure 4. Here we describe five experiments schemas:

**Comparison studies.** The basic form of a comparison study is that we select one or more *measures* of a program's performance, then both the program and a *standard* solve a set of problems, and finally the solutions are compared on the measures. For example, we may compare the average number of subgoal violations generated by one planning program on a set of problems (the measure) with the same measure on another extant program (the standard). Typically, the programs will implement different methods, or they could be different configurations of a single program.

Variations on the basic form depend on what you want to demonstrate. For example, if you want to measure whether the program's performance is consensual, you may compare the program to a panel of human experts. You may also include novices—an interesting control condition to ensure that successful performance requires expertise.[2] Sometimes the performance of a program can be compared with objective, recognized standards. Normative theories, such as probability

[2]Shortliffe ran a panel of experts and novices in his studies of MYCIN [17].

1. How many examples can be demonstrated?
   (a) Are they qualitatively different?
   (b) Do these examples illustrate all the capabilities that are claimed? Do they illustrate limitations?
   (c) Is the number of examples sufficient to justify the inductive generalizations?
2. Should program performance be compared to some standard? its tuned performance? other programs? people (cognitive validity)? experts and novices (expert performance)? normative behavior? outcomes? (either from the real world or from simulations)
3. What are the criteria for good performance? Who defines the criteria?
4. If the program purports to be general (domain-independent),
   (a) Can it be tested on several domains?
   (b) Are the domains qualitatively different?
   (c) Do they represent the class of domains?
   (d) Should performance in the initial domain be compared to performance in other domains? (Do you expect that the program is tuned to perform best in domain(s) used for debugging?)
   (e) Is the set of domains sufficient to justify inductive generalization?
5. If a series of related programs is being evaluated,
   (a) Can you determine how differences in the programs are manifested as differences in behavior?
   (b) If the method was implemented differently in each program in the series, how were these differences related to the generalizations?
   (c) Were difficulties encountered in implementing the method in other programs?

Figure 4: Criteria for evaluating experiment design

theory, provide one kind of standard; for example, some researchers argue that because human experts are incapable of integrating probabilistic information consistently, their performance should not set standards. Another kind of standard is provided by real or simulated worlds; we might evaluate a complex planner by seeing whether it generates plans that succeed in the world. All these examples suggest that our measures and standards depend heavily on what we want to demonstrate and, ultimately, on our research goals.

A related scheme, though not strictly a comparison study, has humans judging or scoring the program's performance. This happens when we need to measure whether programs get the "right" solution, but the test problems have so many acceptable solutions that a program and a standard cannot be expected to generate the same ones.

**Ablation and substitution studies.** We can evaluate the contribution of individual components to the performance of complex systems by removing or replacing those components. Removing components (ablation [13]) is informative in systems that can solve problems without them; for example, we might assess the contribution of caching by running a system without caching. It takes little insight to predict *some* effect, the goal is to find out whether performance on all types of problems is equally affected by the presence of a cache, and if not, what interactions between the cache and the problem type explain the variance.

Many AI systems are so brittle that they collapse when components are removed. In these cases we might substitute "dumb" components for those we hope to show are "smart"; for example, we might substitute an exhaustive control strategy for a sophisticated opportunistic one. This suggests

**Tuning studies.** By tuning a system to perform as well as possible on a set of test data, we can learn how much performance can be improved, how difficult it is to achieve, and whether the resulting system can still solve other test cases. From a research perspective, it seems wasteful and potentially misleading to tune systems just to increase their performance, without addressing these questions.

**Limitation studies.** By testing a program at its known limits, we can better understand its behavior in adverse conditions. We can push a program to its limits by providing imperfect data (rearranged, noisy, incomplete, or incorrect), restricted resources (computation time or available knowledge), and perverse test cases.

**Inductive studies.** One way to support claims of generality is to solve "new and different" problems. If we claim that, say, a mechanical design system is general, then we may want to run problems in many areas of mechanical design—pulley systems, I-beams, extrusions, and so on [8,15]. Even if we don't claim a program

is general, we must at least test it on problems other than the ones we used to develop it.

The purpose of evaluation at this stage is to convince the researcher and the community that studies of a program (or programs)—independent of their results—are well-designed and complete. Experiment schemas, if we could specify them in adequate detail, would offer researchers a shorthand to describe their studies (e.g., "we ran a comparison study between versions 1, 2, and 3 of the program, measuring hit rate on three data sets that were characterized by their average signal-to-noise ratio ...").

## 1.5 Stage 5: Analyze Experiment Results

Now that our method has been implemented in a fully-instrumented program, and our experiments are well designed, we can ask whether the system works and why it works. The first three questions in Figure 5 evaluate the program by its performance on the task. Questions 4, 5 and 6 assess the utility of the program. Questions 6 and 7 identify the program's frailties and strengths: when does it break and what components contribute to its successful operation? Question 8.e is most salient to generalization of results because it justifies good performance for the program in terms of the task, a relationship that might be exploited in the design of other methods.

The purpose of evaluation at this stage is to convince the research community that your methods are viable, find their limitations in both performance and scope, and suggest further research.

# 2 Recommendations

Each stage in our simple model of research affords opportunities for evaluation; indeed, iterations at each stage and cycles through the model are driven by evaluation. The first recommendation is that we as AI researchers should convey to the community the answers to the questions we ask at each stage of research.

Second, we need to develop our methodology. The research cycle and the questions and experiments outlined above form only a skeleton of a methodology. One way to slowly fill out this skeleton is to carefully describe our experiments and results. Another is to publish methodological papers. For example, we want to see more extensive discussions along these lines:

- Discussions of experiment schemas, with appropriate control conditions, measures, standards, and analyses of results. We hope to see schemas that are more detailed and task-specific than the few we discussed in Section 1.4.

  We also hope to see discussions of specific evaluation criteria in the context of experiment schemas. We know that the criteria we presented in Figures 1 – 5 are

1. How did program performance compare to its selected standard? (e.g., other programs, people, normative behavior, etc.)
2. Is the program's performance different from predictions of how the method should perform?
3. How efficient is the program? time/space? knowledge requirements?
4. Did the program demonstrate good performance?
5. Did you learn what you wanted from the program and experiments?
6. Is it easy for the intended users to understand?
7. Can you define the program's performance limitations?
8. Do you understand why the program works or doesn't work?
   (a) What is the impact of changing the program even slightly?
   (b) Does it perform as expected on examples not used for debugging?
   (c) Can the effect of different control strategies be determined?
   (d) How does the program respond if input is rearranged, more noisy, or missing?
   (e) What is the relationship between characteristics of the test problems and performance (either external or internal if program traces are available?)
   (f) Can the understanding of this program be generalized to the method? to characteristics of the method? to a larger class of tasks?

Figure 5: Criteria for evaluating what the experiments told us

in some respects a bad compromise: too general to be very informative, and not general enough to encompass all research in empirical AI (e.g., they don't tell us how to evaluate methodological papers like this one).

- Discussions of what it means to replicate and generalize results. When can we claim that a particular method is general? When it has been implemented in one program and run on many test cases? When it has been implemented in many programs in different task domains? Perhaps we should be asking, instead, how we can bound the applicability of methods. Are we obliged to claim generality on inductive grounds (e.g., the program ran in several domains)? We don't say "the sun will rise because it always has," but rather, "here's why the sun will rise." Is there a similar *deductive* notion of generality in AI? (See [4] for further discussion.)

- Discussions of the methodological role of programs. Why should we build programs? This is an important aspect of a broader debate on the empirical AI research cycle, which requires considerable elaboration beyond the sketch in Section 1.

The third recommendation is that editors, program committees, and reviewers should begin to insist on evaluation. Editorial policies rarely *preclude* evaluation—we all think it's a good idea—but neither do they require them. This is one reason that papers almost never describe an entire research cycle—a problem, a task, a view, methods, programs, experiments, and results. Editorial policy is perhaps the most effective way to guide empirical AI practice toward more complete evaluation.[3] It can also encourage the following uncommon kinds of papers:

- Short studies with extant systems. Many of the experiments discussed in Section 1.4, which address how and why systems work, are extensive, self-contained, publishable studies. And even after a system has been thoroughly explored, it may still serve as a testbed for studies with other methods, knowledge representations, control structures, and so on. AI systems take so long to build that we really cannot afford to drop them until we have learned all we can from them.[4]

- Negative results. We need to know when methods don't work as expected, when systems perform *less* well as they become more knowledgeable, when scaling up causes problems, and other negative results. When did you last read an AI paper that said something didn't work?

[3]Lately, the machine learning community has mentioned evaluation criteria explicitly in Calls for Papers, and Langley has advocated evaluation in his editorials [9].

[4]A good model of this kind of work is a book on extensive experiments with MYCIN, edited by Buchanan and Shortliffe [2].

- Progress reports. We should publish progress reports throughout the development of large AI projects, not wait until they are "finished." Lenat's CYC project [11], for example, is slated to continue throughout the 1990's. We cannot afford to wait until then to see how CYC is progressing—especially since it is pioneering techniques and ideas that we will need much sooner.

When we began to write this paper we took the view that science is not legislated, controlled, or forced upon its practitioners, but is a voluntary business in which loose organizations of individuals somehow produce coherent, cumulative progress. We obviously believe evaluation and methodology will help this process; it isn't very hard to do, and it suggests new research, generalizations, and hypotheses that one wouldn't discover otherwise.

# References

[1] Avron Barr and Edward A. Feigenbaum, editors. *The Handbook of Artificial Intelligence, Volume 1.* William Kaufmann, Inc., Los Altos, CA, 1981.

[2] B. G. Buchanan and E. H. Shortliffe. *Rule-Based Expert Systems: The MYCIN Experiments of the Stanford Heuristic Programming Project.* Addison-Wesley, Reading, MA, 1984.

[3] Bruce Buchanan. *Artificial Intelligence as an Experimental Science.* Technical Report KSL 87-03, Knowledge Systems Laboratory, Stanford University, January 1987.

[4] Paul R. Cohen and Adele E. Howe. Is there a method to our madness?: Case studies in evaluation. 1988. To appear in *IEEE Transactions on Systems, Man and Cybernetics.*

[5] L. D. Erman, F. Hayes-Roth, V. R. Lesser, and D. R. Reddy. The HEARSAY-II speech understanding system: Integrating knowledge to resolve uncertainty. *Computing Surveys*, 12:213–253, 1980.

[6] John Gaschnig, Philip Klahr, Harry Pople, Edward H. Shortliffe, and Allan Terry. Evaluation of expert systems: Issues and case studies. In F. Hayes-Roth, D. A. Waterman, and Douglas B. Lenat, editors, *Building Expert Systems*, chapter 8, pages 241–280, Addison-Wesley, 1983.

[7] James R. Geissman and Roger D. Schultz. Verification and validation. *AI Expert*, 3(2):26–33, February 1988.

[8] A. Howe, J.R. Dixon, P.R. Cohen, and M.K. Simmons. Dominic: A Domain-independent program for mechanical engineering design. *International Journal for Artificial Intelligence in Engineering*, 1(1):23–29, July 1986.

[9] Pat Langley. Research papers in machine learning. *Machine Learning*, 2(3):195–198, November 1987.

[10] D.B. Lenat. *AM: An Artificial Intelligence Approach to Discovery in Mathematics as Heuristic Search.* PhD thesis, Department of Computer Science report STAN-CS-76-570, Stanford University, 1976. Doctoral Dissertation.

[11] D.B. Lenat, M. Prakash, and M. Shepherd. CYC: Using common sense knowledge to overcome brittleness and knowledge acquisition bottlenecks. *AI Magazine*, 6(4):65–85, 1986.

[12] T. M. Mitchell. Version spaces: a candidate elimination approach to rule learning. In *IJCAI 5*, pages 305–310, 1977.

[13] A. Newell. A tutorial on speech understanding systems. In D. R. Reddy, editor, *Speech Recognition: Invited Papers Presented at the 1974 IEEE Symposium*, pages 3–54, Academic Press, New York, 1975.

[14] Allen Newell and Herbert A. Simon. Computer science as empirical inquiry: Symbols and search. *Communications of the ACM*, 19(3):113–126, March 1976.

[15] Mark Ferral Orelup. *Meta-Control in Domain-Independent Design by Iterative Redesign.* Master's thesis, Mechanical Engineering Department, University of Massachusetts, September 1987.

[16] Jeff Rothenberg, Jody Paul, Iris Kameny, James R. Kipps, and Marcy Swenson. *Evaluating Expert System Tools: A Framework and Methodology.* Technical Report R-3542-DARPA, Rand Corporation, July 1987.

[17] E. Shortliffe. *Computer-Based Medical Consultations: MYCIN.* American Elsevier: New York, NY, 1976.

[18] D. Waltz. *Generating Semantic Descriptions from Drawings of Scenes with Shadows*, pages 19–92. McGraw-Hill, New York, 1975.