Part Two

Expert Systems: How Far Can They Go?

Randall Davis, Editor

A panel session at the 1985 International Joint Conference on Artificial Intelligence in Los Angeles dealt with the subject of knowledge-based systems; the session was entitled "Expert Systems: How Far Can They Go?" The panelists included Randall Davis (Massachusetts Institute of Technology); Stuart Dreyfus (University of California at Berkeley); Brian Smith (Xerox Palo Alto Research Center); and Terry Winograd (Stanford University), chairman. Part 1 of this article, which appeared in the Spring 1989 issue, began with Winograd's original charge to the panel, followed by lightly edited transcripts of presentations from Winograd and Dreyfus. Part 2 begins with the presentations from Smith and Davis and concludes with the panel discussion. Although almost four years have passed since this discussion took place, the issues raised and the points discussed appear no less relevant today.

Smith's Presentation[4]

Panels are most exciting when the panelists disagree. So you may be hoping that I'll take exception to what the previous speakers have said. Unfortunately, I have bad news to report: I happen to agree with many of the points Terry and Stuart made. I especially agree on the surface—for example, with their claims that expert systems will be best in what Terry called "systematic domains": situations where the world comes laid out with such clean, delimited, unambiguous regularity that it really can be captured in a finite set of axioms or rules. Similarly, I agree with their rough consensus that expert systems will be correspondingly brittle—or will even fail outright—in cases that require lots of common sense or intuitive judgment.

In fact, if you don't listen very closely, you might come away thinking we're all boringly similar.

But let's push a little harder, because there really are differences among us, not only in emphasis but in foundational stance. One way to see this is by looking at how we each compare computers with people. Stuart and Terry both set up a contrast, arguing that expert systems lack certain properties that people have (I've already listed some: common sense, intuitive judgment, an ability to assess the limits of a given set of opinions or beliefs). For my part, I want to focus on ways that people and computers are the same. I want to talk about inherent limits—about what philosophers call "brute facts," that no man, woman, or machine will ever get past.

First, though, some remarks on the use of models.

The Inevitable Model

When we design computer systems, the first thing we do is to select a model of the domain in which the program is going to work. We may not formulate it explicitly (especially early in development), but it will always be there, a kind of abstract scaffolding in terms of which the system represents the world. So, for example, a system to administer drugs would probably model such things as a drug's absorption rate, as, say, proportional to the patient's weight or height. Similarly, Palo Alto's automatic traffic lights are based on a particular model of cars that includes some sense of how much metal there is and how long it takes to drive across the intersection (bicyclists sometimes discover the limits of the former parameter; farm equipment drivers the latter).

A model can be thought of as sitting between the computer system and the real world that the computer system is intended to be about, as suggested in figure 1.

What exactly are models? That's too hard a question even to start answering tonight. But they're certainly common. For example, clay and balsa models of cars are used in wind tunnels to test factors of aerodynamic drag. Blueprints, in their own way, can be viewed as models of houses. These examples are both concrete. Abstract models are more common in academic fields, such as the sorts that are used to model socioeconomic systems, teenager personality traits, the trajectories and collisions of sub-

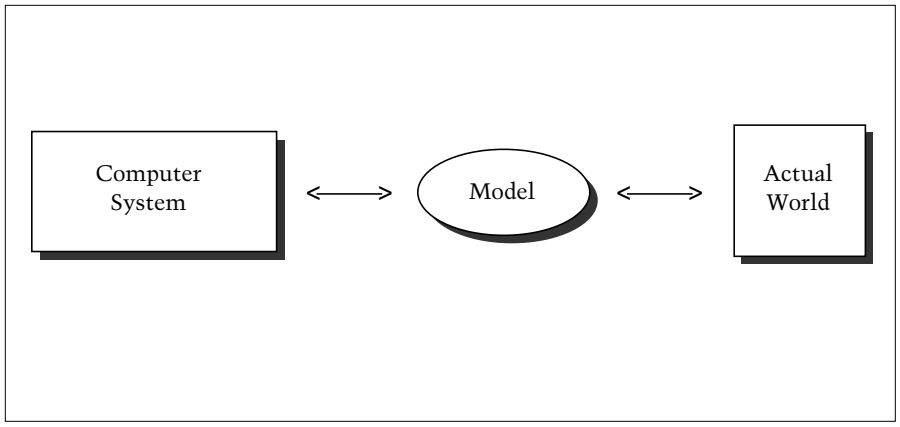

Figure 1. The Mediating Use of Models.

atomic particles, and so forth. Of particular interest to those of us in AI and computer science are set theoretic and abstract models of programs, representation systems, and languages. If you've been to any of the semantics talks this week, you will have seen lots of these in use.

There are some things that all models have in common, even though in other ways they can be very diverse. Let me cite just a few—properties that will be familiar to many of you, since I'll ultimately claim they hold for language and thinking, too. First, models deal with the world at a certain level of abstraction, i.e., to model the world is to conceive of it in a certain delimited way. You identify the objects you are interested in, the properties you care about, the relations between them that matter, etc., and discard the rest; i.e., you "register" the world in terms of a more or less coherent set of concepts, categories, and constraints, paying attention to the details you think are salient and throwing away the things you think you can afford to ignore.

For example, let's go back to the drug model. At least as I imagined it, it ignored the patient's musical tastes. Traffic light models similarly ignore the temperaments of bicyclists and tractor drivers. As these examples suggest, models ignore things at lots of different levels. Though we can hope that the architect thought about it, a hospital blueprint won't model the efficiency of the resulting operating room or the protein structure in the wooden floorboards.

Models have to ignore things; that's how they work. It's a feature, in other words, not a bug. In fact that's what "abstraction" means: it comes from the Latin word for drawing away.

Why do models ignore things? My best one-sentence answer is this: I take it as a fundamental tenet that the world is an infinitely rich, complicated place, and if you (or I, or our computers) were to try to pay attention to everything, we'd simply drown in details. We ignore so we can cope.

Here's a stronger way to put it: models inevitably do a certain amount of violence to their subject matter. You have to do that violence—ignore some of what's going on; gloss over details; categorize together things that are, in the end, different; pretend there are sharper boundaries than are really there—or else you'd be so sensitive that you would be paralyzed, unable to proceed. This kind of violence—arising from the partiality of models—is an inevitable feature (I will claim) not only of programs, but of any kind of language, representation, even thinking. It's an inherent fact: true of computers, and true of us.

Full-Blooded Action

In contrast to modeling, something that's not partial is action. If, instead of modeling it, you actually drive through an intersection in Palo Alto, or inject a drug into a patient's bloodstream, the action you take is not an action in the model. It's an action in the world. This is far and away the most important fact about action. When the robot arm attached to the computer reaches out and actually does something, or if you or someone else acts based on an output of a computer, the action that's taken is not constrained to operate within the particular level of abstraction at which the model is formulated. If you adjust the length of the floorboards in the operating room, you will affect the room's efficiency, and you will affect the protein structure, whether you or the computer think in such terms. In that magic moment when the planning, thinking, formulation, representation, whatever, leads up to the voltage being raised or the signal being sent, and the action is finally and irrevocably taken, then the level of abstraction of the reasoning or language that led to it all falls away, and something infinite happens. We may ignore some aspects of the world, but the world doesn't ignore any aspects of us.

So this is our second brute fact: actions are full-blooded. You don't act in the middle box of figure 1 (though you may, either before or after the fact, construct a model of your action, but that's different). You act on the right.

The Current Status of Theory

Why is this interesting? What does it have to do with expert systems? Several things. The first is that we can use these facts to explain the current state of our theoretical understanding and the state of the art. I said that models intermediate between computer systems and what they're about. Now the general inquiry into what things are about, and into how systems (like language) that are about things relate to those things, is called semantics. Lots of things go under the name semantics, but by far the most important set of intellectual tools for studying semantics is called model theory.

On the face of it, this at least seems like a good name. It suggests that model theory would be so-called because it is a study of models and of how they play this intermediate role between representational systems (like computers and languages) and what those systems are about. But, as so often happens, this entirely natural

interpretation is also entirely wrong. Model theory isn't a theory of models at all! Rather, it's a theory that uses models to study semantics. We—i.e., all of us, all of modern intellectual history—simply don't have a good theory of models (or, in fact, of any similar phenomenon, like language, computation, representation, etc.). It's an astonishing fact, but it's true.

What we do have—what model theory actually is—is a theory of the relationship between so-called formal systems (again: languages, computers, and so on) and the model. In figure 1's terms, it is a theory of the relationship between the left hand side and the middle. That's as far as current theoretical techniques have progressed. It follows, therefore, since actions take place in the world on the right, that we don't have a theory of how well representational systems perform, in the sense of taking actions. And the situation won't change until someone develops an adequate theory of abstraction, modeling, and all the rest.

A strong conclusion, but it's a fact, and a sobering one.

This analysis explains lots of commonly held views. For example, what computer science calls "program correctness" (what's proved in proofs that go by that name) are relations between programs and models, not between programs and the worlds they really operate in. This, in elementary terms, is why programs that have been proved correct can still make mistakes, as any practitioner knows.

(As an aside, I think we should be careful about using the term "correct" in any statements that could reach the general public, since it is so easily interpreted as meaning that a program will provably do the right thing, which is not necessarily true at all. To say that would be to make a claim about the right side of figure 1).

Another thing this analysis explains is what we are good at. As I said at the beginning, Terry and Stuart, in their own ways, pointed out that current computer systems are good in domains that are systematic, about which we have good clean theoretical accounts, etc. That is, real-world systems for which we have good models. Just as we'd expect. Take mathematics, that domain in which our models are, if not perfect, as close to perfect as we're ever likely to see. You'd expect a model theoretic analysis of mathematical systems to be damn good—and indeed they are. Furthermore, mathematics has another property that contributes to the virtual perfection of our analyses: mathematics isn't a domain in which you take actions. (Sure, you can add 2 to 3, but I don't really call that an action. You don't thereby go and stir up the mathematical realm, changing 2 to 2.01. Nor can you, in spite of an infamous state legislature's attempts, legislate pi to be exactly 22/7.) In other words, it's not surprising that mathematics is the domain in which our semantic analyses are the best, and in which computers, more than anywhere else, outstrip their human rivals.

What Then?

I've identified two brute facts (the partiality of models, and the full-bloodedness of action), identified a potential gap between the two, and recognized a limit in our current theoretical understanding of the relationship between models and the world. Where does this leave us with respect to the question set before the panel?

Let's start with the last: the recognition of current limits on theorizing. Can the situation be changed? Would it be possible to have rigorous analyses of the relation between models and the world? This is an enormous question, but there's time for two quick comments. First, with respect to the possibility of intellectual development, the answer is yes: I firmly believe that the model-world (middle-right) relation can in fact be studied, with uncompromised discipline, dispassion, and theoretical integrity. Not only do I think it's possible; for lots of the reasons we've already been through, I believe it's crucial for computer science, semantics, and AI. But it is what marketers would call a "big ticket" item. In particular, it will take us outside the limits of what is called the formal tradition, a consequence that I think will shake our field all the way down to its foundations. But that's ok; formalism has no patent on rigor.

(Another aside. Such a development would have as many benefits as costs. At the moment, some people feel that our understanding of computers can be bisected into two relatively non-overlapping parts. These are the people who believe we should study the left-hand side relation "technically," during the eight hours of the work day, and then spend two hours of "conscience time," in the evening, on the right-hand side relation, as if it were a separate aspect of our intellectual or even political life. However, as I have tried to argue, since computer systems are successful only to the extent that they can coordinate their representations and their actions, it follows from the above analysis that an intellectual framework adequate to the task of understanding how to build a good computer system will have to integrate its analysis of both sides of the diagram.)

In sum, I believe we've got a real piece of intellectual homework in front of us. But enough of that for now. Let's turn to the other issue, about the gap between partial modeling and full-blooded action. I claimed as a fact of life that correctness is inherently unattainable, in general (i.e., in full world domains). People can't be correct; machines can't be correct. Remember, as I said at the outset, that in none of this discussion have I made an important separation between people and machines. We're dealing with inherent, inexorable conditions. God made it this way; no amount of clever programming is going to change it.

So how do people cope with these limitations? Well, to start with, they admit them. In fact it sounds odd even to ask, say, about a prospective American ambassador to Finland, whether he or she will be correct. About a given action the question makes sense (though its answer will typically be debated). About whole human systems, however, we don't even try. What do we do instead? We ask whether people will be reliable.

The situation for computer systems, I predict, will be much the same, and ultimately for the same reasons. As we build systems to deal with increasingly rich, complex, open-ended domains, where our models are

The partiality of models is inevitable not only in programs, but in any kind of language, representation, and even thinking. It's an inherent fact: true of computers and true of us.

less and less sure and exact, we will increasingly need to resort to terms like reliability (and I don't mean this in the technical sense of mean-time-between-failures).

What does reliability involve? Lord only knows, but maybe a few things can be said. One is that reliability seems to involve general intelligence, so that you have the ability to reason about the limits of the particular model that you're using. We need, that is, to reason about such things as: is the model appropriate? is it going to get us into trouble? what kind of trouble? will we be able to cope with the trouble if and when it arises? how complicated will it be? how much time will we have? how serious will it be if we fail? We need to contrast any given perspective or model with others, compare it with other situations we've seen before (cf. Stuart's emphasis on recognition), etc.

Let's take a concrete, and dramatic, case in point. Some of you may remember that in 1960 there was an indication, in the recently-installed early warning system in Greenland, of a massive impending Soviet missile attack. It was an error, of course; it turned out that the system's radar signals had bounced back off the moon. How did this happen? It turned out that moonrises hadn't been been thought of by the designers, so they weren't in the system's model.

That much is obvious and fits our earlier analysis. But let's look hard at what happened next—i.e., at what the people in charge did. Their first thought was to contact Washington, but as it happened an iceberg had cut the telegraph cable. So that was out. Then they started thinking about the situation. They obviously knew they didn't have much time. Various factors suggested that something might be awry. For one thing, the whole system was new. In addition, they realized that Kruschev happened to be in New York, and it seemed unlikely that the Soviets would have chosen such a time for an all-out attack. To this date I doubt whether the itineraries of Soviet dignitaries have been incorporated into our launch-on-warning radar systems. You're never going to get it all.

What's going on here, I am suggesting, is that people deal with the limitations of any and all particular models in two ways. First, they exercise judgment, common sense, and intuition. It is clear that one of the reasons we deploy general intelligence is as a way of comparing and contrasting different models, recognizing the limits of any given one, contrasting the situation to others that are similar in lots of different ways—all with the aim of circumscribing the situation from different sides, knowing full well that no unique analysis will be necessarily correct. If, as I suggested earlier, all thinking involves modeling, then this is clearly no panacea; even the best intuitions won't give you a perfect view on the world. But these levels of models, comparisons with experience, recognition of limits, analogy and contrast, and so on, can at least smooth the edges of any given model's limitations. And of course these are exactly the kinds of rational abilities expert systems almost completely lack.

Second, people recognize that in spite of their best efforts their reasoning will be inadequate, so they include all sorts of habits for dealing with the resulting problems. Even in the mostly wildly optimistic future one can imagine for AI, computer systems will be limited in the same way. At the moment, we haven't got more than the most rudimentary clues about how to create general intelligence, circumstantial appreciation, etc., let alone incorporate the character and moral judgment required to cope with its limits. Again, expert systems are particularly poor in this regard, even against the meager backdrop of the current state of the art. And—and this is where the inherent limits I've been talking about take hold—the best the future can offer is a kind of moderate reliability. The exactness and precision we associate with machines exactly won't generalize to the sorts of full-blooded, real-world situations where reliability is the only serious measure.

The moral I want to leave you with, in other words, is this. An inexorable tension separates the precision and etched boundaries of theoretical models, on the one hand, and the ultimate richness of the world, on the other. We, and every system we ever build, are just going to have to live with that fact.

Davis's Presentation

There are three things I'd like to accomplish with my remarks:

- First, I want to be clear about the focus and the scope of our topic here,
- Second, I want to comment on some of the issues raised in what the others have said,
- Third, I want to respond to the original challenge Terry set out in the description of this panel, explaining my view of the technology, its strengths and weaknesses, and its most appropriate role.

Part I: What We Are About

We're here to discuss knowledge-based systems, not AI in general, and the difference is significant. As we have noted since the technology first emerged from the lab, tasks relying crucially on perception, manipulation, or common sense are beyond the current scope of this art. We have emphasized that the tasks should involve symbolic cognitive reasoning of the sort often required in fields involving specialized training.

The question of the ultimate ability of machine intelligence to capture all aspects of human cognition is intriguing, but not on our agenda here tonight. We are here to consider a much more circumscribed task.

We're focused then on expert systems, not AI in general, but even so the name is problematic. Because it's just that, a name and not a description. We handle those two things very differently. For some descriptions we obtain meaning by a kind of intersection process—a "light beer" is a beer that's light—but names are just pointers: the meaning of "light year" can only be obtained by examining the concept it points to.

So it is with expert systems. They are not "systems that are expert." They are, rather, a commitment to a certain system architecture, explicitness of knowledge encoding, transparency, availability of explanations, and so forth. The "expert" part of the name is an aspiration, not a necessity, and much good will accrue from them even if we never had built one that was truly expert at its task. For that reason "knowledge-based system" is a much better, more technically relevant term (and the one I will use wherever possible henceforth).

We keep doing this to ourselves, by the way. The term "artificial intelligence" has brought us trouble almost from the beginning and now "expert systems" produces more difficulties. The field should have been called something dry and unevocative, like "complex information processing," as once was the case at CMU, and these programs might have been called "domain specific systems," a term that was once used at MIT. (Note that it's the Stanford people who keep getting us in trouble.) But the point is serious and we should take note: next time we label a technological advance, let's be careful for a change.

But since the issue arises, we may ask what it would mean to create an expert system that was worthy of a literal interpretation of the term. What depth and breadth of expertise would satisfy us? I have several comments.

First: as with many things the answer is not precise, but we can turn to human experts for examples and human performance for an important calibration. There are people we are willing to label experts, so we'll take those as working examples. In doing so we should take careful note that these acknowledged examples have many sorts of limits, in both their depth and breadth of skill. The expert chemist may be a mediocre mathematician and terrible at chess, but he is no less an expert chemist for it. Human experts also display tunnel vision—it's well known in medicine that not only the treatment but the diagnosis of a patient often depends significantly on the area of specialization of the examining physician. These well-recognized phenomena do not prevent us from referring to these people as experts, hence the mere existence of limits of depth and breadth and the presence of tunnel vision in our programs should not disqualify them from deserving the term.

Second, it is important to note that we are talking about creating a system that is the "same" as an expert to within some level of abstraction, not equivalent in all conceivable details. Whether we should use the term "expert" for someone (or some thing) who goes about getting an answer by reasoning carefully through every detail depends on how deeply we want equivalence. And in this case we are talking about equivalence of reasoning outcome, not identical reasoning processes. Programs can match the expert at the I/O level; they need not be more detailed analogues.

Third, is process crucial to outcome? Can one be an expert narrowly construed or does real expertise only manifest itself narrowly, yet require something coextensive with human thought as a foundation? As I will argue, we have counter-examples that demonstrate this is not true. It is in any case a question for empirical study, not a matter for pronouncement and categorical claims.

Fourth, to what degree and on what account do we come to trust human experts? How do we know they understand? We typically explore the understanding with a test that examines a limited sample of their knowledge, and extrapolate, saying that they understand, meaning something more than that they can do exactly the problems chosen for the exam. Why do we believe that? If a program passed the same test, would we be willing to say it understood? I believe the answer is no, but why not? If we can understand why not, perhaps we will know how to begin to fix the problem.

Fifth, it is important to keep in mind the limited sense in which humans can be said to understand. On this let me quote an unlikely ally, Joe Weizenbaum:

> It is too much to insist that a machine understands a sentence only if that sentence invokes the same imagery in the machine as was present in the speaker of the sentence. . . . For by that criterion no human understands another human. Since, in the last analysis, each of our lives is unique, there is a limit to what we can bring another person to understand. Yet we agree that humans do understand one another to within acceptable tolerances. The operative word is "acceptable," for it implies purpose. When therefore we speak of a machine understanding, we must mean understanding as limited by some objective.

And, I would add, echoing Joe, when we speak of human understanding, we mean limited by some objective. Let us not therefore commit the superhuman-human fallacy, expecting any more of our systems than we expect of ourselves.

Sixth, it has been widely recognized in the field that the systems we have built so far perform quite differently than human experts. We focus on performance, and as I pointed out at this conference in 1981, that's clearly only one dimension of human expert behavior, which also includes learning, breaking rules, reformulating, graceful degradation, etc. The other dimensions have not gone unnoticed: the question of whether and by what machinery we may be able to attain them is, once again, a question for empirical enquiry.

One final point: we should recognize that the rule-based approach —what Terry has called "narrow articulation"—is only one form of knowledge-based system (despite the sneaky attempts of others on the panel to

make them appear to be equivalent terms). It is one variety of knowledge encoding and system architecture, with a set of strengths and limitations that have been well recognized in the field, made explicit in the earliest papers on the subject, and reviewed in some detail in my 1981 talk.

Part II: Comments on the Other Positions

I have much to say in response to the positions of the others on the panel, but for the sake of time I will focus on Stuart's approach, since his is the best elaborated, and respond only very briefly to Terry and Brian. I find relatively little to disagree with in what Brian and Terry have said.

I found Stuart's model of human expertise quite interesting, intuitively appealing, and descriptive of experiences I have had. It will be interesting to see it evolve and determine what consequences it has.

Unfortunately, I believe it has in its current form some significant drawbacks. Most centrally, I claim that the model is neither necessary to nor sufficient for the task we are about, and additionally, that even if it is valid, there is a pragmatic sense in which it is irrelevant.

I find the model appealing because it describes a familiar sensation. I play tennis, and after years of practice, I've had some of the experiences he mentions, proceeding from studied and awkward play, to smoother more instinctive reactions, sometimes doing things I wasn't even aware of until watching a videotape later. Occasionally, but rarely, I've had that remarkable sensation of total involvement in a closely fought match where I could momentarily see nothing but the ball and yet reacted intuitively to the motion of both the ball and my opponent. In Stuart's sense of the term, then, I believe I play tennis unconsciously, I play intuitively, I play holistically and . . . I play rather badly. And much the same can be said for my bridge and chess games.

You see, whatever this model is about (fluidity of execution, effects of practice, compilation, or whatever), it's insufficient and missing an essential ingredient: getting the right outcome. Its focus is on performance, with little or no attention to competence (in the informal sense of both of those terms). It's partly on target, since in human terms at least, fluidity of execution is certainly a hallmark of an expert. But it's clearly insufficient, except in some rather impoverished sense. One might say perhaps that in playing tennis the thing I am an expert at is playing Randy Davis' tennis game. That I do fluidly, instinctively, and so forth.

In contrast to modeling, something that's not partial is action. Actions are not constrained to operate within the particular level of abstraction at which the model is formulated. Actions are full-blooded.

But there's where we part company, because I would want part of the definition of expertise to be accountable to some acknowledged standard of competence, of getting the right outcome. To me, the interesting difference between an amateur and an expert on cognitive tasks like bridge, chess, chemistry, etc. is knowledge. The source of an expert's abilities on those sorts of tasks is a difference in what he or she knows, and that has been a fundamental part of the motivation for the design of knowledge-based systems.

What about its necessity? Do you need to be intuitive to display expert-level competence? Are there systems that we would call expert, yet that demonstrably do not reason intuitively or holistically?

I believe that in restricted domains the answer is clearly "yes." To cite three famous examples: Dendral, Macsyma, and Xcon all perform at levels matching those of human experts in their respective domains. Given the existence of three counter-examples, we should be unwilling to believe that the model, interesting as it is, is even a necessary part of expertise.

It would be difficult to argue that any of these systems is performing holistically or intuitively, yet each is performing at and sometimes above the competence level of acknowledged human experts in a narrowly defined task area.

Thus I claim that it is possible to be an expert and not reason intuitively. These programs certainly will not be identical to human experts; their errors in particular are unlike the errors human experts make; we have noted already that they are neither intuitive nor holistic, and perhaps more importantly, they do not degrade gracefully as we approach the boundaries of their competence.

But again, we're talking about equivalence limited by some specific objective, in this case competence, and by that standard at least, we have succeeded.

Thus the Dreyfus five-step model seems neither necessary nor sufficient.

There are also other models suggesting what happens as a result of extensive experience. While the analogy is vague, it may indeed be the case that there is something like compilation going on here, with its attendant increases in execution speed and difficulty in recalling the original reasoning.

Perhaps Stuart is arguing for a particular problem-solving paradigm, roughly problem solving by recognition, of somewhat the sort that

inspired early work on both production systems and the Planner family of languages.

Perhaps, but this, too, is insufficient because human experts are also good at solving relatively unfamiliar problems. To demonstrate, let me show you an example I normally use in class to show some of the important limitations of the rule-based approach, but it does well here too.

One of my worst dreams is that somebody will try to build a rule-based system to control a nuclear power plant. I have here the "Four Mile Island" system, with the following three rules:

IF the red light goes on and the reactor temperature is above 1500,

THEN open valve #3.

IF the blue light goes on and the reactor temperature is above 1500,

THEN open valve #41 (quickly!).

IF the green light goes on and the reactor temperature is above 1500,

THEN close valve #41 (immediately!).

And then our expert, who is doing problem solving by recognition, finds himself in the situation where the temperature is over 1500, and both the blue and the green lights go on. Now, what should the expert's intuitions tell him to do? My intuition would be to get the hell out of there, and surely there is something more at work here than the matching of previous situations.

Some of you are no doubt saying, if we knew the pattern of interconnections of pipes and valves, then I could figure out what to do. Yes, that's just the point. To turn the tables on Stuart for a moment, I would say that expertise is much more than just some sort of matching of previous situations, but must also include some of what has been called reasoning from first principles.

Stuart's approach raises one other question that I think is worth some consideration. It may be, as our existing systems suggest, possible to be expert without being intuitive. But it may also be a fact of life that some human experts do reason intuitively, even for well-bounded problems. If so, that has some interesting consequences. It may be that such people are necessarily inarticulate, that is, that no amount of debriefing, no set of rules, indeed, possibly no representation language at all will make it possible to capture that expertise, because the very act of trying to express it aloud may distort it.

There is some anecdotal evidence consistent with this since it is hard to get experts to express the rules underlying their decision making.

Given the early stage of development of our field, we should also be wary of asking them to talk a particular language. It may well be that our current knowledge representation languages are inadequate for the task. Given the length of time we have been at this, it would be striking indeed if our current languages turned out to be adequate. And we should also be wary of introspection as a vehicle: the experts may well report what we want to hear, not what they know.

This, too, is a legitimate question to be approached in the spirit of empirical enquiry. It is not a matter for pronouncement and categorical claim.

Now, from a purely pragmatic viewpoint, the model seems to me irrelevant. While the model suggests several things in the progression from stage III to stage IV, like a reduction in cognitive load, faster and more fluid performance, etc., it is not obvious that it suggests any difference in competence, the correctness of the outcome. So even if Stuart is correct and systems never make it past his stage III, it doesn't matter, as long as we're talking about equivalence limited by a specific objective, the equivalence of outcome.

And to be purely pragmatic about it, even if there is some important difference in outcome that occurs at stage IV, there will surely be much benefit that can result even from stage III systems.

Finally: let's recall that all this concerns the capability of existing, i.e., rule-based systems. One of the nice things about this part of AI work is that it has a pragmatic, engineering orientation and is willing to consider all reasonable tools. If there is some unavoidable and crucial shortcoming in the rule-based representation technology, fine, let's understand what it is and press on to develop a better one.

Part III: Using the Technology Rather than Being Used By It

Let me now respond to Terry's original charge to the panel. I will begin here as I did in my proceedings statement, taking his question literally and repeating a bit of heavy-handed satire I wrote there.[5]

He said:

> We are in the midst of a great wave of enthusiasm about the potential for expert systems in every area of human life and work. There is no agreement, however, as to just how much they can do and where they will run into fundamental limits.
>
> To what extent can we count on rule-based systems for "flexibility" in dealing with unexpected situations? How reliable will such systems be in cases where programmers did not anticipate significant possibilities?
>
> How can a knowledge base be subjected to standards of accountability? Who is responsible for what an expert system contains and what it does?

I replied:

> In reading a newspaper recently I was struck by the profusion of sometimes conflicting expert advice available on subjects as wide-ranging as the economy, political events, health, and financial planning. All of that made me begin to wonder:
>
> Experts: How far can they go?
>
> We are in the midst of a great wave of enthusiasm about the potential for experts in every area of human life and work. There is no agreement, however, as to just how much they can do and where they will run into fundamental limits.
>
> To what extent can we count on carbon-based systems for "flexibility" in dealing with unexpected situations? How reliable will such systems be in cases where their teachers did

not anticipate significant possibilities?

How can a person's knowledge be subjected to standards of accountability? Who is responsible for what an expert contains and what that person does?

I had three reasons to start out this way. First, I wanted to demystify and personalize the task at hand, as a way of heading off technological determinism. The question is not what can **they** do, as if the technology had some inherent power of its own, some inevitable course of development. It is, I claim, a mirror of sorts that reflects both our ability to elucidate our own reasoning and our decisions about how to use it. The direction and consequences are still up to us to determine.

Second, putting it this way sets us out on what I believe is the appropriate direction, with the appropriate priorities. The core question is not "What can knowledge-based systems do?" It is, instead, first, "What do **we** know?" and second, "How easily can we express it?" How well do we understand what makes our own reasoning flexible, for example? And how easy is it to express that knowledge in our current representation and reasoning technologies? The questions we are asking are first about knowledge and only secondarily about technology. Both matter, but the order is important.

Third, while the satire may be heavy-handed, the comparison with people is valid and provides an interesting way to proceed. As I noted earlier, it helps to ask why we have any confidence in our belief that human experts understand. We may not want to use the same criteria for assessing whether a program "understands," but even the differences will be revealing.

We explore people's understanding of a subject with a test that examines only a limited sample of their knowledge, then extrapolate, saying that people who pass "understand" the material. By that we mean they can do more than exactly the problems chosen for the exam. If a program passed the same exam, would we be willing to do the same extrapolation? If not, why not? If we can determine what it is that makes us hesitate in the case of the program we have the beginnings of an intriguing research agenda.

Finally, the satire helps us approach an important question: Given the well recognized limits to the scope of expertise we can capture, how best can we use the technology?

To begin to answer these questions, we need to recognize an important fact of life for AI and knowledge-based systems: they typically tackle incompletely understood tasks. As Marvin Minsky once put it, "Intelligence is that which we admire but do not understand." We consistently seek out and work on tasks and modes of reasoning that are not well understood.

But if it is the fate of AI and knowledge-based systems to deal with incompletely understood tasks, we have at least developed a set of techniques for coping with that difficulty. And that, I believe, is an important point: it has proven to be a useful set of tools for dealing with incompletely understood ideas.

This arises in part because the architecture of knowledge-based systems makes them incremental and transparent. We don't have to be able to specify every part of the decision-making process. We can instead write down as much as we think we understand, try that out, and if it fails, determine the shortcoming and add additional knowledge. In part it's the standard mental hygiene that arises from having any computer model: the speedy, literal-minded application of ideas provides a strong test and often makes their consequences clear. But it is also the ability to revise and augment that model. This is experience, not speculation. The builders of Xcon, for instance, report that this ability to add knowledge incrementally and easily was an important part of why they succeeded, where previous efforts at automation had failed.

I believe it arises also because even our current, fledgling knowledge-based systems technology provides a set of tools for dealing with ideas, not code, allowing us to proceed at what Newell has called "the knowledge level." This is in many ways a natural part of a long-standing progression toward ever higher-level languages. We have gone from using assembly language to tell the machine what to do, to pattern-directed languages like Planner to tell the machine what we want done, to rule-based systems and logic programming languages to tell the machine what to know. Along with that progression comes the ability to suppress uninteresting detail (like the use of registers inside the machine) and the ability to focus on the important aspects of the knowledge.

So I would claim that knowledge-based systems are a good medium for dealing with incompletely understood tasks and reasoning. Of course they have no exclusive license on this: paper is a good medium for expressing incompletely understood ideas. Each is appropriate at a different stage in the development of the knowledge and each has its own virtues and drawbacks.

This then is one of the central points I want to make. I suggest that for all the pragmatic good they may do as experts, colleagues or assistants, these programs may in the long run turn out to be much more important for their role as a kind of accelerator of knowledge accumulation and theory development in specific areas of study.

This too is experience, not speculation. In mathematics we have had the wonderful progression in the state of the art of symbolic integration: it began as a somewhat mysterious, intuitive art (no doubt the practitioners described it then, perhaps with good reason, as a holistic judgmental skill accrued through vast experience). Then the CRC tables began to accumulate some useful rules (demonstrating the useful role of paper as a technology); then began the sequence of programs from Saint to Sin to Macsyma, leading eventually to the Risch algorithm. And it was the accumulation and testing of knowledge along that route, assisted in part by AI technology, that helped intuition mature into theory.

Similarly, the developers of Prospector have reported the expert's satisfaction with the geologic model laid out explicitly for the first time in Prospector's semantic net-like representation.

And yet again: an editorial in the *New England Journal of Medicine* suggested that medical students would learn diagnosis faster and better if it were presented as hypothesis and test, not some arcane art.

We should encourage this phenomenon and use it explicitly: the programs we write will probably not be the ultimately important thing, though extensive pragmatic benefit may result from them. Perhaps more important will be the role of system construction in encouraging the accumulation, organization, systematization, and further development of knowledge about a task.

But we sometimes won't wait until that development is complete before using the program. And that means that the program is going to be used much of its life for incompletely understood tasks. This is my second major point: given that reality, how can we gain the benefits from the technology while avoiding the pitfalls?

I have several suggestions.

First, consider the heuristic that says,

> if the best you can do is a rule-based system, you don't understand.

If we really have no better explanation of a phenomenon than a case by case listing of unconnected examples, I claim we don't understand, in particular because we have no way of knowing how to handle any case that isn't matched by a rule. Recall the production system example I used earlier. There I was arguing for the inadequacy of Stuart's appeal to problem solving by recognition, but it also demonstrates the potential inadequacy of a set of situation-action rules as a basis for understanding.

Now this categorical statement is not totally true, but that's why I labeled it a heuristic. There are some things that appear to be little more than a large collection of special cases. Simon's work on skill in chess, for instance, suggested that the expertise of master players arises simply from their ability to recognize between 10,000 and 100,000 patterns. But we get considerable mileage out of proceeding as if the heuristic above were true.

Second, as a consequence of the program's lack of complete understanding, we have to recognize the inevitability of mistakes.

Any incomplete understanding (in a program or a person) may encounter a case beyond its abilities and not even be aware of that, producing a plausible sounding but incorrect answer. Given the current state of our art, I would be willing to bet that every system we build in the next ten years will fail while in real use, despite our efforts to debug and test it.

You see, whatever this model is about (fluidity of execution, effects of practice, compilation, or whatever), it's insufficient and missing an essential ingredient: getting the right outcome.

If you need to be convinced, consider Xcon, with 80,000 examples run (as of August 1985) and still its knowledge is incomplete. The errors are infrequent, but inevitable, as in any incompletely understood task. Now ask yourself: How many other AI programs have we ever built and tested on 80,000 cases?[6]

So these systems are guaranteed to fail. That's a sobering note. But it need not be a disabling one, which brings me to my third point:

Given the inevitability of failure, don't bet the ranch on what you don't entirely understand.

The implication here is on task selection: choose problems for which the system has the freedom to fail. More generally, the consequences of failure should be inversely proportional to the likelihood of failure (as with all things we do). Xcon, for instance, currently has roughly a 2% chance of failure, but even then the consequence is inconvenience and perhaps a small monetary cost, things we can well afford.

Fourth, we have an important educational task ahead of us.

Computers have for too long had the image of infallibility to the general public, in part I believe because they have typically been used in the world at large for tasks that are well understood, i.e., algorithmic. If we are in fact going to use them in the mode I suggest, then we have to make it well understood that the fast and precise application of knowledge does not make a partial understanding any more powerful. Again, previous technologies have had to deal with similar problems: things printed on paper often have a credibility quite out of proportion to what their content would support.

Fifth, we should choose our problems wisely.

As many of us in the field have suggested, we should select symbolic reasoning tasks in narrowly defined domains, at the right stage of formalization, i.e., where we have available a large collection of case studies, but as yet no complete theory.

Sixth: there are additional criteria for problem selection and system design that apply to any incompletely understood task and hence are useful here.

Start by building an assistant, not an expert. It keeps the lines of responsibility clear and it matches people's intuitive models: unlike experts, assistants are expected to make mistakes occasionally.

Choose problems for which the answer may be difficult to generate, but easy to test via an independent route. Macsyma and Dendral are standard examples: symbolic integration, for instance, is difficult to do but simple to check via differentiation.

Select applications that are not time critical. It's always nice to have time for someone to decide what to do about the program's answer.

Select applications whose sub-parts are loosely coupled. Job shop planning

in manufacturing is relatively loosely coupled: if some components get made faster than we expect, they can often be set aside and stockpiled. Chemical processing plants can be problematic: some reaction products cannot simply be set aside to wait indefinitely.

Sometimes there are applications that are time-critical, dangerous, and so on, yet we still want to build a system because we really do want to accumulate the necessary knowledge. In that case, I suggest we build a training system to be used off-line, not a performance program used in the application. Many of the same problems, particularly knowledge elucidation and organization, will still occur. And that's the point: that provides a safer environment in which to discover exactly how much there really is to know about the task.

Finally: keep people in the loop. The systems do have circumscribed domains and having someone there to be sure we're in the right ballpark is sensible.

Let me summarize then:

- In dealing with knowledge-based systems we are talking about understanding as limited by a narrow defined objective.
- We are talking about equivalence with respect to outcome, not necessarily with respect to process.
- There appears to be no necessary connection between outcome and process.
- The model of expertise proposed by Stuart Dreyfus focuses on performance, apparently to the exclusion of competence, and admits of three counterexamples. Claims of impossibility, or even extreme implausibility, based on it thus appear to be unsupported.
- Stuart's model of "intuition" appears to be neither necessary nor sufficient for intelligence. If this is true, then importantly, it offers no predictive power. In particular, if it is not necessary to intelligence, any failure to create programs that behave "intuitively" has no consequence whatever for the goals of AI. Simply put, if intuitive behavior is not necessary, then any failure to create it is irrelevant. Claims of impossibility or even extreme implausibility of AI based on it are simply unsupported.
- Some human expertise may be necessarily inarticulate; that's an interesting question for empirical enquiry, not categorical pronouncement.
- As a result of their transparency and incremental character, knowledge-based systems provide a useful technology for dealing with incompletely understood ideas and tasks.
- It may be that the most important result is not the program, but the knowledge base that is constructed; it may be that the most important contribution of this technology is its enhancement of knowledge accumulation and theory development.
- In using them as such, we inevitably encounter the problem of putting into service programs based on incomplete understandings. There are a number of steps we can take to reduce the dangers in doing so while still reaping the benefits.
- Finally, the question of whether we will ever create a knowledge-based system in the literal interpretation of that term seems to me to be misstated. The question is, instead, to what degree of detail and for what purpose do we require the program to match human performance.

Just as people are experts within limits, so too might our programs be. That is a question for empirical enquiry, not pronouncement and it is in that spirit that I suggest we proceed.

We have to make it well understood that the fast and precise application of knowledge does not make a partial understanding any more powerful.

Panel Discussion

Dreyfus: I certainly can't respond in the limited time we have to everything Randy accused me of, maybe because I can't write as fast as he can talk. Let me respond to one or two points that I managed to get written down. First, his discussion of his tennis play which he described as being intuitive and yet not being good tennis. As I said when I introduced my model, there are many provisos that go along with it that time prevented my stating precisely and probably many I don't state even in the book.

The point of the model is only that the best advanced beginner performers (i.e., those using aspects as well as rules and context-free features) will perform better than the best novices do. The best competent performers will perform better than the best advanced beginner performers. In other words, people using plans and using them well will perform better than people who don't have plans. The best intuitive performers, to lump the last two stages, will perform (in domains where intuitive performance is appropriate) better than the best people doing competent planning.

As I said in the book, you can see all situations in your domain as similar. You can experience one situation and then you can jump to the intuitive conclusion that everything that goes on in my domain is similar and respond to everything the same way. That would be intuitive response and that would be stupid. I certainly don't want to claim that seeing everything that goes on in your domain of expertise as all the same case and then responding always the same way is good. So I don't think that the fact that Randy is an intuitive and poor tennis player undercuts anything I said. If the best tennis player in the world were an analytical tennis player that would certainly undercut something I have claimed.

Concerning the question of necessity and sufficiency of my view, I don't think I've claimed either that intuition is necessary or sufficient for the most highly skilled performance. Intuition is not necessary if there are first principles from which one can reason.

He talked about that, he talked about reasoning from first principles. What are the first principles of recognizing faces? What are the first principles of driving? What are the first principles of recognizing positions in chess? What are the first principles of business? Those are subjects where there are no first principles: areas which are not structured, the kind of area where you can not use a scientific logical fact-based approach. In areas which do not admit to that, then my claim is that intuition is necessary for the highest level of performance. Intuition is also by itself not sufficient for the highest level of performance, I didn't claim it was. Intuition accompanied by deliberative rationality, accompanied by tests to check against tunnel vision, accompanied by a lot of other things, is certainly better than pure intuitive response.

I don't want to claim or be heard as claiming that pure intuitive response (while I think it is what basically characterizes superior performance) is either necessary in all domains or sufficient in any domain for the best possible performance.

Smith: I want to say three things quickly. First, I'm really concerned with people's tendency to assess the limitations of computer systems solely with respect to the competence of people. We already ask machines to do things that people don't do and I'm glad about that. When an airplane lands in the fog, for example, I'm actually glad there's a computer system doing it and not a person, because it seems to me that airplane landing is a "systematic domain" (to use Terry's term) and I think for those sorts of things computers are actually better than people.

The problem is that, since we ask systems to do things that people don't do, which are nonetheless doable, I'm afraid we're going to ask them to do things that we wouldn't ask people to do, because they cannot be done at all. For example: decide whether to annihilate Europe based on 16 seconds worth of warning. I would be bloody worried if a computer system were asked to do that and bloody worried if a person were asked to do it too. So when you complain about a computer's not being reliable and someone replies, "Well, would you rather have a person doing it?" I often answer, "No, I don't want a person doing it either; I don't believe it can be done reliably at all, by man or machine." We shouldn't proceed blithely based simply on the belief that a computer will be no worse than a person.

Second: about intuition. I'm terribly interested in intuitive feel and all this kind of stuff. But it seems to me that this is irrelevant at the moment. It seems to me that intuitions about the phenomenal feel of what it is for a system to operate are way beyond the state of the art. I have absolutely no idea of what the current systems feel like. If being inarticulate is a sufficient condition for being intuitive, I could sell you a lot of intuitive computer systems today. Just because it's extremely inarticulate is not a reason to think it's intuitive.

I think the real condition is whether the logic is based on recognition or analysis. I think that's a real distinction which is terribly interesting. But I think we should set aside the phenomenal feel of the system that does it one way or the other.

Third: we should find a word for the domains for which we think current expert systems technology is adequate. Terry used the word "systematic domain" and it seems to me that a good outcome of this panel would be for the community to have at least a word—even if we don't yet have a definition—a word to label certain domains that have the kinds of structures that allow this technology to be appropriate. I would actually recommend we take "systematic domain" and use that in our literature and to the public. I think that would be a real accomplishment.

Winograd: In the way I originally intended using that term in the book, a "systematic domain" is not a real world domain. A systematic domain, in the way that I talk, is the middle of Brian's diagram, it is the thing which has been represented in a systematic formal way, which then is set in correspondence with something in the world.

So arithmetic, for example, is a systematic domain which happens to be set in correspondence with important things in the world like counting your sheep and doing your taxes and so on. And that one happens to be a very good fit. The thing you can systematize and what you want to do with it in the world correspond well enough that you aren't worried about that. But if on the other hand the thing you worry about in the world is the health of people and the systematic domain is a list of disease-related categories, then you're in trouble. Then you're facing all these problems Brian was talking about with these correspondences. If we introduce the term we can debate exactly how you want to use it.

Smith: The problem is that numbers and sets are perfectly real world domains for mathematicians. So I don't think the notion of "real world" is going to do it. Computers are in the world too.

Winograd: I guess since you said you liked my term of systematic domain, I like your focus on the thing that makes it real world being the open-endedness of action. You don't take action in the world of numbers. You take action in the world of patients and missiles and whatever, but not in the world of numbers. And that's what makes it "real world."

I have a number of comments on Randy's remarks so let me just pick a couple of ones I thought were interesting. One thing which is important to clarify is whether we really all agree on the use of this word "rule-based" and what it means. I think that the three of us on this end of the group (Winograd, Dreyfus, Smith) were at least implicitly lumping together a variety of things. I noted various words. Rule-based is one of them; Brian talked about models, you might call those model-based systems. I talked about a rationalistic approach. Stuart talked about analytical reasoning. He also said "rule-based or you might call it logic-based." What's important is that there is a category, which includes all of those things, that has a certain set of problems and a certain set of benefits. The current version of rule-based, meaning things like if X, then Y and you put lots of them in and let them fire in

The core question is not "What can knowledge-based systems do?" as if the technology had some inherent power of its own. It is instead, first, "What do we know?" and second, "How easily can we express it?" Both matter, but the order is important.

some order, is, I claim, a somewhat superficial peculiarity of today's technology.

Indeed there are many better ways you can do certain things, which have advantages and disadvantages, but I think arguing about "rule-based" versus causal reasoning or millions of other kinds of things is at the wrong level. There's a lower level and a deeper level at which you want to say "systems which are based on analytical rationalized models for which you have explicit ways of manipulating the models," whether those manipulations be resolution theorem provers or causal reasoning with backtracking or if-then rules. And I think the questions apply to the whole range of things.

The other thing I think is interesting is that Randy gave a perfect example of the kind of point I would use in arguing with him. With his Four Mile Island system, he said "If I saw that, I'd get the hell out."

Anatole Holt, who has been a perceptive observer of AI for over three decades, once described a perfect example of artificial intelligence as a machine that was choosing a perfect chess move while the room was on fire. What human beings have because of not being programmed within a particular restricted domain, is the ability to say, "Wait a minute, this isn't right, I'm getting the hell out of here. I'm not going to try and see if it's this rule or that rule." It's precisely that ability to back off from a domain as initially perceived and treat it as part of a larger domain that distinguishes the sort of narrowly focused expert systems that we now use from what you might call a real expert who carries with him/her that whole ability.

Davis: But note that the Four Mile Island example cuts both ways. It is to be sure an indictment of the purely rule-based approach to reasoning, which says if I've seen this before and can write down the previous circumstances I've seen, then I know what to do now. But note that the very same situation also demonstrates limitations in the kind of problem-solving by recognition that Stuart seemed to be talking about under the label "intuition." If an expert "intuitively" recognizes what to do, those intuitions are going to fail in any situation sufficiently different from ones the expert has previously encountered, in just the same way the rule-based approach will fail.

Let me address a question to Terry: you started off talking about rule-based systems, then mentioned logic machines, then causal reasoning. Where does that circle end? It seems to me that it goes arbitrarily far, including almost any computational mechanism. I'm interested to know what wouldn't be one of those things that falls inside this circle of "analytic rationalized models."

Winograd: I think there's a fairly sharp division in approaches to AI, which the name connectionism is thrown around in connection with. That is, there's one approach which says the job of the knowledge engineer or the programmer is to articulate the rules. You're elucidating how things fit together. The alternative is an approach that says, we're going to build a bunch of elements and toss them together and then throw a bunch of stuff at them and somehow they'll connect themselves in a way which does the right thing.

I would say that that approach is not subject to the limitations that we're talking about here. I think it's also highly unlikely that it'll work in anything like it's current form.

Davis: Someone on the panel pointed out earlier the futility of the approach sometimes used in rule-based systems that assumes we can "throw more rules into a pot and the right result will emerge." Doesn't just the same criticism apply here?

Winograd: I think the focus on rule-based systems, this notion that you toss in lots of separate unrelated things, is a kind of intuitive yearning for a system where you don't really have to fit the whole thing together. I think the kind of underlying sense that makes people like that is the same that makes them like connectionism.

Smith: I agree with Randy on the fact that I don't think there's any natural boundary to the rationalistic.

Dreyfus: If I understand Brian correctly I think he said that modeling had to do with abstraction and then he said that to model underlies thinking. I would 100% disagree with that. One thing that underlies some thinking is the ability to abstract and make models. But I think when you drive a car or you recognize a face, there is no level of abstraction. So I don't think everything, or indeed very much, that we would call intelligent behavior in the real world involves the necessity of making models and abstractions and the connectionist's approach is trying to capture some of the ability that people have without making models.

Notes

4. These remarks were based in part on Brian Smith's "Limits of Correctness in Computers," Fifth International Conference of the International Physicians for the Prevention of Nuclear War, Budapest, 1985. Reprinted in *SIGCAS Newsletter,* December 1985. Also available as CSLI Technical Report, CSLI-85-36, Center for the Study of Language and Information, Stanford University, October 1985.

5. Davis, R. 1985. Expert Systems: What to Do Until the Theory Arrives. In Proceedings of the Ninth International Joint Conference on Artificial Intelligence, 1306–1309. Menlo Park, Calif.: International Joint Conferences on Artificial Intelligence.

6. As of mid-1989, the number of examples run on Xcon has increased to something on the order of 500,000 cases run as a consequence of routine use of the program worldwide throughout Digital; the error rate is between three and five percent. How many other AI programs have ever been run on 500,000 examples? There has clearly been extensive testing and debugging of the program, yet a 4 percent error rate means there have been something on the order of 20,000 mistakes. Both of these are interesting and sobering figures.

Randall Davis is a professor of information science in the School of Management and associate director of the Artificial Intelligence Laboratory at the Massachusetts Institute of Technology (MIT). He received his Ph.D. in artificial intelligence from Stanford University in 1976 and joined the MIT faculty in 1978, where he held an Esther and Harold Edgerton endowed chair from 1979 through 1981. Davis's current research focuses on model-based systems—programs which work from descriptions of structure and function and which reason from first principles. He and his group at MIT have developed model-based systems for troubleshooting, the generation of diagnostics, design for testability, innovative design, and other functions.

Davis has published widely in the field of AI and serves on several magazine editorial boards, including *Artificial Intelligence* and *AI in Engineering.* In 1984, he was selected as one of America's top 100 scientists under the age of 40 by *Science Digest.*

Brian Cantwell Smith is a principal scientist and area manager at Xerox Palo Alto Research Center, 3333 Coyote Hill Road, Palo Alto, CA 94304. He has been a consulting professor of philosophy at Stanford University since 1983 and is a founder of the Center for the Study of Language and Information (CSLI) at Stanford. He is also a founder and first president of Computer Professionals for Social Responsibility (CPSR). Smith received his doctoral degree in AI from Massachusetts Institute of Technology in 1982. His dissertation work on programming languages and reflection (3-Lisp) has led to a focus on foundational issues in computation and cognitive science, the topic of his forthcoming book, *A View from Somewhere.*