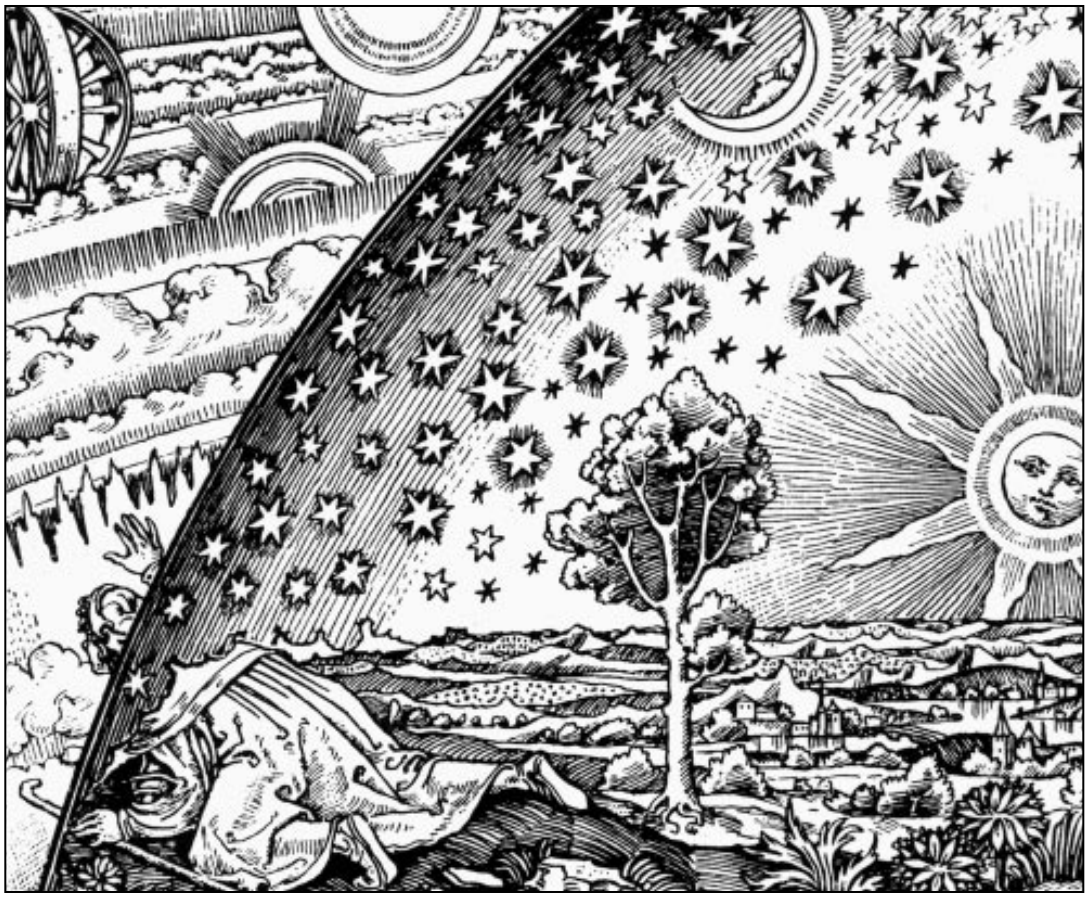

WHERE'S THE AI?

Roger C. Schank

I survey four viewpoints about what AI is. I describe a program exhibiting AI as one that can change as a result of interactions with the user. Such a program would have to process hundreds or thousands of examples as opposed to a handful. Because AI is a machine's attempt to explain the behavior of the (human) system it is trying to model, the ability of a program design to scale up is critical. Researchers need to face the complexities of scaling up to programs that actually serve a purpose. The move from toy domains into concrete ones has three big consequences for the development of AI. First, it will force software designers to face the idiosyncrasies of its users. Second, it will act as an important reality check between the language of the machine, the software, and the user. Third, the scaled-up programs will become templates for future work.

0738-4602/91/$4.00 ©1991 AAAI

It is not that AI needs definition; it is more that AI needs substance…

For a variety of reasons, some of which I discuss in this article, the newly formed Institute for the Learning Sciences has been concentrating its efforts on building high-quality educational software for use in business and elementary and secondary schools. In the two years that the institute has been in operation, it has created quite a few prototypes that go beyond the kind of things that have traditionally been built under the heading of educational software. Behind these new designs are a number of radically different "teaching architectures" that change the nature of how students interact with computers, moving from what is called "the page-turning architecture" to architectures based on simulation-driven learning by doing, story- or case-based teaching, and Socratic dialogues (Schank 1990a; Schank and Jona 1991).

These prototypes have been successful, and for the most part, the institute's sponsors have been impressed with the results, acknowledging that they have never seen anything quite like them before. Nevertheless, researchers at the institute have been plagued by a curious kind of question when they show these prototypes to certain audiences. The question they keep hearing is the title of this article: Where's the AI?

At first, I found this question puzzling. Everything they do is AI. But apparently, AI has a specific definition to many people, and these programs didn't fit this definition. My concern was to figure out what this definition was. And while I am at it, maybe it would be helpful if the field of AI itself understood the answer to this question in a more profound way. It is not that AI needs definition; it is more that AI needs substance, a distinction that I discuss later.

Four Viewpoints on AI

When someone who is not in AI asks where the AI is, what assumptions about AI are inherent in the question? There seem to be at least four prevailing viewpoints that I have to deal with, so this question assumes at least one of the following four things: (1) AI means magic bullets, (2) AI means inference engines, (3) AI means getting a machine to do something you didn't think a machine could do (the "gee whiz" view), and (4) AI means having a machine learn.

The magic bullet view of AI is as follows: Intelligence is actually difficult to put into a machine because it is knowledge dependent. Because the knowledge-acquisition process is complex, one way to address it is to finesse it. Let the machine be efficient computationally so that it can connect things to each other without having to explicitly represent anything. In this way, the intelligence comes for free as a by-product of unanticipated connections that the machine makes.

An alternate version of this view is that AI is something that one could, in principle, discover in one's garage. What the form of this discovery might be remains a mystery, but one would drop the discovered item or technique into the machine, and it would become intelligent. This view is held, quite firmly, by many people who write me letters after having read my name in a magazine article as well as by many venture capitalists (and, possibly, some connectionists).

The inference engine view of AI was brought forth by a variety of people on the West Coast (for example, the MYCIN [Shortliffe 1976] and DENDRAL [Buchanan and Feigenbaum 1978] projects at Stanford University and the PROSPECTOR [Duda, Gaschnig, and Hart 1979] project at SRI International). Their view was, I suspect, an answer to this same question. When the expert system world began to explode, AI experts were expert at finding out what an expert knew and writing down the knowledge in rules that a machine could follow (this process came to be known as *knowledge engineering* [Feigenbaum 1977]). Although such expert systems could, in fact, make some interesting decisions, business types who were called on to evaluate the potential of such systems probably asked, Where's the AI? The real answer was that the AI was in the ability of AI people to find out what the experts knew and to represent the

…the gee whiz view… maintains that…if no machine ever did it before, it must be AI.

information in some reasonable way, but this answer would not impress a venture capitalist. There had to be something that could be labeled AI and that something came to be known as an inference engine.[1] Of course, much of the AI world understood that inference was an important part of understanding, so it made sense that an expert system would need to make inferences too, but to label the inference engine as the AI was both misleading and irrelevant. The business world believed that it needed this label, however, as I discuss later on.

Thus, it came to pass that business people who read about AI and assumed that AI and expert systems were identical began to expect inference engines in anything they saw on a computer that "had AI in it." Many other people in AI were left in a quandary, however, about how to explain that what they did was also AI, which led to questions such as, What is AI anyway? This question was never easy to answer without discussing the nature of intelligence, a subject best left undiscussed in a business meeting.

I term the next view the gee whiz view. This view maintains that for a particular task, if no machine ever did it before, it must be AI. Two important types of programs to discuss within this conception of AI are optical character readers and chess-playing programs. Are these programs AI? Today, most people would say that they are not. Years ago, they were.[2] What happened?

The answer is that they worked. They were AI as long as it was unclear how to make them work. When all the engineering was done, and they worked well enough to be used by real people, they ceased to be seen as AI. Why? The answer is threefold: First, "gee whiz" only lasts so long. After a while, people get used to the idea that a computer can do something they didn't know it could do. Second, people tend to confuse getting a machine to do something intelligent with getting it to be a model of human intelligence, and surely, these programs aren't intelligent in any deep sense. Third, the bulk of the work required to transform an AI prototype into an unbreakable computer program looks a lot more like software engineering than it does like AI to the programmers who are working on the system, so it doesn't feel like you are doing AI even when you are.

The fourth view of AI is one that I myself have espoused. It is a long-term view. It says that intelligence entails learning. When your dog fails to understand his environment and improve on mistakes he has made, you refer to him as "dumb." *Intelligence* means getting better over time. No system that is static, that fails to change as a result of its experiences, looks smart. Real AI means a machine that learns. The problem with this definition is that according to it, no one has actually done any AI at all, although some researchers have made some interesting attempts.[3] According to this view, there is no AI, at least not yet. Thus the question, Where's the AI? is a tough one to answer.

Knowing Something Helps, Knowing a Lot Helps More

Let's consider the SAM-FRUMP-ATRANS experience. In 1976, a group of my students and myself built SAM, a program that summarized, paraphrased, translated, and answered questions about newspaper stories (Cullingford 1978, 1981). The program was slow and cumbersome and was capable of reading only a few stories in a few domains. Nevertheless, it was the first program to perform this task, and we were proud of it. No one asked where the AI was because no one had ever seen anything like it. It qualified as AI based on the gee whiz criterion.

In 1978, we attempted to improve on SAM in some ways, thus came FRUMP (DeJong 1979b). FRUMP was very fast in comparison to SAM. By this time, we had even faster machines of course, but the major speedups were accomplished by limiting inferencing and attempting to understand only the gist of any given story. We were able to blur the distinction between inferencing and parsing in a way that made interesting theoretical claims (DeJong 1979a).

We were excited by FRUMP. It worked on roughly 50 different domains and succeeded on about 75 percent of the stories that were in these domains. The branch of the United States Department of Defense that had sponsored this work wanted to use FRUMP for a particular task. We were put in the strange position (for a university research lab) of having to hire people to code domain knowledge into FRUMP. After a year of this coding, FRUMP's robustness had only slightly improved, and the sponsor was annoyed because it had actually wanted to use the program. Further, I was concerned about having to hire more and more non-AI people to make a piece of software really work. I had always thought that our research money was for supporting graduate students to do research. I was uneasy about actually trying to develop a product. And it didn't interest me all that much to try. What did it mean to be doing

AI, I wondered. It meant building theories and programmed examples of these theories, it seemed to me, in the attempt to understand the nature of the mind as part of the ultimate quest for the learning machine of the fourth viewpoint given earlier. In 1978, I was not interested in building products.

By 1982, though, I had become interested in trying to build AI products. Why? Probably the main thing that had changed was AI. The expert system hype had hit, and the venture capitalists were everywhere. I spoke loudly and often against this overhype and ran a panel that year at the National Conference on Artificial Intelligence (held at Carnegie-Mellon University in Pittsburgh) on the coming AI winter that was likely to result from ridiculous expectations.[4] At the same time, I had begun to think more and more about the FRUMP experience. I realized that AI people really had to produce something that worked at some point, that we couldn't simply write papers about ideas that might work or worked on a few examples and satisfied the sponsors who were pouring real money into AI. With the increase in money coming from the commercial world, the die was cast. We had to produce something then, I figured, or we might not get another chance. Thus, I, too, started a company, called Cognitive Systems, with FRUMP as the intended product.

Why FRUMP never was the product of Cognitive Systems is a business discussion not relevant here. What Cognitive Systems did wind up building was something called ATRANS, which was a program that read international bank money transfer messages (Lytinen and Gershman 1986). The program is working today in various international banks. It parses sentences that all have similar content about how and where money is to be transferred from country to country. In essence, ATRANS is a single-domain (and a narrow domain at that) FRUMP. A version of this program is also used by the United States Coast Guard. At a Defense Department meeting three years ago, ATRANS was identified as one of only three AI programs actually in use for defense.

Why am I telling you all this? Because there is 1 important fact to know about ATRANS. It took something like 30 person-years to make it work. This number is in addition to any of the SAM-FRUMP work. There is an important lesson to be learned here.

To make AI—real AI—programs that do something someone might want (which is, after all, the goal of AI for those who fund AI and for many of those in AI), one must do a great deal of work that doesn't look like AI. Any of Cognitive Systems' programmers would have been justified in complaining that they had come to work there to do AI, and all they were doing was working on endless details about determining various abbreviations for bank names. They also asked, Where's the AI?

The lesson to be learned from ATRANS is simple enough. AI entails massive software engineering. To paraphrase Thomas Edison, "AI is 1-percent inspiration and 99-percent perspiration." AI people will never build any real AI unless they are willing to make the tremendously complex effort that is involved in making sophisticated software work.

But this discussion still doesn't answer the original question. Sponsors can still honestly inquire about where the AI is, even if it is in only 1 percent of the completed work.

From Three Examples to Many

The answer to Where's the AI? is that it's in the size. Let me explain.

For years, AI people specialized in building systems that worked on only a few examples or in a limited domain (or microworld). Sometimes these systems didn't really work on any examples at all; it just seemed plausible that they would work. The practice of getting AI programs to work on a few examples is so rooted in the history of AI that it is rarely discussed. There are many reasons for this practice, but the simplest explanation is that until the creation of the various venture capital backed companies in the early 1980s, almost all AI programs were Ph.D. theses that proved the concept of the thesis with a few examples. It was almost never anyone's job to finish the thesis.[5] Often, it wasn't clear what this finishing meant anyway because these theses were rarely directed at real problems for which there was a user waiting. In any case, even if there was a ready use for the project, no one wanted to tackle the inherently uninteresting task of doing all that software engineering—at least no one in an AI lab wanted to. And even if someone did want to do it, there was no one who wanted to pay for it or who seriously understood how much it would really cost to do the other 99 percent.

Nevertheless, there were otherwise intelligent people claiming that Winograd's (1972) SHRDLU program that worked on 31 examples had solved the natural language problem or that MYCIN (Shortliffe 1976) had solved the problem of getting expertise into a computer.[6] Prior to 1982, it is safe to say that no one had

really tried to build an AI program that was more than simply suggestive of what could be built. AI had a real definition then, and it was the gee whiz definition given earlier.

But underlying even this definition was the issue of scaleup. AI people had always agreed among themselves that scaleup was the true differentiation of what was AI from what was not AI. This measure of AI was one of those things that was so clearly a defining characteristic of the field that there was no need to actually define it on paper, at least I am unaware of any specific statement of this view of what was AI in the 1960s and 1970s that we all adhered to. So I will say it now. And true to form, I will say it in the form of a story:

When I arrived at Yale in 1974, there was a junior faculty member there who, I was told, was in AI. I looked into what he was doing and decided that he was not in AI. Here was what happened: He was working on speech recognition. He had developed an algorithm for detecting which of the numbers 1 through 10 had been spoken into a computer. As far as I could tell, the program did what it claimed to do. I asked, as any AI person would, how the program had accomplished the task. The question was not what he had done but how he had done it. Here's why.

Suppose his program had determined which of the numbers 1 through 10 had been said by comparing the sound that was received to a prototype of what each of the 10 numbers should sound like and then determining the best match of the features of the incoming sound to the features of the prototype. This method seems like a reasonable one for performing this task, and it should work. Why then isn't it AI?

It isn't AI because it is unlikely that it would scale up. The key concepts here are "scaleup" and "unlikely." If the problem that this man was working on was the "detection of 10 numbers" problem, it really wouldn't matter how he did it. Any algorithm that worked would be terrific if someone wanted a program to do this task. But AI has always been concerned with the solution of the deep problem behind the small problem. The issue in this part of AI was how a program would detect any word that was spoken, and the solution being proposed, to be an AI solution, had to address this issue. In other words, the question that I asked myself was whether what he had done for 10 words would work for any word. That is the AI question.

Now, obviously, I decided that his solution (which was indeed the one given earlier) would not scale up. This conclusion was easy to come to because for it to scale up, he would be proposing matching any new word to a database of hundreds of thousands of prototypes. He had not even begun to think about this type of matching because he was really not an AI person in any sense of the word. He was simply working on a problem that looked like AI to an outsider.

Suppose he had been thinking about doing speech recognition as a step-by-step differentiation process. At the time, I would have said that this approach was silly, that what was needed was a theory of recognition of phonemes in the context of predictions derived from language understanding in real situations. It is at this point that "unlikely" comes in. I was, after all, simply guessing about whether his system would scale up. It was a good guess because he hadn't thought about anything except 10 words, and it is unlikely that his solution for 10 words was going to happen to work for all words. What bothers me about this story today is that although I would still like to see speech recognition driven by a good theory of language understanding, it is nevertheless now possible to conceive of a program that really does store all the words of English and compare one to another. Would this program be AI?

Certainly, there are those who would say that it was AI. This debate aside for the moment, it is possible to come to a point of view that defines AI as one of two things: Either an AI program is one based on a theory that is likely to scale up, or it is a program based on an algorithm that is likely to scale up. Either way, I am talking about best guesses about promises for the future.

The point here is that the AI is in the size or at least the potential size. The curiosity here is that when the size gets big enough, it all ceases to matter. We merely need to look at AI work in chess.

Chess was one of the original AI problems. Getting a machine to do something that only smart people can do seemed a good area for AI to work on. Now, many years later, you can buy a chess-playing program in a toy store, and no one claims that it is an AI program. What happened?

For many years, the popular wisdom was that AI was a field that killed its own successes. If you couldn't do it, the wisdom went, it was AI, and when you did it, it no longer was, which seems to be what happened in chess, but the reality is somewhat more subtle. Chess was an AI problem because it represented one form of intelligent behavior. The task in AI was to create intelligent machines, which meant having them exhibit intelligent behavior. The problem was—and is—that

…an AI program is not intended to accomplish a particular task but rather to help shed light on solutions for a set of tasks.

exactly what constitutes intelligent behavior is not exactly agreed on. With my scaleup measure, the idea in looking at a chess program would have been to ask how its solution to the chess problem scaled up. Or to put it another way, were people writing chess programs because they wanted computers to play chess well, or were they writing them because they scaled up to other problems of intelligent behavior?

It would seem that no one would want a chess program for any real purpose, but this question, after all, is a marketing one. If the toys made money, well, fine: It doesn't matter how they work; it matters that they work. But if we care about how they work, there are only two possible questions: First, we need to know if the solution to making them work tells us anything at all about human behavior. Second, we would want to know if it tells us something that we could use in any program that did something similar to, but more general than, playing chess.

Brute-force, high-speed search through a table of possible prestored moves is unlikely to be anything like the human method for chess playing. Chess players do seem to understand the game they are playing and are able to explain strategies, and so on, after all. Thus, the only other question is whether one could use these same methods to help tell us something about how to solve problems in general. In fact, the original motivation to work on chess in AI was bound up with the idea of a general problem solver (for example, Newell and Simon's [1963] GPS system). The difficulty is that what was learned from this work was that people are really specific problem solvers more than they are general problem solvers and that the real generalizations to be found are in how knowledge is to be represented and applied in specific situations. Brute-force chess programs shed no light on this issue at all and, thus, are usually deemed not to be AI.

Thus, the scaleup problem can refer to scaleup within a domain as well as to scaleup in the greater domains that naturally embody smaller domains. But the chess work didn't scale up at all, so the reasonableness of doing such work is simply a question of whether this work was needed by anybody. If there had been a need for chess programs, then chess would have been seen as a success of AI but probably not as actual AI. In the end, AI was never supposed to be about need, however. In the 1960s and 1970s, most AI people didn't care if someone wanted a particular program or not. AI was research.

The correct AI question had to do with the generality of a solution to a problem, and there was a good reason. It is trivial to build a program to do what, say, Winograd's (1972) SHRDLU program did for 31 sentences. Just match 31 strings with 31 behaviors. It would take a day to program. People believed that Winograd's program was an AI program because they believed that his program "did it right." They believed it would scale up. They believed that it would work on more than 31 sentences. (In fact, so did he. See Winograd [1973]). At the time, when I was asked my opinion of Winograd's work, I replied that it would never work on a substantially larger number of sentences, nor would it work in different domains than the one for which it was designed. I did not reply that his program was not AI, however.

The fact that a program does not scale up does not necessarily disqualify it from being AI. The ideas in Winograd's program were AI ideas; they just weren't correct AI ideas in my opinion. What then does it mean for a program to have AI ideas within it? After all, this question is a key one in our search to find the location of the AI.

So to summarize the argument so far, an AI program exhibits intelligent behavior, but a non-AI program could as well. An AI program should scale up, but many do not. And an AI program has AI ideas in it. Further, an AI program is not intended to accomplish a particular task but rather to help shed light on solutions for a set of tasks. This summary was, more or less, the standard view of AI within AI prior to 1980.

The fact that a program does not scale up does not necessarily disqualify it from being AI.

The Great Expert System Shell Game

At the beginning of the last section, I said that the answer to the title question was in the size. When building FRUMP, my students and I realized that the difference between FRUMP and something somebody wanted to actually use involved a great deal of engineering of some rather dull information. To build ATRANS, we had to bite this bullet. Little by little it became clear to me that to build an AI program that someone wanted, an idea that was virtually a contradiction in terms in the 1970s, someone would have to stuff a machine with a great deal of knowledge. Smart is nice, but ignorant tends to obviate smart. Gradually, it was becoming clear that AI would have to actually work and that making it work might mean paradoxically, using our own definitions, not doing AI at all.

This situation was all brought to a head by the great expert system shell game. When the venture capitalists discovered AI, they brought more than just money to the table. They also brought with them the concept of a product. Now, it took me a long time to understand what the word product was supposed to mean, so don't assume (if you are in AI) that you understand it. In fact, I am still not sure I understand it in a profound way. I thought, for example, that a conceptual parser (for example, ELI [Riesbeck and Schank 1976] or CA [Birnbaum and Selfridge 1979]) might be a product. It isn't. I thought that FRUMP might be a product. It wasn't. An expert system shell is a product. Unfortunately, it's not always a profitable one (which is, not surprisingly, the standard measure of goodness). I don't intend to explain here what a product is. What is important to understand is why the venture capitalists insisted that their AI people build expert system shells.

When the expert system folks went into business, I was skeptical that they could build anything that anyone really wanted. After all, no one had ever done that in AI before, and as I have said, it was kind of out of the bounds of AI to even try. But I figured, they were smart people and quite good with Lisp (the programming language used by many AI researchers), and maybe they could pull it off. When I heard about the plan that the venture people had for them, however, I knew they were doomed.

The plan went like this: If the expert system people were given a hard problem, enough of them could build a program that could solve this problem within reasonable constraints. But no one solution would have been a product. For a certain amount of money, they could build a program to solve problem A, but they wouldn't be a great deal closer to solving problem B. Venture capitalists would never invest in a custom software development company. They want a company that makes one thing and then sells that thing 100,000 times. A company that makes one thing and sells it once isn't much of an investment. What the expert system people knew how to do was build custom software. But there isn't much money in doing it. What they were asked to do instead was build a *shell*, an environment for developing software. This project made sense to the venture guys, and the AI guys had to go along. The problem was that although the shell might be great to sell, first, it would be wrong, and second, where was the AI?

The first issue first: Why would it be wrong? The assumption of the venture capitalists was that given the right tool, any programmer could build an expert system. This idea was a marketable one, so it worked in their terms. Of course, it wasn't an idea that had a lot of reality in it. Building a complex AI system is certainly made easier by having a good environment for programming, but knowing what to program is the real issue. So one is left addressing the second issue, namely, where is the AI? This issue was addressed from a business point of view with the concept of an inference engine. The idea was that there was a piece of magic that was the AI and that this magic, plus a software development environment that made it easy to build these things, was salable. And it was, at least initially. The problem was that little of a

really complex nature could be built with these shells. Or in other words, a programming environment plus an inference engine doesn't make up all there is to building an expert system, which might not have been the thing to be building in the first place.

Where was the AI? It wasn't in the inference engine at all. These inference engines were, after all, pretty simple pieces of software that tested to see if the logic of the rules that the knowledge engineers wrote came up with any conclusions. The AI in complex expert systems was in the organization and representation of knowledge, the attempt to understand the domain under study, and the crystallization of what was important in the domain and how experts in the domain reasoned. Now, I was saying at the time (see, for example, Schank et al. [1977]) that the AI was also in collecting the actual experiences of the experts and indexing them so that reminding and, hence, learning could take place, but the expert system folks were in no mood to pay attention. Those that did were usually not involved in the shell game. To put it another way, the AI was where it always was—in the attempt to understand the intelligent behavior in the system being modeled.

The fact that no expert ever experiences anything in his(her) domain of expertise without learning something from the experience and changing in some way was easily ignored.[7] The hope was that static systems would model an expert's reasoning ability at any moment. But experts don't remain experts for long if all they do is blindly apply rules. The act of creating a shell with which non-AI people could write rules to be run by an inference engine was, in a sense, an act of not doing AI. What had been left out was the skill that AI people had been learning all this time, the skill of figuring out what was going on in a domain and getting a machine to model the human behavior that occurred in the domain. What had been left out were the AI ideas.

Some companies tried to remedy this situation by training the users of these new shells to become knowledge engineers. But AI is kind of a funny subject. For years, we have been taking graduates from the best institutions in the country and teaching them AI. Often, even after a few years, they just don't get it. What don't they get? They don't get what the issues are, how to attack an issue, how to have an idea about an unsolved problem, and how to build programs that embody these ideas. AI is a way of looking at complex problems, and often, it is difficult to learn to do. It seemed to me that it was hopeless to try to teach this area to programmers in a few weeks when years often didn't help.

Thus, my claim is that although some expert systems might well have been AI, few of those built with inference engines were likely to be unless one changed the then-current definition of AI. The change is easy enough to describe. AI had been the creation of new ideas about how to represent and use complex knowledge on a computer. Now, the definition was changing toward AI being programs that utilized these AI ideas in some application that someone wanted.

Where was the AI in the expert system shells? It was in the assertion that rules would effectively model expertise and in the programs that attempted to implement this assertion. The only problem was that for complex domains—that is, for AI-like domains of inquiry—the assertion was wrong.

The Larger the Better

One of the real issues in AI, as I mentioned earlier, is size. When we talk about scaleup, we are, of course, talking about working on more than a few examples. What every AI person knows is that a program that works on 5 examples is probably not one-tenth the size of one that works on 50. Outsiders imagine that it might be the same size or, possibly, one-half the size. But what is really the case is that for real domains, the size changes work the other way. Once you have to account for all the myriad possibilities, the complexity is phenomenal. It is critical, if AI is to mean applications of AI ideas rather than simply the creation of these ideas, that size issues be attacked seriously. No one needs a program that does 5 examples. This approach worked in 1970 because AI was new and glossy. It will not work any longer. Too much money has been spent. AI has to dive headlong into size issues.

Now, as it turns out, although this issue might seem to be annoying and, possibly, dull as dust, the truth is that it is size that is at the core of human intelligence. In my recent book, *Tell Me a Story* (Schank 1990b), I argue that people are really best seen as story-telling machines, ready to tell you their favorite story at a moment's notice. People rarely say anything that is new or something they have never said before. Looked at in this way, conversation is dependent on the art of *indexing*, finding the right thing to say at the right time.[8] This problem is pretty trivial for someone who only has three stories to tell. Much like our speech-recognition friend earlier, it is simply a problem of differentiation.

Real problems are needed for prototyping.

Many a grandfather has survived many a conversation on the same few stories.

But when the numbers get into the thousands, one has to be clever about finding—and finding quickly—germane stories to tell. We cannot even begin to attack this problem until the numbers are large enough. To get machines to be intelligent, they must be able to access and modify a tremendously large knowledge base. There is no intelligence without real—and changeable—knowledge.

And this thought, of course, brings us back to the original question: If a system is small, can there be AI in it? In the 1970s, AI systems were all, in essence, promises for the future. They were promises about potential scaleup, promises that the theoretical basis of the program was sound enough to allow scaleup. Now, the question is, How big is big enough to declare a system an AI system?

It is fairly clear that although this question is an important one, it is rather difficult to answer, which brings us back to the implicit issue in this discussion—the concept of an AI idea. When I said earlier that certain programs have AI ideas within them, I was declaring that even programs that did not work well and never did scale up were AI programs. What could this mean?

AI is about the representation of knowledge. Even a small functioning computer program that someone actually wanted could be an AI program if it were based on AI ideas. If issues of knowledge representation were addressed in some coherent fashion within a given program, AI people could claim that it was an AI program. But the key point is that this question, Where's the AI? is never asked by AI people; it is asked by others who are viewing a program that has been created in an AI lab. And the important point is that to these people, it simply shouldn't matter. The answer to the question about where the AI is to be found in a program that does a job that someone wanted to do is that the AI was in the thinking of the program's designers and is represented in some way in the programmer's code. However, if the reason that they wanted this program in some way depends on the answer to this question, that is, if they wanted AI in the program they were sponsoring, they are likely to be rather disappointed.

Five Issues to Think about Before You Try to Do Real AI

If this answer to the question about where the AI is makes no sense to an outsider, as it surely will not, I hope that it makes sense to people who are in AI. AI is in a rather weird state these days. AI people are hoping to live the way they lived in the 1970s but for a variety of reasons, those days are over. We cannot continue to build programs that we hope will scale up. We must scale them up ourselves.

It might be that one can argue that we are not ready to face the scaleup issue just yet, that the fundamental problems have not been solved, that we don't know all there is to know about the mind and how to model it. This statement seems fair enough, not to mention true enough. Nevertheless, because of the realities of the 1990s, we must give it a try. There are things we can do, and there are some good reasons to try. For one thing, sponsors will expect it. For another thing, the problems of AI demand it—we simply must start to look at the scaleup problems for the sound theoretical reason that these considerations will force us to address many real AI problems. But I think the most important reason is that this is where the action has to take place. The sheer amount of difficulty present in the creation of a functioning piece of software from a prototype that worked on a few examples is frightening. We simply must learn how to deal with these issues, or there never will be any AI. AI people cannot keep expecting that non-AI people will somehow magically turn their prototypes into reality. It will simply never happen. The worst effect of the shell game is that it got AI people believing what venture capitalists wanted to believe.

If you buy what I am saying, then the following five issues represent some practical problems that must be faced before you do any real (scaled-up) AI:

Real problems are needed for prototyping: We cannot keep working in toy domains. Real problems identify real users with real needs, considerably changing what the interactions with the program will be, but they must be part of the original design.

Real knowledge that real domain experts have must be found and stored: This statement does not mean interviewing them and asking for the rules that they use and ignoring everything else that fails to fit. Real experts

have real experiences, contradictory viewpoints, exceptions, confusions, and the ability to have an intuitive feel for a problem. Getting at these issues is critical. It is possible to build interesting systems that do not know what the experts know. Expertise can be captured in video, stored and indexed in a sound way, and retrieved without having to fully represent the content of the expertise (for example, the ASK TOM system [Schank et al. 1991]). Such a system would be full of AI ideas, interesting to interact with, and not wholly intelligent but a far sight better than systems that did not have such knowledge available.

Software engineering is harder than you think: I can't emphasize strongly enough how true this statement is. AI had better deal with the problem.

Everyone wants to do research: One serious problem in AI these days is that we keep producing researchers instead of builders. Every new Ph.D. recipient, it seems, wants to continue to work on some obscure small problem whose solution will benefit some mythical program that no one will ever write. We are in danger of creating a generation of computationally sophisticated philosophers. They will have all the usefulness and employability of philosophers as well.

All that matters is tool building: This statement might seem odd considering my comments about the expert system shell game. However, ultimately, we will not be able to build each new AI system from scratch. When we start to build useful systems, the second one should be easier to build than the first, and we should be able to train non-AI experts to build them. I don't mean that these tools will allow everyone to do AI on their personal computers. It does mean that certain standard architectures should evolve for capturing and finding knowledge. From this point of view, the shell game people were right; they just put the wrong stuff in the shell. The shell should have had expert knowledge about various domains in it, knowledge that is available to make the next system in the domain that much easier to build.

These five issues are real and important to think about. They are practical points, not theoretical ones. A little practicality might help the field get to the next level of theory.

OK, But What Do You Really Think?

AI depends on computers that have real knowledge in them. Thus, the crux of AI is in the representation of this knowledge, the content-based indexing of this knowledge, and the adaptation and modification of this knowledge through the exercise of this knowledge. What I really think is that case-based reasoning (Riesbeck and Schank 1989; Jona and Kolodner 1991) is a much more promising area than expert systems ever were and that within the area of case-based reasoning, the most useful and important (and maybe even somewhat easier) area to work in is case-based teaching. Building real, large, case bases and then using them as a means by which users of a system can learn is a problem we can attack now that has enormous import for both AI and the users of such systems.

Case-based teaching depends on solving the following problems: the indexing of memory chunks, the setting up of tasks based on indexing, the matching of student state to an index, the anticipation of the next question, knowledge navigation, problem cascades, Socratic teaching, and button-based interaction.

I will not bother to explain each of these problems because I have done so elsewhere (Schank 1991). The major point is that even though getting machines to tell what they know at relevant times is a simpler form of AI than the full-blown AI problem, it is not simple. Even the kind of practical, developmental form of AI that I am proposing is full of enough complex problems to keep many a theoretician busy. I would just like theoretically minded AI people to stop counting angels on the head of a pin.

So where is the AI? It is in the size, the ideas, and the understanding of what is significant that contributes to the behavior of intelligent beings.

Notes

1. Inference engine is just another name for a production system; see Davis and King (1977) and Waterman and Hayes-Roth (1978).

2. For example, in *Computers and Thought*, one of the early seminal AI books, there appears a chapter entitled "Chess-Playing Programs and the Problem of Complexity" (Newell, Shaw, and Simon 1963). In the entry for optical character recognition in the *Encyclopedia of Artificial Intelligence*, Hull (1987) states: "The reading of text by computer has been an AI topic for more than 25 years" (p. 82).

3. For example, see Carbonell (1986); Carbonell and Gil (1990); DeJong and Mooney (1986); Hammond (1989); Lenat (1983); Mitchell (1982); Mitchell, Keller, and Kedar-Cabelli (1986); Quinlan (1986); and Sussman (1975).

4. Another panel on this same subject that I also

participated in, entitled "The Dark Ages of AI," was held two years later at the 1984 National Conference on Artificial Intelligence. For a transcript, see McDermott et al. (1985).

5. See McDermott's (1981) essay lamenting this fact and its effect on the progress of AI.

6. Dreyfus (1979) criticizes these and other overly ambitious claims of progress made by AI researchers.

7. Now we have some evidence that experts do indeed learn from and use their experiences in daily reasoning and decision making (see, for example, Klein and Calderwood [1988]). For example, both expert and novice car mechanics were found to use their past experiences to help generate a hypothesis about what kind of problem a car might have (Lancaster and Kolodner 1988; Redmond 1989). Architects and mechanical engineers were observed using old design plans while they created new ones (Goel and Pirollo 1989).

8. For further discussion on indexing and the indexing problem, see Jona and Kolodner (1991), Riesbeck and Schank (1989), and Schank et al. (1990).

References

Birnbaum, L., and Selfridge, M. 1979. Problems in Conceptual Analysis of Natural Language, Technical Report, 168, Dept. of Computer Science, Yale Univ.

Buchanan, B., and Feigenbaum, E. 1978. DENDRAL and META-DENDRAL: Their Applications Dimension. *Artificial Intelligence* 11:5–24.

Carbonell, J. 1986. Derivational Analogy: A Theory of Reconstructive Problem Solving and Expertise Acquisition. In *Machine Learning: An Artificial Intelligence Approach*, volume 2, eds. R. Michalski, J. Carbonell, and T. Mitchell, 371–392. San Mateo, Calif.: Morgan Kaufmann.

Carbonell, J., and Gil, Y. 1990. Learning by Experimentation: The Operator Refinement Method. In Machine L*earning: An Artificial Intelligenc*e Approach, volume 3, eds. Y. Kordratoff and R. Michalski, 191–213. San Mateo, Calif.: Morgan Kaufmann.

Cullingford, R. 1981. SAM. In *Inside Computer Understanding*, eds. R. Schank and C. Riesbeck, 75–119. Hillsdale, N.J.: Lawrence Erlbaum.

Cullingford, R. 1978. Script Application: Computer Understanding of Newspaper Stories. Ph.D. diss., Technical Report, 116, Dept. of Computer Science, Yale Univ.

Davis, R., and King, J. 1977. An Overview of Production Systems. In *Machine Intelligence 8*, eds. E. Elcock and D. Michie, 300–332. Chichester, England: Ellis Horwood.

DeJong, G. 1979a. Prediction and Substantiation: A New Approach to Natural Language Processing. *Cognitive Science* 3:251–273.

DeJong, G. 1979b. Skimming Stories in Real Time: An Experiment in Integrated Understanding. Ph.D. diss., Technical Report, 158, Dept. of Computer Science, Yale Univ.

DeJong, G., and Mooney, R. 1986. Explanation-Based Learning: An Alternative View. *Machine Learning* 1:145–176.

Dreyfus, H. 1979. *What Computers Can't Do: The Limits of Artificial Intelligence*, rev. ed. New York: Harper and Row.

Duda, R.; Gaschnig, J.; and Hart, P. 1979. Model Design in the PROSPECTOR Consultant System for Mineral Exploration. In *Expert Systems in the Micro-Electronic Age,* ed. D. Michie, 153–167. Edinburgh: Edinburgh University Press.

Feigenbaum, E. 1977. The Art of Artificial Intelligence: Themes and Case Studies of Knowledge Engineering. In Proceedings of the Fifth International Joint Conference on Artificial Intelligence, 1014–1029. Menlo Park, Calif.: International Joint Conferences on Artificial Intelligence.

Hammond, K. 1989. *Case-Based Planning: Viewing Planning as a Memory Task*. Boston: Academic.

Hull, J. 1987. Character Recognition: The Reading of Text by Computer. In *Encyclopedia of Artificial Intelligence*, ed. S. Shapiro, 82–88. New York: Wiley.

Goel, V., and Pirolli, P. 1989. Motivating the Notion of Generic Design within Information-Processing Theory: The Design Problem Space. *AI Magazine* 10(1): 18–36.

Jona, M., and Kolodner, J. 1991. Case-Based Reasoning. In *Encyclopedia of Artificial Intelligence*, 2d ed. New York: Wiley.

Klein, G., and Calderwood, R. 1988. How Do People Use Analogues to Make Decisions? In *Proceedings of the Case-Based Reasoning Workshop (DARPA)*, ed. J. Kolodner, 209–218. San Mateo, Calif.: Morgan Kaufmann.

Lancaster, J., and Kolodner, J. 1988. Varieties of Learning from Problem-Solving Experience. In *Proceedings of the Tenth Annual Conference of the Cognitive Science Society*, 447–453. Hillsdale, N.J.: Lawrence Erlbaum.

Lenat, D. 1983. The Role of Heuristics in Learning by Discovery: Three Case Studies. In *Machine Learning: An Artificial Intelligence Approach*, eds. R. Michalski, J. Carbonell, and T. Mitchell, 243–306. San Mateo, Calif.: Morgan Kaufmann.

Lytinen, S., and Gershman, A. 1986. ATRANS: Automatic Processing of Money Transfer Messages. In Proceedings of the Fifth National Conference on Artificial Intelligence, 1089–1093. Menlo Park, Calif.: American Association for Artificial Intelligence.

McDermott, D. 1981. Artificial Intelligence Meets Natural Stupidity. In *Mind Design*, ed. J. Haugeland, 143–160. Montgomery, Vt.: Bradford.

McDermott, D.; Waldrop, M.; Schank, R.; Chandrasekaran, B.; and McDermott, J. 1985. The Dark

Ages of AI: A Panel Discussion at AAAI-84. *AI Magazine* 6(3): 122–134.

Mitchell, T. 1982. Generalization as Search. *Artificial Intelligence* 18:203–226.

Mitchell, T.; Keller, R.; and Kedar-Cabelli, S. 1986. Explanation-Based Generalization: A Unifying View. *Machine Learning* 1:47–80.

Newell, A., and Simon, H. 1963. GPS, A Program That Simulates Human Thought. In *Computers and Thought*, eds. E. Feigenbaum and J. Feldman, 279–293. New York: McGraw-Hill.

Newell, A.; Shaw, J.; and Simon, H. 1963. Chess-Playing Programs and the Problem of Complexity. In *Computers and Thought*, eds. E. Feigenbaum and J. Feldman, 39–70. New York: McGraw-Hill.

Quinlan, J. 1986. Induction of Decision Trees. *Machine Learning* 1:81–106.

Redmond, M. 1989. Combining Explanation Types for Learning by Understanding Instructional Examples. In *Proceedings of the Eleventh Annual Conference of the Cognitive Science Society*, 147–154. Hillsdale, N.J.: Lawrence Erlbaum.

Riesbeck, C., and Schank, R. 1989. *Inside Case-Based Reasoning*. Hillsdale, N.J.: Lawrence Erlbaum.

Riesbeck, C., and Schank, R. 1976. Comprehension by Computer: Expectation-Based Analysis of Sentences in Context. In *Studies in the Perception of Language*, eds. W. J. M. Levelt and G. B. Flores d'Arcais, 247–294. Chichester, England: Wiley.

Schank, R. C. 1991. Case-Based Teaching: Four Experiences in Educational Software Design, Technical Report, 7, The Institute for the Learning Sciences, Northwestern Univ.

Schank, R. C. 1990a. Teaching Architectures, Technical Report, 3, The Institute for the Learning Sciences, Northwestern Univ.

Schank, R. C. 1990b. *Tell Me a Story: A New Look at Real and Artificial Memory*. New York: Scribner's.

Schank, R. C., and Jona, M. 1991. Empowering the Student: New Perspectives on the Design of Teaching Systems. *The Journal of the Learning Sciences* 1:7–35.

Schank, R. C.; Osgood, R.; et al. 1990. A Content Theory of Memory Indexing, Technical Report, 2, The Institute for the Learning Sciences, Northwestern Univ.

Schank, R. C.; Ferguson, W.; Birnbaum, L.; Barger, J.; and Greising, M. 1991. ASK TOM: An Experimental Interface for Video Case Libraries, Technical Report, 10, The Institute for the Learning Sciences, Northwestern Univ.

Schank, R. C.; Charniak, E.; Wilks, Y.; Winograd, T.; and Woods, W. A. 1977. Panel on Natural Language Processing. In Proceedings of the Fifth International Joint Conference on Artificial Intelligence, 1007–1008. Menlo Park, Calif.: International Joint Conferences on Artificial Intelligence.

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

Sussman, G. 1975. *A Computer Model of Skill Acquisition*. New York: American Elsevier.

Waterman, D., and Hayes-Roth, F. 1978. *Pattern-Directed Inference Systems*. New York: Academic.

Winograd, T. 1973. A Procedural Model of Language Understanding. In *Computer Models of Thought and Language*, eds. R. Schank and K. Colby, 152–186. San Francisco: Freeman.

Winograd, T. 1972. *Understanding Natural Language*. New York: Academic.

Roger C. Schank directs The Institute for the Learning Sciences at Northwestern University, where he is also John Evans professor of electrical engineering and computer science, psychology, and education. The institute embodies Schank's concepts for conducting leading-edge interdisciplinary research in human learning, providing software solutions for education, and accomplishing the successful transfer of AI technologies from the university environment to the real world. Schank holds a Ph.D. in linguistics from the University of Texas at Austin and is the author of over a dozen books, including *Tell Me a Story*, *The Cognitive Computer*, and the newly released *The Connoisseur's Guide to the Mind*.

THE ELECTRONIC ADDRESSES FOR AAAI ARE AS FOLLOWS:

AAAI membership inquiries:	membership@aaai.org
AI Magazine subscriptions:	membership@aaai.org
AI Magazine editorial queries & letters:	engelmore@sumex-aim.stanford.edu
AI Magazine announcements:	aimagazine@aaai.org
AI Magazine advertising & press releases:	aimagazine@aaai.org
AAAI national conference:	ncai@aaai.org
AAAI Press:	press@aaai.org
Workshop information:	workshops@aaai.org
Innovative Applications conference:	iaai@aaai.org
Spring Symposium series:	sss@aaai.org
Fall Symposium series:	fss@aaai.org
Other inquiries and requests:	admin@aaai.org