AI: Expert Systems Pioneer Meeting
Session 2: AI Science and Technology 1955 to 1965

Moderators:
Burt Grad
David C. Brock

Recorded May 14, 2018
Mountain View, CA

CHM Reference number: X8652.2019

Table of Contents

Participants:

Name	Affiliation
David Brock	Moderator, CHM, Director Center for Software History
Burton Grad	Moderator, SI SIG
Brad Allen	Inference
Avron Barr	Aldo
Denny Brown	Teknowledge
Ed Feigenbaum	Intelligenetics/IntelliCorp; Teknowledge
Mark Fox	Carnegie Mellon
Peter Friedland	Intellicorp
Paul Harmon	Newsletter
Peter Hart	Syntelligence
Gary Hendrix	Symantec
Fritz Kunze	Franz, Inc
Doug Lenat	MCC/CyCorp
Brian McCune	Advanced Decision Systems
Peter Norvig	Google
Alain Rapaport	Neuron Data
Herb Schorr	IBM/ISI
Reid Smith	Schlumberger

Monte Zweben	Red Pepper/Blue Martini
Stephanie Dick	Historian, University of Pennsylvania (attended remotely using the Beam Robot)
Colin Garvey	Historian, Rensselaer Polytechnic Institute
David Grier	Historian, George Washington University
Thomas Haigh	Historian, University of Wisconsin at Milwaukee
David Hemmendinger	Historian, Associate Editor IEEE Annals
Hansen Hsu	Historian, CHM, Center for the History of Software
Dan'l Lewin	CHM, Executive Director
Paul McJones	CHM, Special projects
Len Shustek	CHM, Chairman
Ed LaHay	SI SIG, Meeting Project Manager

Burt Grad: This is the first workshop session, and we're starting with what happened in 1955 to 1965. What was interesting listening to all of you is that a number of you either worked with, worked for, or were taught by the people who did that work in the 1950s and 1960s—John McCarthy and the people like this. You're going to help straighten me out, because I don't know for sure, but it looks like Stanford University, Carnegie Mellon University, and MIT [Massachusetts Institute of Technology] were the three principal sources for the work that was going on. This was primarily, I would say, in the 1950s. It was academic work that we're talking about, but that's the story you all are going to help tell. I got a whole list of names of people and things that went on, but I'd like you to tell the stories.

Founders of the AI Field

Grad: Let me start with this. As I've been told, the work that John McCarthy did was at Dartmouth in the 1950s. None of you were working there or involved then. You come the closest, I guess, Ed, to working with somebody in the 1950s directly.

Ed Feigenbaum: Let me just lay out the 1950s. Alan Turing's paper in 1949, revised and published in 1950, was visionary, but Turing actually didn't really have a program that ran. He had a vision. [Allen] Newell and [Herb] Simon, January 1956, programmed the Logic Theorist as the first of the list processing languages.

Newell and Simon were critical at that point. They took the Logic Theorist to the Dartmouth Conference, and people at the Dartmouth Conference were blown away by the fact that it was actually running and proving those theorems.

Of the two graduate student theses of that period of time in AI, it was mine and a project called EPAM, which was part of the cognitive science part of AI, and Jim Slagle's thesis on doing calculus. Of course, Slagle's work got expanded by Joe Moses into Maximo and then symbolic then in relation of mathematics. It became a very big deal.

Grad: Were any of you involved in that? Tell me. Let's get you involved in this. Go ahead, Peter.

Peter Hart: Burt, I'm not sure whether the timeline quite makes the 1965 cutoff or just a slightly beyond that.

Grad: Be my guest.

Hart: To add to your three universities in this country, the other major center that was very early was the University of Edinburgh. Two of the principal people there were Donald Michie and Bernard Meltzer. They each were the chair and professor of their own departments and they had department names like the Department of Epistemics, for example. Do you remember that, Ed? The Department of Heuristics, I think, was another. It was elevated to the School of Artificial Intelligence

Grad: This is in the 1960s sometime?

Hart: I cannot tell you the exact start date. It might have been a little after that, but it was very, very early.

Grad: Doug, do you have an opinion on that?

Doug Lenat: Yes, I'm thinking it was the early 1970s.

Hart: Maybe early 1970s? It was definitely there by the early 1970s. But I think it might have started a little earlier.

Lenat: Maybe 1969.

Herb Schorr: I don't remember this, but I do think McCarthy and [Marvin] Minsky had a key role in the start of all this, and I think they did their PhDs in mathematics at Princeton. I'm just making a local plug for my school. But there were activities there, and they led to the MIT group initially. McCarthy then went to Stanford I think after MIT.

Peter Norvig: We should plug IBM, too, with [Arthur] Samuel and [Nathaniel] Rochester

Schorr: Yes.

Grad: What did they do? Talk about it. Go ahead, Peter, do you know?

Norvig: Samuel built the checkers playing program, one of the first machine-learning programs and one of the first successful game-playing programs. Rochester was working on theorem proving as well in the early days.

Grad: Were they publishing at this point in time?

Schorr: I'd say yes.

Grad: Were you aware of their work? That's what I've been going to asking. To what extent did you see what they had and how they convinced you?

Schorr: I think Samuel published. I don't remember about Rochester.

Norvig: No.

David Hemmendinger: Yes. We have a fact. Donald Michie started the Development of Machine Intelligence and Perception at Edinburgh in 1965.

Hart: Someone asked about Minsky. I will tell you that Minsky's thesis advisor at Princeton was John Tukey, who was the inventor of the Fast Fourier Transform. I once asked John long ago, "How do you make a Minsky?" He said, "You don't." And I said, "Did Minsky make Minsky?" He said, "Absolutely."

The Work of Marvin Minsky

Grad: But what was so special about what Minsky was and what he did?

Hart: I think other people here are better qualified than me to answer that, but the canon is that there are four founders of artificial intelligence: Minsky, McCarthy, Newell, and Simon. Is there a general agreement about that?

Hemmendinger: Yes.

Hart: Minsky did, you know, very early work on automata theory, and of course he had a long success of people at MIT. But other people can say better than I. Well, he's the first.

Grad: Well, that's a good question. What was he looking to do? How many of you are familiar with Minsky's work? How many of you feel competent to talk about it? Why don't you go ahead, Monte.

Monte Zweben: I had an opportunity to take a class at MIT with Minsky. I was working at the time, but I just wanted to take advantage of that opportunity. He was writing his book, *The Society of Mind*, at the time. It wasn't published yet, but the class had versions of every chapter. It was an amazingly interesting class.

From my perspective, having come from Carnegie Mellon, I found him to be extraordinarily interesting and a little crazy because he didn't let anything constrain his thinking. He basically would blurt out and write down anything that seemed plausible and would back it up, but yet he had this tremendously mathematical background, having been one of the first developers of what we would call neural networks today, with the Perceptron with Papert early on.

Neural Networks

Grad: Stop a minute. What is a neural network?

Zweben: I guess there are many people in the room who can talk about that. A neural network is a machine learning approach that tries to emulate what human neural networks do. It doesn't do it exactly

the same, but basically is a way to do classification tasks that can get through training better and better at classifying a model. It has been wildly successful in perception tasks and now has been commercialized through a variety of different platforms like TensorFlow and Caffé and MxNet to be essentially used for almost any classification task. It is one of the most popular machine learning methods.

Grad: Let's keep going. Is that pattern recognition, or is it something different?

Zweben: It's certainly pattern recognition.

Grad: Do you consider those the same or different concepts?

Zweben: I think that a neural network is one approach to pattern recognition. I think the old Deuter and Hart textbook was one of the best formalisms of that, and some of these techniques are still in that book. I think it's one approach to pattern recognition.

Grad: Ed made the comment before about his thesis was on cognition. Is that in contrast to the neural network pattern recognition approach? Go ahead, Peter.

Peter Friedland: I don't think it's in contrast. I think that there's sort of several different approaches that people have taken to AI and cognitive science. The sort of human hardware-based approach is trying to actually directly model human cognitive processes with something close to the kind of processing that goes on. That's what the people who developed the neuromorphic architectures for computing have done.

Grad: Neuromorphic?

Friedland: Yes.

Grad: What does that mean?

Friedland: It means based upon at least some view of how the human brain operates. For example, one of the closest things was DARPA's Project Synapse, which resulted in several new chips: one from IBM called True North, another from HRL.

Grad: Time period?

Friedland: Well, that's quite recent. The True North chip came out a few years ago.

Grad: These are recent things.

Friedland: Yes. But second to that is with the approach we're talking about: neural nets. Neural nets are based upon a view of how human information processing occurs, what's called the spiking neural model that Monte talked about, but there's no attempt to replicate in hardware or software what the brain actually does. As opposed to trying to run that on something that operates the way that people perceive the human brain operates, it's simply taking a model of cognition and using it as a way to do usually supervised classification, which is what Monte was describing and which has led to the current revolution in what's being called machine learning these days.

Schorr: As long as we're going, you're making my neurons come awake. There's a key paper in this by McCulloch and Pitts. McCulloch worked with von Neumann, and there was a theory of how the brain works. That was sort of, I think in my view, at least, the historian's, the beginning of neural nets and all the work. Then there's a guy named Rosenbluth or…

Norvig: Rosenblatt.

Schorr: Rosenblatt at Cornell, who picked that up. He started doing a lot of this neural net stuff.

Biological versus Cognitive Approaches to AI

Grad: Wait just a second. Do you all agree that the period of 1955 to 1965 was more or less focused on this kind of approach rather than a rules-based model? Peter?

Hart: No, no. No.

Norvig: No.

Hart: No, no.

Grad: Okay, one at a time.

Hart: What I would say is that focused on is too strong a term. But beginning with that McCulloch and Pitts paper, which was I think is called "What the Frog's Brain…"or “What the Frog's Eye Tells the Frog's Brain."

Hemmendinger: No, in theory…

Schorr: But that's not the paper.

Hart: Oh, a different paper? I'm getting it wrong?

Hemmendinger: Yes.

Hart: Okay. Sorry about that. In any case, I guess my main point is that a lot of us in the late 1960s and maybe earlier, mid- to late-1960s, were focused on trying to understand biological systems to get hints as to how to build computer systems. In the area of machine vision, for example, in the mid-1960s and the late-1960s, we were studying neurophysiology and the Hubel and Wiesel Nobel Prize winning work on the occipital lobe and how there are edge detectors and motion detectors. We were building computer vision systems based on those biological models.

Everybody said, "Well, planes don't flap their wings, so you don't build a bird." But we would say, "Yes, but the same principles of aerodynamics apply." I think that that undercurrent of trying to learn from biological systems without exactly mimicking them was really very prevalent in the late-1960s.

Grad: Monte, you're next.

Roots of Modern AI in 1950s and 1960s Research

Zweben: Yes, two points on that. I see three sort of main thrusts that we're talking about here. One is this sort of biological approach to trying to get glimpses of how the brain works and using that to get computers to do similar general-purpose reasoning things that we were just talking about. We also heard a few minutes ago about Newall and Simon trying to build search-based approaches and heuristic search-based approaches to do similar things.

What strikes me as pertinent to this particular conference or talk today is that the most successful AI today that we were just talking about, the machine learning aspects using neural network approaches or what we call deep-learning approaches, are more general-purpose algorithms that are trained on data. The thing that they lack, they lack the most, is what this conference is about, which is being able to see them with a tremendous amount of knowledge that's been extracted from humans. That hasn't been done yet, and the whole expert system that we've been talking about today and the history of the expert system is about codifying knowledge that isn't emergent in these kinds of machine learning systems, but more extracted from experts and codified and used in various different kinds of computing mechanisms to perform tasks very well. That strikes me.

Grad: Doug.

Lenat: I have a couple comments I wanted to make. One omission that I want to correct is extraordinarily important work done by Richard Friedberg at IBM in the 1956–1957 period of time. Essentially, he is the unsung father of machine learning whose experiments had a profound effect, at the very least on me and on my thinking. He did some of the early automatic program synthesis from examples work and so on. In many ways, his experiments failed, but they failed in ways which underscored the need for expertise and symbolic reasoning.

That brings me to my second point, which was touched on by Ed but I want to underscore it, which is the important insight, the realization that Herb Simon and Alan Newell had of the human information processing system being able to be modeled by digital computers. I suppose von Neumann had the same insight in a way, as a symbolic processing system. McCarthy in the late 1950s, 1959, 1958, 1960 period, drew an enormous amount of inspiration and, frankly, details from the first-order predicate calculus and other related work that had been done by Alonzo Church, before him by [Gottwold] Frege, after him by Bertrand Russell and others. In a way, if you look at the Lisp programming language, even the syntax is very similar to the first-order predicate calculus and the idea that you could model anything that humans are able to reason in this formal language.

Something as expressive as a natural language like English, and yet admit mechanical procedures for grinding through the logical consequences, was enormously important insight. In fact, the closeness of Lisp to predicate calculus is the reason why today we're able to hire philosophy graduate students instead of computer science graduates in order to populate our company.

The final comment I wanted to make was again on the tremendous importance of the Logic Theorist that Newell and Simon did in this time period, where they basically had the insight of having heuristic rules guide a theorem-proving process. In particular, one of the main heuristics that LT had was finding skeletal proofs, essentially taking a set of axioms, a set of rules and ignoring quantifiers, ignoring the scope and ordering of quantifiers, ignoring the variables, ignoring the negation signs, and trying to find proofs in drastically simplified axiom stubs that are left. Finding those proofs and then using those to plan and guide its proof for actual, real arguments for real proofs. That is, again, one of the big power sources behind my work still today.

Mark Fox: I just want to emphasize some of the points that have already been raised. The first and foremost for me is if I look at what goes on in AI today, I can trace its roots back to work being done in the 1950s into the 1960s. Whether it be neural nets with Perceptron, whether it be semantic nets and the whole semantic web, it is all rooted back then.

In one of the earliest papers, I think it was Herb Simon's student, which appeared in the book with a lot of McCarthy's students' work, that semantic network area was laid out way back then. Natural language processing was laid out. I think [Daniel] Bobrow's thesis, and it goes on and on and on. All the areas that we talk about today, that we use today, are rooted back there.

Evolution of the AI Field

Grad: What it looks like to me, and this is one of the things I've been fascinated by, the foundations for what we've talked about now in the 1990s and on are there and yet at 1965 or somewhere around there, that direction sort of stops dead.

Zweben: No. No, no.

Grad: My point is that there seems to be a disconnect and we will explore here is what happened and why that direction stops.

Fox: I don't see a disconnect. I see a continuing, evolving field.

Zweben: Yes.

Fox: We continue to build in spite of the market's perception of AI being in a winter or dead, or whatever the case may be. The work has not stopped; it's just gotten better. That was the first point I wanted to make is the roots.

Symbolic AI versus Mathematical Approaches

Fox: The second point I want to make is about neural nets. Now, the majority of people around this table are really rooted in the symbolic AI approach to reasoning as opposed to a more mathematical neural net approach. Not to say we don't understand it, but a lot of us grew up in that symbolic world, so we view things from a search perspective. We view it from a theorem-proving perspective, we view it from a logic perspective, etc. We tend to look down upon the neural nets, and we look down on it because we think that it is not rich enough an approach, a general enough approach that it can solve general reasoning problems.

The world thinks that automated vehicles are driven by neural nets. Well, it's certainly the case that recognition tasks are done, object recognition, etc., but when it comes to the actual control of the car, it's done by a more general AI symbolic reasoning system as opposed to a neural net system.

When you look at Siri, people think, "Oh, we must be using neural nets to do speech understanding." Well, of course we're using a variety of pattern recognition techniques to do the speech understanding, but when Siri answers your question, it's accessing an ontology insight and it's doing some type of very simplistic symbolic reasoning to figure out how to answer your question.

The systems that are in play today are a hybrid system of let's say neural net mathematical and symbolic techniques, so the idea that this symbolic approach is now being superseded by neural nets and all that is incorrect; it's rooted in a misunderstanding of the technologies necessary to solve these problems.

But I do want to say in defense of neural nets, and I'm not a neural net person, that there, that there is still meat up here, though. I do want to say that in the world of computational complexity, we learn early on about the notion of reducibility. Can you take one problem and reduce it to another problem? When we learned about P equal, when P or NP complete, we learned about reducibility. That is what neural nets are facing today. Can you take any of the more complicated, sophisticated, interesting problems that we want to solve and reduce it to a recognition problem? That's what the neural net community is doing today. And they're successful.

Grad: Peter.

Friedland: Yes, two comments. Going back to the historical period you're talking about. The third thrust I was going to mention was this thrust of studying humans not to understand how they're physically processing information, but to think about the reasoning that humans use. Probably the book to me that best encapsulated that that came out in the 1960s was *The Science of the Artificial*. It was by Herb Simon, right? Yes. That's a book that came out in I think 1966, 1967 and that really encapsulated to me why it was useful to me to be talking to people like Lederberg and other famous biologists to replicate what they were doing within a computer system. It wasn't like they were telling me which neurons were firing at any given time. We didn't put them under an fMRI [functional magnetic resonance imaging] or anything like that. We tried to understand how they symbolized their own reasoning. That's one important historical point.

The other point I wanted to make, very much along the lines that Mark was making, is one thing that people I think don't realize is that the explosive growth in deep learning, in stochastic machine learning, is much more because of advances on two fronts outside of AI, which is really and truly massive databases. In fact, didn't you once give a talk about how gigabits are very different from megabytes or something like that? It was about having a vast amount of information for training data. Peter can talk about it a lot better than I can. A lot of Peters in the room today.

Grad: Yes. I've noticed that.

Friedland: The second thing was the realization that you could specialize hardware, specifically GPUs, to do stuff. Those two advances weren't available when people like Neuron and Data Systems were forming companies. That's what's made the revolution far more than advances in the software of AI.

Solving Business Problems

Grad: What's interesting is that you argue that the hardware, not the software logic. I've heard a very academic discussion of who is working with real problems, whether a heuristic kind of thing, but to solve real business problems.

Friedland: Well, a checker player isn't really a business problem, but that was the first real application.

Grad: Do you consider that a business-type problem?

Friedland: No, I said it's not. Well, unless you're in the checkers business.

Schorr: Burt, what time period are you asking that question?

Grad: 1955 to 1965. I'm still in that period of time. My point is this, you've all been giving me a lot of the academic theory. Was there anything going on in business during the 1955 to 1965 period. I remember the beer truck stuff. In 1955, they were trying to route beer trucks around different cities. Was that GE? That's a heuristic approach. We were doing what Peter said. We were asking people, "How the hell do you do it?" and then we were recording that logic and trying to use it with the computers at that point. Was there other stuff like that going on where they were tackling operational problems?

Reid Smith: My comments are from the late-1970s and early-1980s.

Grad: You're later. Go ahead, Brad.

Brad Allen: You know, one thing I haven't heard mentioned here might be kind of on the border of that: I think of things that are focused on delivering functionality for the government as kind of a halfway house between academia and commercial interests. There was a lot of work in the 1950s on automatic translation that kind of hit a bit of a roadblock, but there were functional systems that were built that I think were deployed and used. There are other people here who probably can remember the names of the folks who were involved in that, but that was a thread that had some practical or potentially practical relevance to it.

Grad: Mark, I think you may have been next.

Fox: I just want to mention the person who I think best captured what happened in the Dartmouth workshop was Pamela McCorduck, who wrote the book, *The Machines Who Think*. I don't know if you'd seen that one.

Grad: Yes.

Fox: She interviewed everybody and then captured all that information. I don't know if this fits, but the second thing about practical applications is Herb Simon was also a professor in the business school. Quite frankly, he was probably the first to introduce the whole notion of simulating organizations. This was, I guess, 1960s work, and I think there were a number of theses on using computers to simulate organizations.

Grad: What do you mean by simulate the organization?

Fox: I'm at the end of my knowledge of exactly what that is.

Grad: Herb is next.

Schorr: So, two comments. One, you bring up machine translation. People forget that IBM at I think the 1964 World's Fair had a giant project where they were going to try and translate language in real-time.

Feigenbaum: Russian to English.

Schorr: Russian to English. These neural nets are old. This is, you know, 50 years ago, so that failed. It cost the Director of Research his job, but there was work on that. Further on, on language translation, at Yorktown we had John Koch and unfortunately Robert Mercer and his cohort. Mercer went on to work for Renaissance Technology. He made his money, which is now being used for nefarious purposes.

Brock: That's the same Mercer?

Schorr: Yes. Absolutely. I'm responsible for him. I hired him into IBM, but we never asked political questions. Smart. That was a statistical approach. We gave up on the symbolic approach, and in fact, I tried to get the guys to add a little symbolics into this stuff, but Jelinek who was running the damn project would have nothing to do with that. That's the symbolic junk.

The third thing I'm going to say is we passed over Marvin Minsky. His was the first laboratory where I saw robotics being demonstrated, and I think he pioneered, in my mind at least, a lot of that. I went up there. I was at Yorktown and I looked at it, I saw this thing going on. I said, "We can do that better. We're better engineers." We MIT guys produced products, which were used by IBM internally, but we never made any money on it.

Thomas Haigh: Yes, I think instincts are good for that in terms of thinking there is a connection between what you remembered from GE and what becomes AI. But what you need to remember is in the 1950s, there's no computer science. There are computers. There are scientists. There are people doing things with them, but there is no computer science discipline, so no computer science departments. There are no AI professorships.

What there is is this kind of cluster of methods. You're thinking of OR [operations research], which is coming out of World War II initially as scientists help the military do things in the war. Scientists can help businesses do stuff with mathematics and all these techniques in peace time. Then during, I think, the course of the 1950s it gets narrowed down to a smaller number of methods. But there are things like minimax, linear programming, and Monte Carlo optimization—you know, planning algorithms.

In the same cluster of ideas that the people whose work gets remembered in one way as the founders of AI, in another way, there's a lot of overlap between that and the founders of operations research. But operations research gets turned into a different thing that happens in different kinds of academic departments, done by different people. Herb Simon wins the Nobel Prize for Economics and is in a

business school, so that's maybe the most dramatic personal example of that: his legacy is going from a very coherent set of ideas in two different directions run by two different communities.

Those people don't identify with expert systems and AI and wouldn't be at this meeting. You'd have to have another meeting for them.

Brock: Yes, that's right.

Misconceptions about AI

Feigenbaum: Yes. Excuse me. I want to backtrack on two things that came up in this discussion. One, these are two things that constitute historical distortions or are simply bad history. The first is the general idea that everyone in this room knows; the way I put it in the 1968 IFIPS plenary session lecture was that AI is seen as the home of the no win guys because any time there's a win it gets plucked out of AI and it gets called something else. I'll give you the prime example from the period that you were asking for. If Alan Newell were still alive and you asked him the question, “Where did AI in the 1950s start?” he would say it started with a program by Selfridge and Dineen on discriminating Xs from Os. He would tell you that that's the single project that got him interested in AI.

Grad: Xs and Os like in tic-tac-toe?

Feigenbaum: Remember, we're in an era where what were computers doing? They were doing numerical calculations; they were doing H bomb simulations and ordinance stuff. To do anything that touched on a cognitive act like discriminating two letters was news.

Anyway, it got worked on by several AI people who started companies, and by 1965, it was a whole industry called OCR [optical character recognition] that was all by itself and selling its stuff to the world.

Grad: You consider they were using an AI capability to do that recognition.

Feigenbaum: No. What I'm saying was that in these practical applications there are AI roots that go very deep, and that has carried itself through into the present day. Although, these days, AI is getting too much credit for things. Any software that runs and does an interesting job is called AI in the newspapers.

The other thing I wanted to mention was that there is this historical fiction that there were these AI winters. There were no AI winters. The first declaration of an AI winter was by Pat Winston at MIT, and he was declaring that 1965 through 1970 was a period of AI winter. Well, he just wasn't paying attention. That's the period in which expert systems were born. In the 1990s when the interest in expert systems faded, that's the era in which all this machine learning came back.

Commercial and Operational AI Applications

Grad: Here. Let me tell you again, I'm a business man. I worked for companies. I was trying to solve real problems. I was at GE in the 1950s. We were trying to find dispatch rules to run stuff through job shops. You needed rules. So, how'd you do it? That's when we had simulation programs we ran, modeled our factory and put together various dispatch rules for testing. Would you consider that part of this heritage, or was that a completely different direction? Peter?

Hart: Getting back to your original question of what commercialization was there in that 1955–1965 era, I think it would be difficult to apply practical commercial applications that today's community would consider, if you will, genuine artificial intelligence. It's partly a definitional matter of how raw you want to make AI. If you go back to that particular decade, I think you'd be looking more at what Ed was pointing out, at various kinds of applications of pattern recognition and pattern classification and image analysis.

For example, the optical character recognition that Ed mentioned was barely becoming practical then. I will tell you that in the early 1960s, Dick Duda and I wrote the first program to improve OCR accuracy by using surrounding contexts, for example. You might ask today, "Well, I don't know, was that AI?" Well, whatever definition you like. But I think you would have to be looking a little more broadly than either symbolic processing of the sort that Mark was alluding to or neural nets, that have come up multiple times, and have to have a little broader view of what you want to consider AI.

Grad: This is exactly what I'm trying to get a picture of. There were operational problems in running factories and designing these things. Those were things at GE we were very much focused on. We had a very strong operations research group there at GE during the 1950s. I was working on building an automated factory, and I wasn't thinking about symbolic logic. I didn't have any idea about any of these things that you're talking about, and yet we actually ended up with solutions to these things that we used in running our factories, designing our things, doing engineering design. Who's next?

Allen: It's fascinating to me that this theme has cropped up as strongly as it has as quickly as it has at this meeting: struggling with this false dichotomy—for example, as Tom has mentioned, the notions of practical approaches in operations research and the things that were more heuristically based and symbolically based. I wanted to chime in at this point because I have an anecdote since this is a history kind of focus thing and it's the most mortifying experience of my entire career, which actually, Monte is responsible for.

Monte rooked me into getting up on stage in front of the entire AAAI conference and debating George Danzig about whether or not linear programming or heuristic-based, rule-based systems were a better approach to doing optimization. I did the best that I could, but it goes to the point that we've constantly been changing our minds over this entire period going back to the 1950s about what's AI and what's not AI.

Some days we lay claim to these older techniques, statistics and OR, and so forth. Other times we say this is different in some fashion. But as Mark pointed out earlier, I think, it's the blending and the mixing of

all this that really constitutes progress. And to Ed's point, that's never really gone away; it's just various aspects of it have been under the spotlight at one time or another.

Defining AI

Grad: We obviously have to break at some point, so I want to cover what we can. This is a period I thought we weren't going to get much about, and I'm just blown away. Who's next? Monte, you get another 30 seconds.

Zweben: I think, this debate about what's AI and what's not AI has plagued us as a community, especially back in the 1950s and 1960s. Mark just whispered in my ear, “How many computers did we have back then?” We didn't have that many to work on as a field. If we strictly stay within this period of time and what you mentioned, Burt, about dispatch rules, I think all of the work that was done in advanced planning and scheduling that I did and that Mark did, both at NASA and in Westinghouse and then later in private companies, all based itself on some of the dispatch rules that were developed in the 1950s and 1960s that we expanded to use symbolic heuristic methods to try to incorporate human guidance into these solutions. Then we took it another step and used Simon and Newell's approaches to try to put a search framework on it. And then we used some of the MIT approaches to try to build constraints as a way to guide that search.

So as Mark said, I think it really is all very incremental progress that came from these 1950s and 1960s that people were starting with, and I think that when we start talking about the 1970s, 1980s, and 1990s you'll start to see that.

Reid Smith: My thesis supervisor was Bruce Buchanan, and I think as a way of putting me in my place one day he said, "You realize that Fortran was considered automatic programming in the 1950s. Yeah?" I'm reading here a little article that is called "Automatic Programming: Properties and Performance of Fortran Systems I and II" by Backus in 1958. Brian, you know two of the students from your or Cornell's Ph.D. making machine used to work with me in Schlumberger. One of them is still doing program synthesis, and I think it would be fair to say that she can trace her roots back at least to the 1950s.

Rand and AI Work in the 1960s

Avron Barr: Quick question for Ed: I know there were events at Rand during this period. Was there ongoing AI research work at Rand too?

Feigenbaum: For those people who don't know, Rand at that time was a primarily Air Force-based think tank in Santa Monica, California. Monte was saying there weren't that many computers in the world; it actually had one of the first. It was a copy of the Institute for Advanced Study machine, called the Johnniac, which is downstairs, believe it or not.

Friedland: That's not a computer. You never used it. I tried to use it. What a mess.

Al Newell was a Stanford undergrad and he was a student. George Polya was here, and he was much influenced by George Polya, but he actually had to get a job. He got a master's degree at Princeton, but he had to get a job, and he got a job at this think tank doing computer code for training of radar operators for operating the big, new radar systems that we were building to detect Soviet bombers. That's where he met Herb Simon because Herb was a consultant. They really were a great pairing in the way that they thought about the issues. So, Al asked the boss, Paul Arbor, the head of the Numerical Analysis Department, if he could be a one-man office in Pittsburgh, a one-man Rand office in Pittsburgh, and the answer was yes. So, Al moved from Rand to Pittsburgh, but he was still a Rand employee.

The other person who's more or less unknown in all of this is Cliff Shaw. Cliff is this genius programmer at Rand who was working constantly with Newell and Simon in Pittsburgh to get these things running on the Johnniac. They would communicate by teletype with each other mostly or by phone. Cliff was writing these first few versions of the IPS, which led to eventually to Lisp, McCarthy's Lisp.

Now, the question is, are there other things going on at Rand at that time that we could get into this conversation here, that we could call AI? Nothing jumps to mind, but there were some things that we do now that we just take for granted, touchpads on laptops and so on, that was an ARPA project at Rand, but it wasn't really AI.

Then AI got reborn again at Rand, but that wasn't in the period of 1955 to19'65; it was later in the 1970s.

Grad: You mentioned to me a thing called theorem proving that they did at Rand. Was that at that time period, or was that later?

Feigenbaum: No. The theorem proving was Newell, Shaw, and Simon Logic Theorist papers.

Grad: I see, okay.

Feigenbaum: Proving theorems in propositional calculus. So, Rand eventually came back in the game. For example, they had an expert system shell, the Rand RITA, the Rand Intelligent Terminal Agent, I think. I'm not sure about this, but they may have done some expert systems for the Air Force.

Grad: This is the later period, though?

Feigenbaum: Later period.

Rule-Based and Scheduling Approaches

Fox: I want to address the points you raised, Burt, about the existence of dispatch rules and what the distinction is between that rule-based approach that comes out of the OR side versus rules out of the AI side. Dispatch rules are primarily used to provide a prioritization of things in a queue. An order is a queue. What's lacking from the use of dispatch rules is the whole notion of a problem space in which a search is being performed.

I was looking back, and I was trying to trace back the whole production system model to human problem solving. Newell, Shaw, and Simon wrote a paper in 1958 on the whole model, I believe. For us, search in a problem space means that we have this space of states that we're going to explore. We have ways of taking a state and generating new states. There are various ways in which you can move through it—backward, forward, middle, out—which doesn't exist in the OR dispatch rule world.

Brock: Right, right.

Fox: Okay. It's a totally different way of looking at problem solving that the simple rules that existed as dispatch rules that manufacturing were using. Unless you understand that perspective, it's difficult to distinguish because they're both called rules.

Grad: But that was just exactly my point. Those of us who are doing job-shop scheduling, there were the people who did theoretical work, a fellow named Alan Rowe—he went out to Rand Corporation. He had these very elaborate factors you would take into consideration, how you would balance something and you put different weighting on different things. We simply said, "Try a simple thing. Do the shortest job first and once a day do the one that's been in the queue the longest." Which do you think won?

Fox: An SPT is always the one that people like. EDD, earliest due date, is another one that's used. The simplest rules were the ones that were being used. Those of us who did scheduling in manufacturing, which includes Monte and Brad and I don't know who else around the table, were well aware of that, but those rules were designed as ways of sequencing jobs in front of a machine. There was no notion of search in a problem space, and that's what really distinguished what we were starting to do in our approaches to problem solving.

Now, we weren't just simply, "Here's some rules. We're going to generate a search in a problem space." We were combining techniques. Constraint satisfaction techniques were very, very important in what we're doing, which grows out of our consistency, constraint satisfaction, which goes back to the early 1970s in that particular case. But there's a real difference between OR dispatch rules and when we use the term "rules" in …

Grad: You're using OR. I'm just talking heuristic stuff.

Fox: Right, but the point I'm trying to make is, those rules are simply ordering rules. They tell me how to sequence jobs in front of the machine. Whereas when we refer to heuristics, we're referring to

using them to guide us in a search in a problem space, and that was really the key behind expert systems that Ed created, that Newall and Simon created. That is very different in a way of thinking about problem solving.

Friedland: Let me go back a little bit to the historic area you're talking about. I had to cheat and read to get the dates right. Thank God, we have all the world's knowledge here.

So, before the term AI was ever used, there was the term cybernetics, which dates back to the 1920s. Now, cybernetics was focused originally more on feedback loops and understanding human regulatory processes, but actually this notion of neural nets came from the cybernetics folks. The actual date here is the split from AI… let's see. Originally, cybernetics and AI were considered sort of the same, but AI was founded as a distinct discipline of the Dartmouth workshop and for some an uneasy coexistence, AI gained funding and prominence. Consequently, cybernetics sciences such as the study of artificial neural networks were downplayed, the discipline shifting to the world of social sciences and therapy, but the first mention of neural networks actually came from von Neumann.

Hemmendinger: Yes.

Friedland: Even before we had an AI field, that notion of modeling human processing was sort of accorded to the cybernetics field of Norbert Wiener and before him von Neumann and the like. If we're talking history, we shouldn't forget that part of our history.

Allen: Getting back to the theme of things that came out of AI work in that time period that had ramifications later, I think it's probably worth underscoring the fact that the things that Newell and Simon and McCarthy were trying to do pushed them to drive new ways of dealing with system software in general. The work that McCarthy did on timesharing systems and some of the stuff that came out of Newell and Simon and Rand and so forth had an impact later in terms of the development of the classic systems that later got led to the operating systems we all know well and love from that era, where people were trying to share scarce computing resources in a way that they could make progress on these problems of AI. There was an effect with regard to that, and I think that's worth calling out.

Emergence of Expert Systems

Grad: To close this particular session, my question is, what to your knowledge were the first what you would have called an expert systems application? Were there any of them in the 1950s that you're aware of? Peter?

Hart: I think the answer is no, if you accept the definition that most people here would use for expert systems, but I will tell you that my long experience has been that people came up to me repeatedly from that much earlier period and said, "Oh, we were doing the same thing; we just didn't call it an expert

system." If you look at what they actually did, they were trying to encompass expertise just like any computer model and embody some sort of expertise, but with very different technology.

Grad: That's exactly the thing I'm getting at. At GE, it happened to be we were trying to do automation in a factory.

Hart: Yes.

Grad: In a very narrow area, a 39-frame motor or a particular type of meter, how did the engineers decide what the dimensioning was on that particular one they got? Because they were all custom built. How did they decide? We went there and did what you guys are talking about. We interviewed the engineers. They had contradictory views, but we said, "Okay, if we put this together, here's what we will do. And we can now automate it, the design of that particular motor or that particular meter."

I'm sure there were other things like that. There was work going on at Hunt Foods and Orren Evans also came up with decision tables at the same time as I did as a means of recording that logic, that reasoning.

Were there other cases going on in the 1950s and 1960s of that kind? There must have been. You weren't aware of them, probably. I don't think any of you ever knew what we were doing at GE, and we certainly weren't aware of what you were doing. We were totally disconnected, which is fascinating to me, that those were two completely separate things. All this wonderful work is starting to go on in the 1950s and 1960s. I mean, I knew of Newell and Simon. Their work wasn't relevant to me

Emergence of Machine Translation and Natural Language Processing

Allen: Forgive me if I should have heard it, but I personally didn't. I was interested to know the beginning of the story of natural language processing in this decade, 1955 to 1965, if someone could speak to that. How did that line of research emerge in this period?

Grad: IBM was doing stuff.

Schorr: IBM did it, but it didn't work, so they threw it out the door.

Brock: I mean, it's not synonymous with machine translation, which was what we were talking about before. Is it? Or is it?

Allen: No. No, not at all. Not at all.

Brock: So?

Allen: When was Winograd's dissertation?

Hemmendinger: Who was doing it in the Russian to English translation?

Fox: That early 1970s, I want to say.

Zweben: Yes, Terry's later.

Schorr: It was before my time.

Grad: In the 1960s there was certainly work on machine language translation at IBM, very heavily.

Zweben: Chomsky was doing work early on too, wasn't he?

Brock: Yes.

Grad: There were a lot of people working on that at that point. Were there other of those kinds of problems being worked on? We talked pattern recognition. Natural language translation was one of those kinds of things. What are some of the other problems that were being looked at then?

Brian McCune: Robotics.

Grad: Okay. What else?

Brock: Wait, what period for robotics?

Grad: 1955 to 1965.

Hart: There was very little robotics of the sort of not just the mechatronics; anything with any smarts in it, I would say there was almost nothing. There was Gray Walters Turtle. There were one or two other things of that ilk that looked for an outlet to plug itself into or just had a single photodiode to go toward the light or get away from the light.

Grad: Handwriting recognition, I know, was being worked on. The OCR stuff people were trying to do at that point in time.

Hart: Yes, OCR definitely. Another big area was photo reconnaissance for the military. They funded a lot of work to try to get information from aerial photographs.

Grad: Is that a pattern recognition kind of issue?

Hart: A very hard version. Yes.

Expert Systems and Inference

Grad: We're now going to move into expert systems. We're literally on 1965 to 1975 time. I'm trying here to lay the foundation. There was a bunch of companies formed and Ed, who's not present, to deny what I'm saying, helped to start a number of companies, but a number of others started in parallel. Where does expert systems technology start? Who starts it? Is that all here at Stanford? Is it at other places in parallel? Who's doing expert systems?

Then we will finish up that session and go into the companies after that. Let's go with that model. What do all of you, any of you, consider the first of the expert systems projects that get started? Where is the technology? You wanted to talk before, Avron.

Barr: Actually, it's a good segue from what I was saying. The problem of coding human heuristics into software is a pretty general problem, and people were doing it from the very beginning of software. You have to. That's all we have to go on is how the humans solve the problems. What I think, and I'm proposing this as a definition, was the beginning of expert systems technology was when people started building tools to make that easier so that they could debug the knowledge separately from the rest of the code. They were developing methodologies for actually acquiring knowledge from human people in a systematic way and using some of the technologies from artificial intelligence, like knowledge representation and inference mechanisms, to embed in those tools to separate out the different aspects of the technology, of the tools.

Grad: What do you consider the key elements in an expert system?

Barr: Well, I'll make a pass at it, but I'm sure people will have better ideas. Certainly, knowledge representation is one of them, the ability to use that knowledge to make inferences is another.

Grad: What does that mean? Say what you mean by knowledge representation.

Barr: Data structures in the computer that can be used to make human understandable representations of what people know. Categories, subcategories, rules.

Grad: Classification?

Barr: Things like that.

McCune: And that can be reasoned about by a computer.

Barr: That then can be reasoned about by what I was going to call a separate mechanism, a separate technology.

Grad: Well, what is it? What would you call the second one?

Barr: Inference. Inference.

Friedland: An inference engine.

Barr: Yes.

Grad: What does inference mean?

Barr: Taking two pieces of knowledge and making some new use, correct knowledge from them.

Grad: Is it reasoning?

Barr: Well, yes, absolutely.

Grad: Logic?

Zweben: Not necessarily logical.

Grad: Oh.

Lenat: Okay, at the risk of getting some disagreement here, I would say the main distinction, when the bell was rung, between pre-expert systems roots and real expert systems was the moment when people realized that one could incrementally approach competence at a task by representing explicitly, symbolically, expert rules, one after another, and getting the nascent program, the developing expert system to do less and less awfully, better and better at that task. That is the conceptual leap to say instead of flow charting a program with well understood logic from beginning to end and temporally from beginning to end of the project, we're going to jump and take that leap of faith and believe that if we extract rules of thumb, rules of good guessing, rules of good judgment from the experts, then gradually, incrementally, those programs will get better and better. They'll get better and better because at any given moment, any conclusion that the program comes to can be justified by looking at step by step what the line of reasoning was. The expert looking at that line of reasoning in case the program got the wrong

answer will be able to say, "Oh, yes. Here is something that I forgot to tell you, or I stated wrong, or I stated in the wrong context" or something like that.

Grad: The words I've heard used were adaptive learning. Is it that?

Lenat: No.

Grad: That is not what you mean?

Lenat: Completely unrelated to that. Think of it as incremental approach to competence by believing that one could get programs to declaratively represent these rules. In cases where you weren't at the salable or demonstrable or reliable level yet, the programs could get better and better by having experts look at the trace of rule firing behind any wrong answer and incrementally improve that program.

Grad: You would have an initial logic or initial rules and then you would keep reexamining the rules. Is that what you're saying?

Lenat: Well, you keep running them, and empirically, the programs would get better.

Feigenbaum: Burt, let me tell you a story. The story's fill in here for abstractions. We're working with the chemists on building the knowledge base for the DENDRAL system in 1965. We were focused on amino acids because that's what Joshua Lederberg knew, the mass spectrometer of amino acids he was working with.

When we ran out of Lederberg's knowledge of mass spectrometry, we tried to recruit the head of the Stanford Mass Spectrometry Lab to be interested in our problem. He was a friend of Lederberg's. We took him up to the Stanford AI Lab where the thing was running. This was Carl Gerace. We showed him what the program was capable of and said, "Okay, now why don't you give it some problems."

He gave it a problem completely outside the domain of amino acids, and it flopped and it didn't do anything. It didn't even know what to do. That's the incremental. It had no idea about alcohols or ethers or thioethers. Lederberg and I had planned this carefully, and we turned to Carl and said, "Okay, Carl. What is it that you know that it doesn't know? Why is it stupid? What do you know that it doesn't know?" He came out with it. Typed it in.

Grad: Could you add that to the program, or did it have to be a separate program?

Feigenbaum: What do you mean?

Grad: My point is, focus. You worked on a particular thing, amino acids. Could you take that same framework and add others, or was it effectively a separate set of logic that used the same tools to work with?

Feigenbaum: All the same thing. Not separate.

Lenat: Same inference engine, more rules.

Grad: Ah.

Feigenbaum: A knowledge base in these chemical structures will never be full because the knowledge of chemistry is vast. You don't want it to be too small because it isn't useful if it's too small. But you keep adding knowledge to the knowledge base. It gets smarter and smarter.

The Inference Engine and Inductive versus Deductive Reasoning

Grad: You speak of the inference engine. Was that part of the initial expert systems thing?

Lenat: Yes, absolutely.

Denny Brown: All right, Burt, you asked about representation. I think that's important. Well, we talk about representation, inference, and control. On the inference side, thinking in terms of how humans do things, we talk about deductive reasoning and inductive reasoning, and they seem like different things. Ed mentioned earlier during his initial conversation about the change from general problem solving to intense knowledge based. That I think is sort of the key bit that distinguished the change from general problem solving in AI to expert systems as a particular subset which was general inference engine, so a general deductive capability of some kind, augmented with specific rules about a particular domain, a particular problem area. The ability then to have that general deductive reasoning model inductive processes like DENDRAL.

Grad: Help me here. The difference between the inductive and deductive: the inference engine is deductive?

Brown: Typically, the inference engines that we talked about in the expert system world were deductive reasoning methods.

Grad: What is inductive then?

Brown: Inductive is where you're working from data to generate more conceptual generalizations.

Brock: States, yes.

Knowledge Representation and Heuristics

Fox: I'll give my own personal view, having started in graduate school in 1975 and watched some of the earlier stuff or read some of the earlier stuff. I take a narrower view of expert systems, so when I think about expert systems, I don't think about reasoning with the experience, I don't think about rich knowledge representation.

That type of stuff was ongoing within AI at the same time, but when I think about expert systems, for me, it started with the recognition of two seminal systems. One was MYCIN, and the other one was Xcon/R1. What's seminal to me about those is MYCIN demonstrated within the field of medicine some very interesting diagnostic capabilities based upon rule representation. Xcon/R1 demonstrated at Digital that you can actually solve a real problem and configuration task in the manufacturing or engineering world that Digital was having a tough time doing because a lot of their systems were bespoke, they were custom systems.

Those two systems together demonstrated to me and I think to the world that there was something called expert systems, and they created a validity for this technology beyond the academic environment that didn't exist until that time.

The key thing in my mind in understanding what an expert system is from those two perspectives is the Newell and Simon notion of the problem space: there are states, which is your memory; there are rules, which transform states into new states; and there are various ways in which you can search it. What delayed, I think, Newell and Simon's use of rule-based systems in terms of practical applications was their interest in developing a model of human problem solving as opposed to saying, "Hey, we're going to go out and solve a whole bunch of real problems with this."

On the other hand, my interpretation of what Ed and his team were doing is they were working on some really interesting problems out there and taking rule-based concepts and applying them.

Now, is there a relationship between MYCIN's approach and an Xcon approach? The answers are both rules-based, but they're structured very differently. To me, the MYCIN approach is a subset of what occurs in the general production system and problem-solving approach in that it recognized a particular structure to the rules which could then be translated into a network and in which you could take advantage of in terms of how you problem solve with that network. Basically, they come out of, in my mind, similar roots.

Expert Systems and Problem Space

McCune: Well, I just wanted to go back to where expert systems came from. Before expert systems, everyone was dealing with general problem solving by theorem provers, A* search, pattern recognition algorithms, neural nets. The problem was that you had this very general technique, you could feed it some data, it would sort of turn over, and you could publish a paper. That's really nice, but it wouldn't solve anybody's problem. The insight into expert systems was to get hyper-specialized and to constrain this massive search space into a much smaller problem that was feasible.

I take one exception to what Mark was saying in that I think DENDRAL and Maximo were the first expert systems because they noticed the fact that you had to get expertise from human experts, encode that any way you could. In fact, in the case of DENDRAL, I think it was really a generate and test algorithm.

Feigenbaum: Yes.

McCune: Not a production system to improve or something else. And that was the right way to do it for that problem.

Feigenbaum: The knowledge of chemistry and the knowledge of mass spectrometry were represented in if-then rules. That's how we coded knowledge.

McCune: But you had a generator.

Feigenbaum: That's irrelevant. We're talking about the knowledge. The generator is in the reasoning engine.

McCune: Yes. That's what I mean.

Expert Systems and Heuristics

Friedland: I think one word that hasn't come up yet that I'm really shocked about is heuristic. One of the reasons why Ed's group was called the Heuristic Programming Project, at least when I joined it, was because most of the rules gained from experts were not rules of absolute logic. There was not, “If A, then B, always true in the universe.” It was expert knowledge which said, “This is a good guideline, a heuristic, something which helps you solve a problem but is not formally guaranteed to be correct.”

That was key in extracting expert knowledge, that whether it was represented as rules or some other format—in my case, using both rules and what we called skeletal plans, but we'll go into that later. As with any expert, whether it's in medicine or chemistry or whatever, the rules they provide are not infallible, but they have expertise. They're expert guidelines to solving a problem. To me, that was very key.

In fact, once in the heyday of expert systems, I heard some junior person talking at a conference about what he wrote: I think he was calling it a deterministic expert system, which sort of struck me as a

contradiction in terms because expert systems, like human experts, aren't guaranteed to be infallible; they're guaranteed to be better than a novice trying to solve the same problem.

McCune: That's a good point.

Grad: We differentiate between linear programming, with which we get a correct solution—we know it's the best solution—versus a heuristic solution—one that works pretty damn well but isn't guaranteed to be optimal.

Friedland: That's exactly the right point.

Feigenbaum: Heuristics are the art of good guessing.

Haigh: All right, from my historian viewpoint, I've got an observation and a question. The observation is that I've also written about the history of database management systems, which were starting out in the early 1960s around the same time. I think there may be a comparison there: coding application by application and expecting application programmers to do it is extremely hard in terms of what DB indexes had to do for searching, indexing, hashing, data retrieval methods, data independence. Putting that out into standardized code would be extremely hard to write but would be usable for cross applications and compartmentalizing that from the application of specific knowledge. That's the observation part. I guess this is something that's happening probably in several areas of software and computing around this period.

The question is, clearly people have a strong idea of the core components of an expert system, but what we haven't talked about is why it's called an expert system and the work that that does. That's something that I think historians would like to have covered here, because the other things you can find in a textbook, but we have you all together. It's not called a “general purpose heuristic machine” or a “human readable rule-based inference engine,” right? It's called an expert system, and that's quite a bold claim. It's not a novice system. It's not a journeyman system. It's an expert system.

That term took on, so it must have been doing work for members of the community. I assume, but I don't know for sure, that Ed Feigenbaum thought of that term, so he can tell us why he thought about it. But I would take it from the rest of you as, do you remember the first time you heard that phrase "expert system," and were you excited by it? Did that as a phrase, as a concept open up doors for you and help you do the things that you wanted to do?

Hart: I want to follow on your comment and Mark's. I heard one of Ted Shortliffe's very first MYCIN talks at Stanford, and I was very impressed by it. I went out and got some money from the US Geological Survey to work on mineral exploration, which is an extremely hard problem for humans, by the way. But as it turned out, I had misunderstood about two-thirds of what Ted had said, and I disagreed with the one-third that I did understand.

So when I developed with Dick Duda this Prospector system that I've mentioned, the knowledge representation was entirely different, and the control regime was entirely different because I thought I was imitating Ted and his certainty factors, which were kind of an ad hoc thing I disagreed with. I was an unreconstructed Bayesian so we used Bayesian decision theory as the uncertain or probabilistic reasoning component. We presented that at a Hawaii conference, and I haven't thought about this for 40 years from that day until this. Ten seconds ago, it occurred to me that Doug wrote a review and said, "Prospector is a system to watch." And I said, "I love you, Doug."

But fast forward, like a good four or five years to a mature Prospector, and I would say that it had two really important consequences—the historians will have to check this. The first is that it was the first real use of Bayesian reasoning, at least that I was aware of. At that time, I was very good friends with Judea Pearl, and we used to play guitars together and such. He went off and really formalized that entire area. Wrote a wonderful book about it and really made that a very important part of AI for a very long time to come, I think to the present day.

The second bit consequence is that we published an article in *Science* magazine, which was picked up by the mass media, *The New York Times*, and everybody else, with the headline, "Prospector Discovers a Mine." Though that wasn't quite literally true, it was pretty close. I was hoping that Paul would be able to comment on the extent to which that got people excited about the real-world practicality of expert systems.

Paul Harmon: Not in the 1965 to 1980. I'm only post-1980.

Hart: That *Science* article was right around 1979 or 1980, as I recall.

Harmon: Okay, well, it was before I was focused.

Hart: A lot of things came from that *Science* article.

Defining Expert Systems

Zweben: From my perspective, I think we're conflating on the word "expert system" with two separate things. First and foremost, I think the big idea of not having a general-purpose problem solver or sort of emulating human thought and performing some real-world tasks and codifying human knowledge in some way to perform some task is a great definition of an expert system. But then there's the computer science part of it where there was a bifurcation, in my view, of expert systems in how to implement them. One was very symbolic, and there were rule-based systems that had key value pairs and did pattern matching. OPS 5 developed from that, and there was a whole boatload of companies that you guys started that started to build out those systems for that.

Then there was your work on explicitly representing measures of uncertainty in these systems, both of which were applicable to real-world problems, both of which were really coming from one fundamental failing of computer science in that we were all programming in programming languages that had very strict rules of functions, of control, and it was hard to codify this human knowledge in a programming language.

These two different approaches of explicit measures of uncertainty and propagating uncertainty metrics through some sort of network, and there were a few different approaches—Bayesian, non-Bayesian—that emerged. Then there were these other approaches of packaging this knowledge which were more symbolic. But really, this was driven from a lack of a programming language to codify that knowledge and expert systems started to get defined by these different approaches.

Summarizing, as I see it, an expert system is a codification of a very specific task that a computer can do. And then there were different camps that programmed that in different ways, symbolically and numerically.

Norvig: I wanted to react. Somebody said that the operations research techniques, linear programming, give you an exact answer. That's right. It gives you an exact answer to an inexact and wrong model.

Friedland: That's exactly right.

Norvig: George Fox said that all models are wrong but some are useful, and I think the expert systems were in part a way to shift that balance to say, “Let's have a model that's less wrong. We'll give up and we'll have some reasoning, and maybe we'll put a little bit of the inexactness into the reasoning rather than into the model.”

Grad: We'll go eat, and we're going to come back and continue this after lunch. Just to summarize, what all of you have said is that fundamentally, capturing the knowledge of people who are experts in a particular field, like Josh Lederberg, and combining it with the computer science type people who know how to deal with these things, that provides the basis for what you've been calling an expert system.

END OF THE INTERVIEW

