The History of Computer Chess: An AI Perspective

Computer History Museum Presents

Moderator:
Monty Newborn

Panelists:
Murray Campbell, Deep Blue Project, IBM Corporation
Edward Feigenbaum, Stanford University
David Levy, International Computer Games Association, International Master in Chess
John McCarthy, Stanford University

Recorded: September 8, 2005
Mountain View, California

Total Running Time: 2:05:57

CHM Reference number:

Monty Newborn: Well, it's my pleasure to be here. I'm thrilled to be a part of this great evening and this great function that the museum has put on, and I would like to begin by just thanking the few people that I've worked directly with. John [Toole], thanks. I had a list of people to thank, but John beat me to it, but I would like to thank Dag Spicer who's sitting in the front row and Kirsten-- where's Kirsten Tashev? They have been terrific. They're going to do some great things for this museum in the future, so I'll wait and see what's next.

Our agenda is: I'd like to make a few general remarks. I'm going to then introduce the panel. Each panelist is going to take five to ten minutes of discussing various aspects of computer chess that they've been involved in and would like to discuss. I may ask them a few questions myself when we're finished with all of them. Following that, the audience will be invited to ask questions and we will have microphones in this row and in this row, and if you want to ask a question please come down into your row and wait your turn. Each person will have one question, and one question only, because I think we'll have a lot of questions. This is a subject that provokes a lot of questions.

The exciting thing about this computer chess is the people that have been involved in it, and I'd just like to go back through it just a little bit. For those of you that are computer scientists that are studying all the various areas of computer science, computer chess has touched upon many of the leading names that have gone into other areas as well. We can talk about Babbage, who discusses analytical engines, but he managed to work in a little bit of discussion about how computers might play chess in his writings. We go on to Alan Turing who of course was famous for his work in the theory of computation, to Claude Shannon, to von Neumann who discussed the minimax algorithm and game playing algorithms, to Norbert Wiener who speculated on how computers might play, to Nobel Prize winner Herb Simon who won the Nobel Prize not for chess, but for his work in economics, to John McCarthy (who's hiding at the end of our board) who pioneered the programming language Lisp and was one of the first contributors of a real working program, to Donald Knuth (who is sitting somewhere in the audience) who discussed the alpha-beta algorithm and it was sort of the bible of what to expect from the alpha-beta algorithm as programs progressed, and to Ken Thompson who was- not clear whether he's more famous for his work in UNIX or C or with his chess program, Belle. I don't know if Ken is here tonight. I'm hoping he is, but maybe he's not. So these were the thinkers and a little bit of the doers, but of course the doers came shortly after and there were some terrific doers, from Alex Bernstein at IBM who developed the first program, to Alan Kotok who worked with John McCarthy on the program that wound up playing against the Russians, to Richard Greenblatt at MIT, to David Slate and Larry Atkin, to Thompson again -- I can't get away from Thompson -- to Robert Hyatt, to Hans Berliner, to the Spracklens. Now if you talk about chess, it seems to be a man's world, but Kathe Spracklen was the exception to the rule. Cathy was a tiger and was the brains behind the Sargon chess program that was the first major commercial chess program to be widely distributed. And of course on top of all these names that were the doers, we're fortunate to have with us Murray Campbell who is one of the great doers. Murray is one of the main members of the Deep Blue team which consisted of Murray and CB Hsu, Thomas Anantharaman and Joe Hoane, and then they had a whole sprinkling of other people that helped them. In addition to this motley bunch, which is a tremendous collection of talent, we have David Levy sitting at the table here. David was famous as a writer, as a critic, as a skeptic, as an organizer, and is one of my good friends for many years. My feeling, as I've been involved in this project since 1970, is it's been a great, exciting 35 years and this perhaps is the culmination of it.

I think I'm entitled to two quick short stories about the years that I've been involved in computer chess and one of them happened near here in Napa Valley. I think you guys all know what Napa Valley's famous for, and there's a guy named Gallo that has a vineyard in Napa Valley. And he was a very gracious guy and he was a bit interested in chess and so he decided- well, he had been having chess tournaments every summer for a while. He'd invite chess players from all around America to come to his vineyard and play in this tournament and of course he's out to promote his wine. And so as they played chess, he would spruce 'em up a little bit with a little wine, and the degree to which they took the tournament serious was always a question mark. So one year, in 1975, one of the guys that snuck into the vineyard was a computer. And of course nobody took computers very serious in 1975, but lo and behold, David Slate and Larry Atkin -- and Keith Gorlen who's hiding in the audience, Keith is somewhere way over here --, Keith was… David, Larry and Keith entered their program in this vineyard tournament in the class B section. Now the class B section is typically the level of… the best player in every high school probably has a class B player or two. And, lo and behold, it won the first game. And its opponent of course was little bit spruced up, and so these things happened. It won the second game. Oh, these things happen. The third game, the fourth game, and the fifth game -- it went undefeated. It won all five games and, lo and behold, it was the champion of this class B tournament. Well, the one thing that has happened over all these years is the pattern of denial in the human race. And lo and behold, nobody could really take it serious that this program won the tournament because everybody was a little bit tipped up. So Slate, Atkin and Gorlen went back to Chicago to Northwestern University and prepared their next strike at the human race, which was the Minnesota State Championship if I am correct. And lo and behold, I think their program may have won that championship. Is that correct, Keith? You don't remember. Okay. Well, the important thing was that the skeptics in Napa Valley just couldn't believe what happened, But it did happen. That was 1975, and from 1975 on computers started to play at the level of what might consider an expert chess player.

My second event that I wanted to talk about happened with the chess program Belle -- Ken Thompson's program, Belle. And it's a story about Zook the Book. Zook the Book was a New York chess player. In New York there's a lot of chess players that wander the streets and will play for a quarter against anybody willing to gamble a quarter with them; if they want to gamble more, all the better. And these guys are very sneaky guys, and Zook the Book was one of these guys. He'll play anybody for anything. Anyways, he came to the tournament the evening that we were having the second or third round of a world championship in New York, I believe at the New York Hilton, and he came up to Ken Thompson and he looked him in the eye. I'm not sure if he was looking Thompson in the eye or Belle in the eye, but he said, “If I win ten in a row, do I own its soul?” So Zook the Book-- of course Thompson wasn’t sure what to say -- Zook the Book… The reason he was called Zook the Book was that he had memorized every opening line that one can memorize from the opening and he was very good at opening books. Zook the Book sat down hoping to own Belle’s soul very shortly; this was about 12 at night. Ken was very tired, he wanted to go home, but Zook the Book wanted to play and Ken was always very gracious and would take on all comers to Belle at any time. After Zook the Book had been beaten seven or eight games in a row and getting more and more depressed, he finally got up and sheepishly walked away and Belle went to bed shortly thereafter. So those are my two stories, sort of highlights on my years in computer chess.

I’d like to introduce the panel now and I’m going to take a minute to go through each one. Each one has an extremely prestigious background. I want to introduce Ed Feigenbaum first. Ed is in the second place here. Ed was born in Weehawken, New Jersey and completed all his degrees at Carnegie Mellon, finishing his PhD in 1960, and he’s currently a professor emeritus at Stanford University. Those of us in computer science probably consider him the father of expert systems if that’s fair to say. Is that fair to say?

Edward Feigenbaum: Fair to say.

Monty Newborn: And when I was a graduate student -- I’m just a year or two older or maybe three, it’s not clear -- I read his book “Computers and Thought”, and that was very much a part of my education. He’s also authored a book called “Applications of AI in Organic Chemistry: The DENDRAL Program” and one of his great contributions to computer science was his DENDRAL program for analyzing molecules. He also co-authored a book called “The Fifth Generation: AI and Japan’s Challenge to the Computer World” and he did that with Pamela McCorduck, who we’ve seen her writing about Data General’s computers and prior to that. Ed is a member of the National Academy of Engineering since 1986. He’s a fellow of the AAAI. In 1995, he won the prestigious Turing Award from the ACM and he received an exceptional civilian service award from the United States Air Force in 1997. Ed. Will you give him a hand? <Applause>

Monty Newborn: At the end of the table, we have John McCarthy who was born in Boston in 1927. He grew up on the west coast in LA, got his undergraduate degree from Cal Tech in 1948, and wound up going to Princeton thereafter, getting his PhD in 1951. 1951! That’s a long time ago. He taught at Princeton, he taught at Dartmouth, he taught at MIT, and he’s spent most of his career here at Stanford. He’s currently a professor emeritus at Stanford. Ed [John] is a fellow of the Computer History Museum and has played a part in the advising and the development of this Museum, an important role. Along with Ed Feigenbaum he has been awarded the ACM Turing Award, back in 1971. He’s been awarded the Kyoto Prize in 1988. He was awarded the National Medal of Science, which is quite an achievement, in 1990. He’s a member of the National Academy of Arts and Science, the National Academy of Engineering, and the National Academy of Science. So it's a trick to be a member of all three of those organizations. There's probably very, very few people in the world that are in all three. Now Ed Feigenbaum might have been considered the father of expert systems, but expert systems is a sub problem of artificial intelligence, and I think it's fair to say that John can be considered the father of artificial intelligence, and I think that's an incredible feather in his hat. He developed Lisp in the 1950s, he proposed an XML-like language in the 1980s. He co-authored the Kotok-McCarthy chess program in the middle '60s and that was the program that played against the Russians. I hope we hear a little bit about that. He has written a number of books. The most interesting to me is "Formalizing Common Sense: Ascribing Mental Qualities to Machines". He's written a paper called "Making Robots Conscious of Their Mental States" and he wrote that in 1995 and he's quite interested in free will even for robots, which was the title of another of his papers. So we may have robots that are quite free. <Applause>

Monty Newborn: The third person I'd like to introduce is Murray Campbell. Murray is of course a member of the Deep Blue team. He was born in Canada, educated at the University of Alberta where he received his undergraduate degrees, and he received his PhD from CMU in 1989.

Murray Campbell: '87.

Monty Newborn: 1987, I'm sorry. 1987. So he's the youngest of the bunch. He teamed up with CB Hsu early on and Thomas Anantharaman and they developed the program called Deep Thought. And Murray joined IBM in 1989 with Hsu and Anantharaman and they continued to develop Deep Thought which turned into Deep Blue. He is currently at IBM's TJ Watson Research Center in Yorktown Heights and he's managing the Deep Computing in Commerce department. So we'd like to welcome Murray. <Applause>

Monty Newborn: And our fourth panelist is David Levy. He was born in London, England in 1945. David and I have been friends since 1971. We've been through dozens and dozens of chess tournaments together, rehashed dozens and dozens of games together, and enjoyed watching the programs get better and better every year. David was the Scottish chess champ at the age of 22. Levy played John McCarthy a friendly game of chess at an AI workshop in Edinburgh in 1968, and he made a bet of 500 pounds with John at that time regarding the future of computer chess. We'll hear more about that when David comes and talks. In 1977, David defeated KAISSA. He defeated chess 4.9, 4.7 later and then in 1978 won his bet from McCarthy for several thousand dollars. And John was quite gracious in coming up with the money. I'm not sure if it set him back very much, but he paid his share. David is the author of an unbelievable number of books. I think David just types and types and types. He's written about 40 to 50 books on the subjects of computers, chess, and a wide range of subjects and is extremely prolific. He's President of the International Computer Games Association, which is the organization in charge of organizing all the big tournaments these days, and he's been in charge of that organization for a number of years. And last but not least, from my perspective, he helped me organize the IBM-Kasparov Deep Blue Match and I always appreciate that very much. David. <Applause>

Monty Newborn: Okay, our next phase is each one of our panelists is going to talk for five or ten minutes and I'm going to ask David to be the first one. He's going to talk about his wager I believe. David?

David Levy: Thank you. Good evening. As Monty said, in August 1968 John and I started a bet that became a milestone in computer chess history. We were at a cocktail party in Edinburgh during one of the machine intelligence workshops organized by Donald Michie who was founder and head of the first AI university department in Britain. During the party, John invited me to play a game of chess, which I won. And when the game was over, John said to me, "Well, David, you might be able to beat me, but within 10 years there'll be a program that can beat you." And I was somewhat incredulous at this suggestion. I'd recently won the Scottish championship and it seemed to me that programs had a very, very long way to go before they got to master level. I knew of course of John's position in the world of AI for which I had the greatest respect, but I felt that he simply underestimated how difficult it is to play master level chess, and I was also a bit brash and I've always had a tendency to make somewhat large bets. So I offered to make a bet with John that he was wrong. And he asked me how much I wanted to bet and I suggested 500 pounds which at that time was a little more than a thousand dollars.

Now to put that into perspective, in those days I was in my first job after graduating university and the bet represented more than six months' salary for me. <Laughter> So John wasn't quite sure whether to take the bet, so he called over to our host, Donald Michie, for advice. And Donald was sitting on the floor a few feet away from us and he asked Donald what he thought. And Donald immediately said to John, "Could I take half the action?" <Laughter> And that of course gave John a lot of confidence and so we started the bet, we shook hands, and that's how it started, with each of them betting me 250 pounds that I would lose a match to a computer program within 10 years. Later the bet grew bigger. The following year, Seymour Papert and Edward Kostrowicki [ph?] joined the list of opponents and the final amount at stake when we ended the bet was 1,250 pounds.

But I had never felt that I was going to be in any trouble, and it turned out that I was right. In August 1978 at the end of the 10-year period, I played a match at the Canadian National Exhibition in Toronto against the reigning world computer chess champion program, and although they put a very pretty girl facing me to make the moves and smile at me when I was thinking, I still won the bet. In 1979, the following year, I made another bet for a further five years. This was with Dan McCracken and was for 1,000 dollars and again I won. But by the time I won in 1984, I could see the writing on the wall so I got together with Omni Magazine and I said to them that I would like to offer a prize for the first team or the first program to beat me, whenever that might be, and said that I would put up 1,000 dollars if they would add 4,000 of their money to it. And so we did. We announced a 5,000 dollar prize with absolutely no time limit. By 1989, the group at Carnegie Mellon University, of which Murray Campbell was a key member, had created Deep Thought, which was a veritable monster of a chess computer, and it started to score major successes in tournaments involving very strong grand masters. It even won a tournament in California ahead of a former world champion, Mikhail Tal, and I knew then that my number was up. And sure enough, when I was challenged to a match at the end of 1989, I was horribly crushed by a score of four to zero, but I was reasonably satisfied because I'd lasted for 21 years since the original bet.

Fortunately, Garry Kasparov was willing to take up the battle on behalf of mankind, and that gave the struggle to improve the best programs even more emphasis. The Man versus Machine contest and my bet in particular elicited some interesting and provocative comments from various experts. In the human chess world, I encountered two diametrically opposing views from two former world champions. Mikhail Botvinnik, who was a real titan of chess, who held the world championship for 12 years during the period from 1948 to 1963 with a couple of gaps, said to me, "I feel very sorry for your money, David." On the other hand, the Dutch mathematician Max Euwe who held the world championship title for two years from 1935 to 1937, as soon as he heard about my bet he wanted to take a share of my side, but I said no.

And in the world of computing, Monty Newborn made a prophecy during the 1977 World Computer Chess Championship, one year before I played before my first match, and this was a prophecy that was somewhat optimistic in its timeframe, but it came through more quickly than I believed possible. Monty said, "Grand masters used to come to computer chess tournaments to laugh. Now they come to watch. Soon they will come to learn." And learning they have been. Grand masters have been employing chess programs as analysts for several years and since 1987 huge databases of games from master play have been employed to help strong players prepare for games in tournaments against specific opponents. In addition, there are databases of end game positions that have taught the chess world the truth about some positions that have been incorrectly analyzed in the literature. In one case, the configuration of pieces in the end game that had been thought and stated in all the books to be drawn was discovered through computer analysis to be a win for one side. So, Monty, now you are right. Today grand masters turn on their chess programs when they want to learn. Thank you. <Applause>.

Monty Newborn: That's it, David?

David Levy: I could go on for three or four hours, but there are other people.

Monty Newborn: Good. Thank you very much. Ed Feigenbaum, you're next.

Edward Feigenbaum: I'd like to start by reiterating what a few people have said. I was a volunteer on this exhibit project and I just want to say the staff at the Computer History Museum -- Kirsten [Tashev] and Dag [Spicer] and their people -- are really fantastic so I really appreciate what they did and the exhibit that they mounted. I also want to say that looking at this audience, this audience is scary. I mean this is like, you know, among the best people who ever practiced computer science and computer technology are sitting here in this audience so it is a humbling experience to be talking to this audience. And the third thing I wanted to say in prelude was that in contrast to David and some of the others on this panel, I am not a chess maven at all. I've been following the history of chess playing machines casually as an AI scientist and I'd like to address my remarks tonight as an AI scientist, but don't ask me any questions about chess.

The first comment. I just want to make three comments in my short remark. The first is that I was around -- and Monty has promised to ask me questions about this -- at Carnegie Mellon University (which was then incidentally Carnegie Institute of Technology) when Newell and Simon and their colleague at Rand Corporation, Cliff Shaw, decided to pick chess as one of the domains of work in which they were going to explore new ideas in thinking machines. It wasn't the only one. They started out with proving theorems in propositional calculus, and they did other things as well, but chess was a major focus. And they did it because they thought that it was a route that would lead them to understand human problem-solving better, because it was thought to be an extremely challenging problem for human problem solvers. And they weren't the first who thought that of course. We know Turing wrote about this in the late '40s, early '50s, but he also talked to others in Britain about this earlier. And he himself thought that, and so did Shannon. Shannon wasn't mentioned in the rundown of people, but he was important in all of this, and other people at Los Alamos that worked on this effort. So these people were great visionaries and they took the problem of chess playing seriously. And I think that it's really worthy of our attention and it's worthy of the first of the great exhibitions at the Computer History Museum.

They were wrong on the timescale. In '57, Simon gave a talk in which he predicted that a machine would beat the world's chess champion in 10 years and he was wrong by 30- it was a 40-year history, not a 10-year history, but that's the way it is with AI. <laughter>. And we're really happy. The problem is that this feat was achieved by a technique that we all thought was basically impossible. So it serves as what you might… If you're one kind of scientist, you would think of it as an outlier point in the space of achievements of artificial intelligence. If you're another kind of scientist, you'd say it's really a challenge to the fundamental hypothesis that AI has been living with, growing up with, over the last 50 years. It's about to celebrate its 50th birthday. And that is… Let me try to sketch that by starting at the bottom.

Problem-solving is viewed by AI scientists as search in a maze. The mazes are enormous. Some people have -- not me, I haven't checked this myself -- but some people have estimated that the size of the chess maze is 10 to the 120th [power] and that's relatively small compared to what you might call real world problems. So the size of the problem space -- those are called problem spaces -- is enormous, and that any kind of what you might call brute force search, although it ended up in the famous chess playing programs as being not totally brute force, but that brute force search was fruitless. That no conceivable computer could search out the solutions to problems in the time remaining in the universe or even store intermediate results in all the matter that existed in the universe. So of course we invented another way and that was the use of knowledge. And that wasn't an invention that came later because I was there at Carnegie Tech when Herb Simon -- I mean Al Newell was doing one hell of a lot of programming with Cliff Shaw at the other end of the teletype line at the Rand Corporation -- but Herb Simon was reading lots of chess books and he was really becoming… you know, he never became a David Levy in terms of quality of chess play, but he really got to know chess very well. And Alex Bernstein who worked at IBM was himself an expert at chess. So the idea of applying knowledge to the chess problem space was not a new idea and it became the standard idea in artificial intelligence.

These experiments that Monty was referring to before with the DENDRAL program for hypothesizing organic molecules from mass spectral data, we actually ran what you might call titration experiments, in which we ran search versus knowledge. We just kept adding -- titrating in -- more and more knowledge of chemistry and watching the search be reduced and watching the quality of the problem solutions improve and we published all of that. So AI people began to talk about what's called the knowledge versus search spectrum. And almost all the points on the knowledge search-spectrum ended up on the knowledge side, not on the search side. So it was really remarkable when in the '90s the developments that you've heard of began to produce world class problem solvers in a problem that the early and great visionaries that I referred to earlier on in my remarks had thought were such a quintessential wonderful problem. You apply factors of let's say ten to the seventh more computation than you had before. Why that has any impact on a problem space that's ten to the hundred and twentieth big. What's ten to the seventh? That's nothing. You'd think that knowledge would win. And also people process information very slowly and they are extremely good chess players even though computer programs now beat them. Anyway, I think that we still have a lot to learn from chess playing as a domain of enquiry. We have now- when I said one data point, I meant chess playing, but actually that's a mistake. There are dozens, hundreds of these different chess playing programs that we need to analyze carefully from the point of view of knowledge versus search, and try to understand why the search strategy paid off.

Okay, final comment has to do with combinatorics and creativity. I've been over this territory numerous times with my former colleagues working on the DENDRAL project: Joshua Lederberg, Nobel Prize winner in medicine who collaborated on the DENDRAL project and Bruce Buchanan. The DENDRAL project was one of these vast combinatorial… excuse me… The DENDRAL problem solver was faced with one of these vast combinatorial problem spaces and it used knowledge in organic chemistry and mass spectrometry and NMR spectroscopy to narrow that problem space, to home in on correct answers. And so we've discussed often the question of what could lurk in those problem spaces. Could there lurk some unique or very surprising quote "creative" unquote solution to some problem? After all, the problem space is so vast that you can't see it. So as you search, if you have the right heuristics will you come upon some really creative solution to some problem, and is that really the answer to what is creativity?

In 1956 when Newell and Simon first started to prove theorems in Whitehead and Russell, there was one theorem in chapter two which was a longish theorem. It took about two pages for Whitehead and Russell to prove. And I think that the logic theorists proved it in something like two or three lines. And Simon wrote this up and sent a letter to Bertrand Russell -- it's all discussed in Simon's autobiography -- but you might say how was it possible for a program in 1956 on those primitive early machines of, you know, 4K memory or 8K memory to come up with anything quote "creative" unquote like that? It was just finding it in a space. So with that in mind, I was very much struck by the remarks that Kasparov made, both written and interview remarks, after the game that he lost where he talked -- I don't have the quotes in mind exactly, but there was one, Kirsten could quote them accurately -- one he talked about something like he got into this… he thought he was looking into the mind of God. And, you know, that's not literally true, but he was spooked, and he even said he was spooked. He said that there was a move in one of the games that really got him jittery and he… well, David, you can tell the story better than I can. But anyway, what's that all about? What that's all about is these huge combinatorial spaces and humans can see so little of it and there are numerous avenues in those spaces where you can find something truly novel, truly surprising, and that's what we call creative. <Applause>

Monty Newborn: Our third speaker is Murray Campbell and I'm excited to hear what he says. He told me just before we came up on stage that he's going to talk about learning, and I think learning in the context of Deep Blue, and I'm out to hear and learn something myself.

Murray Campbell: So as someone who's spent a lot of years in my life trying to get a program to play better chess than it did, the one thing that I regret most is that Deep Blue did not have an ability, an effective ability, to learn. And in fact the entire machine and program was designed and programmed and tuned by hand by myself and my colleagues, Feng-Hsiung Hsu and Joe Hoane. And I can't tell you how many times I sat down with grand master Joel Benjamin, who was working with us on the project, and he would try and explain something that he thought Deep Blue was doing wrong and try and explain what the right idea was and he would not be able to put it in operational terms, in terms that I could program. He would say, "Well, it's just: this attack is winning, you know, it's obvious." <Laughter> And this knowledge bottleneck, getting the information out of the human experts… I'm sure Ed has had to deal with that for years and years, but it was very frustrating. And if only Deep Blue had been able to learn by itself rather than having to be programmed every step of the way, life would have been much easier for me. Not to say that there hasn't been some great successes in machine learning with respect to game playing. For example, Gerry Tesauro at IBM developed a backgammon program that played world class backgammon after training itself by playing games against itself, training a neural network. So it's clear that it can be done in some domains, but in my opinion there haven't been any great successes in learning in chess, either because they've been very narrow attempts or they've been things that just don't scale very well.

Let me give one example. A common attempt to learn in chess is to do what's called "tuning the evaluation function". The evaluation function recognizes patterns in a chess position and assigns weights or values to them and then composes those values to create a numeric evaluation of the overall chess position, so that you can compare positions and make your choice which is the best position. And this is an important part of a chess program, but it's in some sense the last thing you do. You've had to build the entire structure. You have to decide what the patterns are that are relevant for a program to know in order to play good chess, and in what contexts are those patterns relevant and how they change depending on the context. And then of course we get to the value that they are assigned. And there's even other important parts of learning in a chess program; for example the search, learning where to spend… where to focus your efforts, maybe focus your efforts on a certain part of the possibilities and focus less on others. So it's clear that there are lots of other kinds of learning that have been given relatively little attention. I would suggest that an interesting challenge would be to create a program that played chess by learning basically from the same inputs that a human player learns from. A human player has a few typical ways of learning. They learn from a teacher who will give them carefully chosen examples that are meant to illustrate a point that hopefully can be generalized, they learn from books, they learn from observing other players as they play, and of course they learn by playing games themselves. So if one could build a system that, just using the same inputs as a human, could reach very high-level chess, I think that would be well beyond the current state of the art, that's for sure. And I think it would be nice because it takes advantage of the things that drew the early AI researchers to chess in the first place, which is that chess is a well-defined problem, it has a well-defined set of rules and goals, it has a rating scale so you can easily measure your progress, and it has a huge pool of human experts out there who are willing to give you input and provide opponents and help evaluate what you're doing. And of course chess is far too difficult to actually solve in any reasonable way. So to summarize, I would say that a machine that could learn chess in the same way that people learn chess would be an interesting challenge for the next five, ten, fifteen years. Thank you. <Applause>

Monty Newborn: Thank you, Murray. Well, we've waited for a while to hear the words from the father of artificial intelligence himself and I'm excited to hear what John has to say in this regard. John, would you like to say a few words?

John McCarthy: Yeah, sure… my usual habit. I have only moderate excuse to speak about chess programming, which is to say I started to write one, and then I got some students to take it over. And this was the program for the IBM 704 that played, using an IBM 7090, the match with the Soviets. And they had a better program; a worse computer, but a better program. And when they described it to me in 1965, I already knew that it was better because it had some ideas that I hoped to put in, but had not succeeded in getting the students to do it.

Okay, now I think I want to be a bit technical in this and I want to be a bit historical, but I'll start the history quite a bit earlier. Now, evaluation functions have been mentioned, but evaluations functions came along long before there were computers playing chess. In particular, almost any chess book will tell you simple statements like: a queen is worth nine and a rook is worth five, and some others will say well a queen is worth ten and a rook is worth five, but they all say a pawn is worth one, and so forth. So anybody that teaches you how to play chess will tell you something about the evaluation function. Now the pre-computer guy who did the most with evaluations was Aaron Nimzowitsch, who wrote a book called "My System" which was heavily based on an evaluation function that he devised that took not merely materiel, but also positional aspects into account. And Alex Bernstein, who was the first person to write a program for the full board, decided to adopt, insofar as he could, Nimzowitsch's evaluation function. So we really did have a mathematization of chess to a certain extent long before computers came along.

However, the pre-computer mathematization of chess as far as I know didn't discuss at all the process of search. That is, of course everybody agreed there was a process of search, but nobody had anything very specific to say about it. Now the first proposals for chess programs, and the very first chess programs, used minimax, in which they calculated all the moves, or in some cases just some of the moves that were promising, and all the replies or the promising replies, and so forth, and formed this tree of moves as far as was computationally feasible for the computer of that time. In Bernstein's case, it was seven by seven by seven by seven, so he examined about 2,500 end positions. The Los Alamos program also used minimax, and Turing's hand simulation used minimax. Now the first advance over that was alpha-beta, which said: well, you don't really have to evaluate all the positions, because if you move and the opponent can refute it, then you don't have to look for other refutations once you've found one. That's a bit of an oversimplified description of alpha-beta, but when it works well -- and it requires good move ordering to work well -- it reduces the number of positions that have to be examined to the square root of the number that have to be examined otherwise. So if you are going to look at a hundred million positions with minimax, then you could hope to look at only ten thousand positions using alpha-beta.

Now I think Ed is mistaken in that there haven't been heuristic advances in computer chess over these years, and Murray has talked about some of them. But nevertheless brute force, as it's called, using some heuristics, has beaten human chess, and human chess is more sophisticated. So why is that true? And I think it's true because chess is peculiar. Namely, the amount of branching in chess is rather moderate, so that you can follow out the branching. If there were a lot more branching, then the chess brute force wouldn't work. And in particular the programs for playing the Japanese game of Go, which have had a lot of work put into them, are far worse than chess programs partly because of the greater amount of branching. Now the question has to do with what… Now I think alpha-beta is compulsory. Now different people invented aspects of alpha-beta at different times. It uses my nomenclature; that is, I called it alpha-beta in my invention of it, but I didn't get it quite right. And the first comparison of it with minimax, where it was shown to be definitely better than minimax, was with Mike Levin and Tim Hart around 1960. And then quite independently of that, a Russian by the name of Alexander Bruneau [ph?] invented alpha-beta and wrote a really proper scientific paper on the subject. But my claim is that alpha-beta is compulsory for chess programs. All the programs since, I don't know when, 1960, or maybe the middle '50s, have used it. But yet it's not totally obvious, because really some very good people who thought about chess programs, Shannon for example, did not think of it, although a lot of people who did build chess programs independently did think of it. So it's not a unique idea, but it's a good one.

When I first visited the Soviet Union in 1965 and got challenged to play the Soviet program, the head of the group that was doing the programming, named Alexander Kronrod, offered the slogan that chess is the Drosophila of artificial intelligence. And to some extent Kronrod's slogan is correct, but also to a large extent it is not correct. It certainly isn't a Drosophila for all of artificial intelligence, but then the Drosophila isn't the Drosophila for all of genetics. But it does have the peculiarity that it doesn't branch too much. Now the question of looking at the move tree as a whole, it seems to me that this has been experimented with adequately by Ken Thompson, who showed that you could make a program which would construct tables of position versus the right move in the position, for chess positions that had up to five pieces. Now of course faster machines and with bigger memory would improve that a little bit, but you're not going to get anywhere near the 32 pieces of actual chess even if you had the whole galaxy to be your memory. So I think Thompson shows us that we need something else. You know, chess has the property that lookahead to a certain depth with alpha-beta and with a good evaluation function and some other things that enable you to follow the most promising lines further is something that will allow a computer that is very, very fast to beat the world champion.

Now I'd like to see a computer chess tournament running in the opposite direction, namely, that has extreme limitations on the amount of computation that the program could do. Namely, with current PCs maybe one millisecond of computation should be allowed. Now my contention is that eventually people would make machines that do one millisecond of computation and can beat almost all human players, perhaps even the champions -- I don't really know that -- but they're going to have to be clever. And that will serve as more of a "Drosophila for AI" than a contest on who can get hold of the fastest computing equipment. Thank you. <Applause>

Monty Newborn: Well, I have two questions for the panel and then we'll open it for the audience to ask questions. Actually, the first one is really directed to David Levy, although I think everybody can maybe have their say on it. We're here to celebrate the fact that Deep Blue beat Kasparov. If Deep Blue hadn't beaten Kasparov, we'd be sitting waiting for somebody else, or maybe Deep Blue to beat Kasparov in a couple of years. Deep Blue beat Kasparov, and it's not clear that he beat Kasparov if you talk to Kasparov. It's clear that Kasparov lost to Deep Blue, but it's not clear exactly what happened. We're now almost a decade later and we can take a look back and maybe try and digest whether it was a legitimate defeat. And I would like to ask David Levy the question: was it a legitimate victory by Deep Blue? And maybe somebody else on the panel has something to say -- maybe Murray.

David Levy: Well, I think that it was most definitely a genuine victory. The sporting result was that Deep Blue scored three and a half points and Kasparov scored two and a half.

What actually happened is that in the first game of the match, Kasparov won, in the second game he lost, but the way he lost was something that he couldn't comprehend, because he was expecting to play a program that was a bit improved on the program he'd played a year earlier in Philadelphia, and instead he played a program that was much improved. But that wasn't the only thing, because in the second game of the match, the program played the Ruy Lopez opening against Kasparov which is probably the best known opening in chess literature. And the Ruy Lopez is characterized -- particularly the variation they played -- it's characterized by very slow, subtle maneuvering with long-term strategy being the most important factor in deciding what moves to make. Things happen extremely slowly and some of the games are very long with this opening. And in positions like that, everyone knows, or everyone knew, computers just can't play those positions because they require long-term strategy. And yet Kasparov was sitting there and move after move after move after move, Deep Blue found a really strong strategic move, the basis of which was in a long-term strategy. And slowly but surely, Kasparov's position got worse and worse and worse.

And I hate to think what was going on in his mind because he was witnessing something that before the game started he didn't believe was possible. And as a result of that game, which he lost, his psyche was completely destroyed. There were some other extraneous factors that caused him to worry unfoundedly, but what really happened was that his psyche was destroyed by seeing a program play in a way that he never believed possible, and certainly it was a very, very far cry away from the Deep Blue he'd played the year before. And in a six game match, if your psyche is destroyed in game two, you're not going to play at your best after that. And although he tried very hard in games three, four and five, he was unable to make more than a draw in any of those games. And after five games, he was totally mentally drained. He was psychologically destroyed, and so game six, when he came to game six he wasn't really in a fit mental state to play because the program had already destroyed him. And that is perfectly legitimate. If you're playing a match at chess against a strong opponent, you want to win by scoring more points than them, by destroying their psyche, by outplaying them, by playing in ways that they're not expecting. All of these methods are perfectly legitimate. So yes, Monty, I think it was a serious and genuine victory.

Monty Newborn: Thank you, David. Would anybody else..? <Applause>

John McCarthy: Two remarks. Destroying his psyche reminds me of Bobby Fisher who wanted to do that. The second remark is that even if Kasparov had won that match, I believe the Computer Museum would still have managed to hold this meeting. <Laughter>

Monty Newborn: Maybe Murray would like to say something.

Murray Campbell: I just wanted to add that during game two, we were all smiles during that game because we were thinking back about a year earlier, sometime in the summer of 1996, when a certain set of positions had arisen on the chess board during our training and we decided to add a particular feature to the evaluation function which came into play and made a huge difference in that particular game. And so we realized, as that game progressed, that this feature -- which was related to when to open up the position and when to keep it closed but keep the option of opening it up -- we could see that this was putting a lot of pressure on Kasparov. And whether or not we won the game, we were very pleased to see all that hard work from a year earlier come to fruition in that particular game.

Monty Newborn: Thank you, Murray. So I think it's fair to conclude -- and I haven't heard anything too terribly contradictory -- that Deep Blue's victory was a legitimate victory. And that John says we would be here anyways, but it certainly increased the probability by having Deep Blue win the match. And I'd like to congratulate Murray one more time. I've congratulated him many times. <Applause>

Monty Newborn: I think it's appropriate to open up for questioning from the audience.

Don Knuth: Well, people were mentioning alpha-beta. I just wanted to mention I wrote this paper giving a mathematical analysis of alpha-beta, and a month later I got… on my desk at Stanford was a box of prunes. And I couldn't think who have I offended by this? And, you know, because we call it pruning -- alpha-beta is a pruning technique -- but then I looked at the box a little closer and here it had been purchased in the East Bay at an Alpha-Beta market. So I received this box of Alpha-Beta prunes. <Laughter> Okay.

The question I had for the panel is -- if you'd like to comment -- I think programs that play chess are a lot different from most computer programs in one respect. That is, if you take most computer programs and if you sort of at random clobber several instructions in the program, then the program dies. But you take a program that plays chess and you sort of randomly change instructions -- you know, a plus to a minus or something like this -- it just gets slightly weaker. And I'd like them to comment about that.

Murray Campbell: I certainly observed that. You can't imagine, the program's been playing for weeks or months, and we discover the most horrible bug, that we can't imagine how the program could have been playing, but it was. And it's just… It's searching through so many possibilities and it's so robust that it just somehow makes its way through those bugs, which makes it harder to find the bugs in the first place of course.

John McCarthy: I'd like to add something to that. One of the bugs that Arthur Samuel found after some trouble in his program, that it had the sign of the evaluation function backwards. <Laughter> And yet it seemed to play pretty well. <Laughter> And there's a reason for that. The reason for that is that if you want to… There is a form of chess in which you want… A variant of chess in which you want to lose, in which you want your king to be checkmated. And the correct way to play that is to play as if you wanted to win until the last minute, and then start throwing the opponent your pieces. So there's a concept which has never been really described by anyone, which is the question of a powerful position that enables you, in a powerful position, to achieve whatever you want. And Samuel's program looked ahead quite a long ways and given the backwards evaluation function it wanted to have as low a score as possible. And it turns out that the right way to have a low score after ten moves is something like to play as though you wanted to have a high score for seven moves and then start throwing things away.

Monty Newborn: Okay, the next question please?

Steve Squires: I was really struck by the realism expressed by the panel, and their extraordinary ability to celebrate 40 years towards solving this incredibly hard problem, and then coming up with yet new problems to solve. And I'm really serious about this, because I remember when Toole and I were at DARPA and a call came in from the director's office saying, "Did you know that there was a skunkworks unapproved project at CMU that was using lots of Moses [sp?] design cycles to produce some chips for this computer to play chess? What do you think?" And we said, "Yeah, we know about it. What's wrong with it?" And they said, "Well, it's not an approved program." We said, "Well, actually it is. It's a skunkworks. That's the way it's supposed to work." Okay. They didn't bother us. Now the New York Times asked for an official interview with me. I said I really wasn't aware of it, it really didn't matter, but if it was going on at CMU and they thought they were learning, it was good enough for me.

Now for a more serious topic. So one of the things that impressed me was your candor in saying the system was not learning. It took 40 years of extraordinary computer science, AI, luck, hacking, wizardry to get to this point, and then you set the new challenge of "we'd like it to learn in the ordinary way". And then John McCarthy, who got us all into this trouble said, "Yes, but without such a large computer. With a computer which is about only as fast as this rather error prone rotten thing that's between my ears," or anybody's ears. But I have a question for McCarthy. Would you allow some number of, say, idealized cubits to be in this computer and if so… an idealized cubit means a cubit which really works assuming that the error problem has been solved, just to reduce it. So what is your estimate for -- take the parameters you gave for the processing speed of this thing or your thing, which is way better than my thing -- and how many cubits do you think it would take to, first, redo it as well as Deep Blue (Deep Thought) did it. And second, how many cubits would it take, and how hard would it be, and what timeframe, to get to the point where it would learn? I thought I'd give, you know, 10, 20 years, bet you a thousand dollars, whatever it is. 1024 dollars…

John McCarthy: Okay. Well, I attended a course of lectures in quantum computing a few years ago and learned about as much as I'm capable of learning at my age. But anyway, quantum computing is still substantially a mystery. So it's known that it can do factoring. It's known that it can't do NP complete problems. I don't know how it's known that it can't. So I don't know how quantum computing could be made to do chess. I know, or at least I believe, that with the present knowledge of quantum computing, even if you could make a quantum computing, nobody would know how to make a chess program that was a lot better than non-quantum chess programs, but some clever person as clever as Peter Shore, might very well succeed in doing that.

Monty Newborn: Ed, would you like to make a comment?

Edward Feigenbaum: Yeah, I'd like to take off on a different angle from Steve's question to John, not on quantum computing, about which I know very little, but I do want to discuss this issue that John McCarthy raised about limiting the amount of computing. That is, posing as a challenge to AI scientists a game in which you would limit the amount of computing available. Now if you go back to the very beginning of the modern history of AI, in fact if you… One good way to look at this is to look at the sections of the book that I think Monty referred to before, the book "Computers and Thought", which was an anthology put together in 1962, and they're about evenly divided between two halves of AI: artificial intelligence, which is sort of the "I don't care how we do it as long as we do it well" part, versus simulation of human cognition, which is "I'm aiming for verisimilitude, testability versus human data, in information processing models of human thought." John, in his suggestion, I think was saying if we do that we will stimulate more thinking along the lines of simulating how clever inadequate computation needs to be in order to work at all, which is what we were trying to do in the early days in simulation of human cognition. I'd like to recall, though, that -- I think I mentioned it in my remarks -- one of the goals of AI, the engineering side of AI in 1956 and '60 and so on, was ultra-intelligent computing. That is, some of us were interested in psychology like Herb Simon, and Al Newell, and even my early work and -- Don Norman's in the audience, his work and others -- but others wanted just to build programs which were as smart as could possibly be. And what I'd like to understand now, -- from what you might call the victory of big computation over human computation in chess -- I'd like to understand: what are the bounds of that? What happened there? If chess is a good area in which that can be accomplished, then I'd like to know just why. How can I pick other problems that will be just like that, where I can beat the hell out of human performance with high speed computation? And I don't think we understand that now. I mean John said it… did you say it was sort of an accident, that chess was..?

John McCarthy: No, a peculiarity of chess.

Edward Feigenbaum: And we don't understand why. If we understood it scientifically, we could find other problems that fit.

John McCarthy: I want to tell one DARPA story. I have a student named Dave Wilkins who wrote a thesis on recognizing patterns for sharp positions in the middle game in chess. And I was talking to the DARPA program manager and justifying it and saying, "Well pattern recognition here would be useful for pattern recognition in other problems besides chess." And he said, "Well," -- he was a military officer -- he said, "Well, all right, but when he publishes his work of his thesis would he kindly not acknowledge our support."

Monty Newborn: That's an unusual one. We have a number of people that want to ask questions and I'm torn between listening to the responses to the questions and hearing what the questions are going to be. Can I possibly ask the people that are going to ask the questions to be very brief and if we can have… I don't know how we get brief answers from our panelists, but either that or we're going to have to cut it short. So let's try another couple of questions and see how it works out.

Michael Dearing: Michael Dearing, and my brief question got lengthened by the last two questions. I wanted to ask McCarthy: when you said let's have a millisecond of time, you said limited computation. I heard a time limit, but I didn't hear a space limit. I mean our brains process slowly, but we've got a lot of neurons there. So is there a limit on how many peta transistors I'm allowed to employ during that millisecond per move?

John McCarthy: Well, how active the human brain is is still something which is unknown. What is known, for example, only recently, is that when someone is doing mental arithmetic, a certain small part of the brain starts using much more energy. And what this says to me, or what this suggests to me, in accordance with my previous prejudices, is that most of the time most of the brain is inactive.

And that means that comparisons of computing with the brain based on counting the number of neurons in the brain are not sound. I'm sorry, did I answer the question? <Laughter>

Michael Dearing: Yeah, I've been looking at the FMRI data. No, the question was: if you're going to limit on computation -- put whatever limit on space you want, it doesn't have to be all the neurons of the brain -- but with everything now being parallel processors and custom chips and programmable database and so forth, there's many, many orders of magnitude difference between what one serial Pentium can do and what I can do on a couple of boards of programmable chips. So if you're serious about a millisecond challenge, I'd like there to be a space limitation as well.

Monty Newborn: Please.

Man in Green Sweater: Kasparov would argue, and I would argue, that it wasn't a fair contest between IBM and Kasparov because Kasparov could not see what the machine's tendencies were, how it played. There was no history, so Kasparov was at an immense psychological disadvantage before it even started. At the grand master level you study your opponents. You know what they do. You want to put them into a discomfort zone and you want to keep them from putting you into a discomfort zone, and you couldn't do that. I believe Kasparov wanted to get some trace of how the machine played and do it again, and IBM wouldn't do it. So I don't think it was a fair contest. I mean Kasparov lost, but IBM did have this immense psychological advantage. My question is: why are chess playing programs so good and contract bridge playing programs, by comparison, simply not comparable?

Murray Campbell: I just wanted to have one comment on the first point. Kasparov indeed had a large body of games and history to him that other grand masters know about and take advantage of. When we were building Deep Blue and programming it, we didn't pay attention to any of it. We programmed the machine as a chess player and not a Kasparov player, so in that sense Deep Blue knew nothing about Kasparov. It only knew about chess. So as for the other question, bridge, I think there's a number of games where there are… it's not in the sweet spot of where search and knowledge are just right and alpha-beta is the perfect solution. Bridge, with the randomness and hidden information, certainly makes it more difficult. Poker is another example of a game where computers are good, but not world class yet. And of course Go is the best known example of a game where computers are far from world class.

Monty Newborn: Okay, thank you very much.

Man in Green Sweater: Kasparov's point, though, was that he was at a disadvantage. I'm not saying that Deep Blue was not a Kasparov machine, but in Kasparov's normal preparation, what he expected, he should have known, or he felt he should have known, some of the tendencies.

Monty Newborn: Thank you, Murray. Next question?

Rand Woo: I'm Rand Woo. In my spare time I also do computer Chinese chess, so it's a little bit different. So I learn from the system of the computer chess a lot, but I would like to ask the panel members: in one sentence summarize the single-most important thing you learned through the computer chess history. Thank you.

Monty Newborn: Would somebody like to take a shot? What you've learned the most from..?

Rand Woo: The most single important thing you learned through the history.

Monty Newborn: Good question. What have we learned from computer chess? David, in one sentence.

David Levy: I think we've learned that problems normally requiring human intelligence for their solution can be solved by computers without using any intelligent techniques. <Laughter, applause>

Rand Woo: That's great.

Monty Newborn: John, can you top that?

John McCarthy: Yes, some problems. Now, to tell you the truth, not having studied carefully computer chess, I have a feeling that a lot has been learned that has more technical than I have been able to follow. So, for example, what has been published about Deep Blue is something that is quite technical, but I haven't managed to read it.

Monty Newborn: Ed, would you like to make a comment? What have we learned?

Edward Feigenbaum: Yeah, my sentence would be: It is remarkable, to the point of being almost unbelievable, that there is one outlier point on the knowledge/search spectrum called chess playing programs, when all the other data points that we have about excellence in AI program behavior lie at the other end of the spectrum.

John McCarthy: Well, checkers also.

Monty Newborn: Checkers is almost solved.

Rand Woo: It's not solved yet.

Monty Newborn: Murray, who's spent the most time recently batting his brains around on this problem, what have we learned?

Murray Campbell: I couldn't say it better than David. I think he's pointed out that there are some problems that it's better to throw computation at than try and be too clever.

Rand Woo: That's great. Thank you very much.

Monty Newborn: John [Toole], let me ask you, what do you think is reasonable on our time schedule here? Can we take another 15, 20 minutes of questions? Okay, we'll take another 15 or 20 minutes of questions. Would you please introduce yourself?

Ellen Oman: Yes, my name is Ellen Oman. This is a good follow on to what you just talked about, because both Murray Campbell and you, John McCarthy, seem to be calling for methods that were more analogous to the way human intelligence works. I mean it seems to me that AI has up until now worked at appearing intelligent, but inside having processes that are not analogous to the way humans think. And so this goes along with your question of the "Drosophila of AI". Perhaps chess is not a good Drosophila. I'd like to ask: Will a computer be able to complete the New York Times Saturday crossword puzzle more quickly than a human being? And I mean this seriously.

Monty Newborn: Would anybody like to answer that?

John McCarthy: I can answer that.

Monty Newborn: John?

John McCarthy: When they can do it at all, they'll be able to do it more quickly. <Laughter> Okay now..

Ellen Oman: I guess my point is that language is ___________.

John McCarthy: I must say that in my views I'm eccentric, and one of the areas in which I'm eccentric has to do with computers and language. And the work over the last 50 years on computers and language has concentrated on syntax. And my opinion is that the more important thing about language is the semantics: the meaning of things. To take a trivial example, suppose you want to translate into French and Russian the sentence "I went to Moscow". You can translate it trivially into French: "Je suis allez a Moscow", barring pronunciation. If you want to translate it in Russian, then what you have to know is whether the speaker is male or female, and also whether Moscow is a place that you would normally walk to, or a place that you would normally travel to in a vehicle. They would use different words for translating "went". So there are four cases. There's one English word, which translates into one French combination, would translate, according to this other information, into four different things in Russian. And it seems to me that, maybe the linguists haven't ignored this specific problem, but they've ignored a lot about the semantics of language. And I think to really do crossword puzzles, perhaps the semantics of language comes in.

Ellen Oman: I guess my point is that language is so central to human intelligence that it's hard to imagine a "Drosophila of AI" that does not include a deep understanding of semantics. I mean crossword puzzles are extremely difficult because they challenge you to think of so many ways that a single word can be used. And there are puns, there are jokes, there are missing things, there are patterns. It goes to every part of intelligence vis-à-vis language.

Murray Campbell: I should just mention I think there's already a grand challenge that's very language-oriented, and that's just the Turing test. As imperfect as it is, it's still something that has not yet been achieved that requires very good language skills.

Edward Feigenbaum: I've just got to say that I disagree with that whole line of reasoning. It's so often said, and there's no real substantial basis for saying it. For example, if we take Einstein: If Einstein was in some way language disabled and he couldn't really -- like he spoke some very weird Mongolian language that no one could understand except some graduate student who would translate -- would we still say that he was not intelligent? Or, suppose that he spoke only physics language and not regular human language? So it's a- I don't know, it's a very linguistics-centric view that says that language plays such a central role in all of this, or at least human language.

Murray Campbell: I think we need a whole swarm of flies, chess being just one of them.

Edward Feigenbaum: But the language thing I think is worthy of another panel here, John, so that somebody from the audience could make a bet with some computational linguists up here on stage. <Laughter> Like for example, the one that was just posed, about the New York Times crossword puzzle and so on, I can imagine someone with enthusiasm who doesn't really care much about his pocketbook would say "15 years we could do it" and then, you know, it would really happen in 50 or something like that. But that's worthy of another panel.

Monty Newborn: Boy, I'm torn to say something myself at this point. The animal world is an interesting dual to keep your eye on when we're studying intelligence. And while we've talked about human intelligence and machine intelligence, animal intelligence is quite an interesting aspect. And there's a book around "If a Lion Could Talk". I don't know if anybody's seen it, but what a lion would say is intelligent might be quite different from what a human being might say is intelligent. And I think we look at things as intelligent from a human standpoint, and it's not clear if that's the exclusive way to look at things.

This is a very exciting debate going on here. One of the things that, as a professor for many years, I've been to many lectures and the speakers will give a talk for an hour and at the end they'll ask for questions and the audience will sit there stone-faced and nobody will say anything and everybody walks home. But if you talk about chess, computers and intelligence, it heats up. So next question.

Cat Powell: Hello. My name is Cat Powell. I'm a volunteer greeter/docent here at the Computer History Museum. And I have a question for Ed. First of all… Two questions. The first one is are you by any chance related to Mitchell Feigenbaum at the Los Alamos- the famous chaos?

Edward Feigenbaum: The answer is no, and the great thing is I get credit for two careers.

Cat Powell: I just thought maybe you might be his brother or his relative.

Edward Feigenbaum: And then there's this Mitchell Feigenbaum envy. I mean, when you do a Google search of Feigenbaum and then you show up all these Mitchell Feigenbaum hits... And then there's Joan Feigenbaum, who was a student of ours in computer science, who has more hits than Mitchell or myself and she's no relative of mine either.

Cat Powell: Okay, my next question is related to artificial intelligence, so it's for you, the AI expert. Tic-tac-toe has a very limited problem space. It's probably one of the smallest problem spaces. And chess has a larger problem space, but it still is limited compared to maybe the crossword puzzles that the other lady was speaking about. And so my question for you is concerning AI algorithms.

Edward Feigenbaum: I'm sorry, concerning what?

Cat Powell: AI algorithms. The algorithms you use to solve problems for chess. Do you consider, or do you actually look at, cultural bias when you're using algorithm to solve chess?

Edward Feigenbaum: So I think that breaks up in… it's a very interesting question and it breaks up into two pieces. One is when knowledge-based systems program builders --which is the kind of program builder, well, I shouldn't say that I am, I should say, as John did, what we ask our students to do -- does that use knowledge which is culturally biased. And then there's the second question about chess, which is specific to chess, and that's something you're going to have to ask these two guys because I simply don't know the answer. I don't know the cultural answer and I don't know the chess answer to that one.

But on the other one, I would say that in the areas that we have explored most, I think the short answer is yes. But it will surprise you what culture. The culture we're talking about is scientific analytic thinking culture. Or maybe another way to put it is: when CP Snow was talking about the gap between the two cultures, we're talking about the techies, not the fuzzies, as we refer to them at Stanford. But we're not talking about the cultural difference between, let's say, the Germans and the Americans, or the Americans and the Japanese. When you get.. In what we, let's say, call western medicine, and you see that applied in Japan, it's just the same as western medicine applied in the United States. It's the same scientific analytic culture.

Monty Newborn: Thank you very much. I agree with you, Ed, as a small point. Can we have the next question please? Your name?

Eric Miller: Eric Miller. Hi. Basically the question is: there's been a lot of talk about making AI smarter, better and all that. What about the other way, making an AI that a 1400 player can beat, instead of something that can beat a Kasparov. A program that a modern computer, with no necessary time limits, no time limits, no search space limitations, that a regular 1400 player can beat because the program makes mistakes every so often, will hang a piece. Because I have never played a chess program that has ever hung a piece. If I see his piece out there, I'm afraid to take it because- I know, I'm paranoid.

Monty Newborn: David has developed a number of commercial programs over the years and I think maybe he has the best answer.

David Levy: I'll tell you a true story that happened to me in the late 1970s. I was consulting for a small company down south called Texas Instruments, trying to develop a chess program for a home computer that they were developing. And I wrote the algorithms, I wrote complete specification, and I went to Texas about 11 times over 18 months to advise the team on what to do. And one day I got in there and the guy who was in charge of the project, a very tall Texan, he came into the office with his Stetson hat and his leather boots and he was about six foot ten tall. And he put his leather boots on the table and said, "Golly gee, David, I don't like your program." And I said, "Why not, Len?" And he said, "It beats me every goddamn time." So I thought for a moment and I said, "That's a very fair comment, Len. I'll do something about it." So I put in a losing mode. At the first level, instead of him playing on level one, which was beating him every time, the program's level one was changed so that it would lose, but it would lose very subtly. It would start off by adding a very small random number to every evaluation. And as the game progressed, the size of the random number range increased, so that you actually got the impression that you were playing really well. <Laughter> Hang on, that's not the end. There were one or two rules like "thou shalt not give checkmate", and "thou shalt not make a draw". And so we put this level in, and the next time I went back a very strange thing had happened. Len had played against the losing level and naturally he'd won every game and that made him very happy. And he was so pleased after he won about a dozen games or so, he moved up to level two, which was the old level one that he lost every game to. And you know what? He started winning one or two games on level two, because he'd got so much confidence from level one. So yes, you're quite right. All chess programs should have levels in for all levels and strengths of chess player. <Applause>

Monty Newborn: Thank you very much. Your name please?

Barney Powell: Hi, I'm Barney Powell. Hi, guys. When I was doing my PhD thesis, kind of a few years after Murray's, and Murray was just getting going with the whole Deep Blue, and I think just after it beat Kasparov, I asked him a question which was, "If you were to change the rules of chess by just a little bit, so that knights moved over one and up three, instead of over one and up two, then what would you have to do to make Deep Blue play that game, or how well would it play?" And Murray looked at me said, "We'd just have to start all over." Okay.

Murray Campbell: Well, we'd have to build a brand new machine because it's so brittle that any change like that would completely destroy it.

Barney Powell: Right. And in addition, the evaluation function would be off, and you'd have to figure out where that would come from, but there wouldn't be any human games to look at, so it was an interesting challenge. That points to the larger issue which is that by focusing ruthlessly on just one game, we feel like we've made a lot of progress. But we've learned something and then we try to generalize out, but that kind of ruthless focus… it's possible that all the knowledge and all the power came from humans studying the game and finding just that sweet spot and just that combination. So that when you now go and look at another problem, we really don't know if it'll work or not. So there's an issue of generality.

What I worked on for my work was this concept of a general game playing program, and a different Drosophila, which was rather than play chess, make a program that could take in the rules of any game without any human assistance and play that game well. And I am happy to be able to share that just this year -- this is kind of a long time later -- there was the first general game playing program competition at the AAAI conference in Pittsburgh, and the programs were given rules that the humans had never seen before and they had to struggle and play. It was really difficult, they weren't very good, but a program that had more knowledge in it and sort of better suited to the general strategies of games did a better job. So there's a new level of competition coming up and I think that's really exciting.

Monty Newborn: Very interesting. Thank you. <Applause> We'll try and finish the seven questions.

Stan Miller: Hi. My name is Stan Miller. I work at Carnegie Mellon. I've been doing some robotic stuff there. So I sort of get a bit of a dissatisfaction in a sense that, you know, we conquered this problem, but in some way we didn't do it with the elegance or some je ne sais quoi, pardon my pronunciation. So my question really is for Murray and it's this: Obviously Deep Blue didn't acquire its knowledge in the same way that humans did, but once it had that knowledge base through all the years of programming, the man years, the millions (I assume) lines of code and all the chips and everything, do you think… Is there really that much of a difference between how it applied that knowledge, or is it just an architectural difference like it had a worse evaluation, but it did more searching? Don't humans sort of use a similar sort of database, in a sense, and is it really just a different way of accessing the same knowledge base? And then I'll ask another question later, but I don't have time.

Murray Campbell: The human approach is traditionally assumed to be a large database of patterns or chunks that are recognized and then the position is recognized relative to those patterns and the essential differences are pulled out and some logical reasoning is done and a small amount of search, a few dozen or a few hundred positions perhaps. So it's very hard to make a comparison with a few billion positions in an evaluation that's completed in, you know, a millisecond or a microsecond.

Stan Miller: So you're saying it's a qualitative difference, not a quantitative difference. You're quite certain of that.

Murray Campbell: Reasonably certain. I can't say I know enough about what's going on inside our heads, but it seems very likely to me that we're not looking at a billion possibilities when we calculate it.

Stan Miller: I'll leave that. I keep coming back to the comment that, you know, Kasparov thought he was looking into the mind of God and it just… I'm sensing a dissonance. You know, it's almost like he was fooled, like he would be fooled by a puppet or some sort of artifice or something like that. So it just confuses me.

Edward Feigenbaum: I think one thing you're not really getting is: it's our job to have a strong sense of dissatisfaction with where we are. Remember that our goal is something like this: John [Toole] will have a panel about a hundred years from now and you won't know whether we are automatons or humans. <Laughter>

Stan Miller: All right, this is really quick. I just want to say, and Eugene [Miya], you might remember the name of this game they play at the Hackers Conference, but you sit down at the table and you don't know any rules. And if anybody can remember the name, it was invented at MIT. It's called like Mum or something. You sit down at a table, you literally don't know the rules, and you tend to lose.

Monty Newborn: Thank you very much, very interesting. Next question?

Man in Plaid Shirt: My question I think would be a good follow-up on the previous question, and the question is: basically I think we all agree that AI has been the victory of big computation, or brute force computation, over human-like computation. Or basically, as somebody said, Deep Blue did not at a very fundamental level have the ability to learn. So my question is that maybe…. is this the wrong approach to achieving intelligence and wouldn't AI be a misnomer? Are there any other possible approaches to intelligence you guys have thought about? John, Murray?

Murray Campbell: Repeat the question.

Man in Plaid Shirt: My question is basically that: as we know that Deep Blue did not fundamentally have the ability to learn, so this approach to artificial intelligence that we have -- the search based approach -- isn't this probably a fundamentally wrong approach to achieving intelligence maybe? Shouldn't we be looking elsewhere?

Murray Campbell: I would just say that I think that whether or not Deep Blue learned is orthogonal to how it went about solving the problem, and I could easily imagine ways and research programs to enable Deep Blue, as it is now, to learn -- as opposed to having to invent a new system that would be learned.

Monty Newborn: David, would you like to add something?

David Levy: How can we say the approach is wrong when it works?

Man in Plaid Shirt: I guess my question is: okay, have you thought of other ways of achieving intelligence apart from the brute force computational way?

David Levy: Well, there was an attempt by one of Murray's colleagues at CMU, Hans Berliner, to devise a search strategy that worked in exactly the same way that human chess masters think. And when I first read about it, which I think was in 1979, I thought hurray, this is somebody who's actually come with a really intelligent way of searching the game tree. This is bound to work. And I was completely wrong, and so was Hans. It didn't work. And I think one must go back really to say that if something works, then why fix it?

Edward Feigenbaum: But David is right. That's the single most surprising thing about these decades worth of research on chess playing programs. What he just said. We all thought that. We were wrong. Murray was right. He's also right that learning is an orthogonal problem to problem-solving.

Monty Newborn: We've got to be a little quicker and we've got to limit the questions to 15 seconds. So you've got to say what you want to say quick, okay? Oh, we're over here, I'm sorry.

Man in plaid shirt: I lost my chance, almost. Thank you. We human beings are quite fallible. I want to say, even just chess grand masters how often, you know, they just lose their mind I guess sometimes. I asked two grand masters a question like, "Have you ever come across two-way discovered check?" And they never could answer that with a yes. Because I have myself come across that, and I don't know of any master game which has a two-way discovered check. I have not come across anything, but I had my own game once which had a two-way discovered check. It is a part. I just want to say that. This is a different topic. This is the actual question I wanted to ask: Kasparov was beaten by, let us say, a machine thinking of say 10 plys, 10 plys at most in every second or something like that. So I'm saying, if Kasparov is beaten, suppose somebody else comes up who is somewhat better than Kasparov. Now will an 11-ply machine be required to beat this man, and that will take many centuries to produce. Now can you comment on that please?

Monty Newborn: Sure. Murray?

Murray Campbell: I think Moore's Law shows that it won't take many centuries, in fact, to create. An additional ply of search is a factor of 3 or 4 in computation so that's- you wait a few years and..

John Salmost: <Inaudible>.

Murray Campbell: No, factor of three or four per ply.

Monty Newborn: That's it. Every additional ply is about a factor of three. All right, next question?

Man in Grey Shirt: Okay, I have been interested in computer chess, but from the opposite point of view, and that is the human point of view. And then I was going to ask David specifically a question. I've watched a lot of chess games and I see that masters sometimes slow down. And sometimes they move very quickly and they hit the clock, and then sometimes they just slow down and on occasion they just sit and sit and sit. And I've always wondered if I could put an MRI on their brain, or CAT scan, and see what's going on. Now you're a chess master. Are you searching longer and how do you get psychologically beaten, and why is your morale important and you can be destroyed psychologically, when you lose if this just is search and nothing else or computations?

David Levy: Well, that's two questions. The first question: are you searching longer when you're thinking a long time? When you're thinking a long time it's because you're not satisfied that you've yet found the right move in the position. You might sit and look at a position and feel, for example, that you're winning, but not see how to win. And you might have an intuitive feeling that there isn't actually a direct forced win in a position, but you haven't found it yet. And in that type of situation it's very tempting to sit and think for a very long time to try to find the forced win, because that will end the game. Conversely, if you're defending, you might have an intuitive feeling that there is one way -- and one way only -- of saving the game, and you haven't yet found it. So that again gives you a good reason to think for a long time.

As far as getting destroyed psychologically, I think it can happen to people in all sorts of problem-solving situations, and certainly in all sorts of games, where something happens that they weren't expecting, or where they feel that they've not given it their best. There can be all sorts of psychological reasons why someone's morale crumbles in a chess game. Playing chess is a very, very tense occupation. One analogy I give is that: imagine you're sitting in an exam for your university degree, and you're not allowed to cross out a single mistake. Think of the pressure that would be on you, if you knew that you would fail the exam if your examiner found a single mistake. That's what it's like playing chess. Every move must be right. You make one mistake against a player like Kasparov, and you're dead. That's why it can be so demoralizing if you feel you haven't given it your best.

Man in Grey Shirt: But after that demoralization happened, did his search ability or computational thing become slower?

David Levy: Sorry, I can't hear you.

Man in Grey Shirt: Did he become slower player because of this demoralization?

David Levy: Indirectly, yes.

Man in Grey Shirt: Or can he search through the same space?

David Levy: You can, but when you become demoralized and depressed, it's very difficult to think clearly, and so you find yourself being distracted by the depression and just unable to think properly.

Monty Newborn: Thank you, David.

Man in Grey Shirt: Now what did you mean by intuition?

Monty Newborn: We've got to get the next question first. Your name please?

Man in Blue Shirt: We were talking about millisecond modern computers and peta transistors and some of you mentioned concerning the space. I just want to talk about it. In 1979, some of these people might be interested in here that the Atari 2600 computer had a chess program which was done at a millisecond or one megacycle computer and it had 128 bytes. And I mean 128. I don't mean 128 kilobytes or anything. I mean 128 bytes and it stored the position. Now the people there, they divided the program up into somebody wrote the display and somebody wrote the program. Does anybody know who wrote the program?

Monty Newborn: Wait, we have to just ask the questions.

Man in Blue Shirt: Well, the question is what you were talking about here, that you could. This was a good program. It would beat people to play chess in high school in 128 bytes at one megacycle.

Monty Newborn: So what's the question? Okay, so there are some computer chess programs that run on a very small amount of memory. I'm not sure if that's the answer to your question, but that's the case. Next question please.

Ken Freidenbach: My name's Ken Freidenbach and I have one very quick question and one more substantial question, and they both relate to the game of Go. We're here celebrating four decades or 40 years of computer chess. By my calculations, we're also 35 years into computer Go, since in 1970 Albert Zobrist wrote a thesis, and in 1971 Jonathan Ryder under John McCarthy wrote a program that Don Knuth played against. So I'm going to give a paper next month at the 3rd International Conference on Baduk, or Go as it's known in Korea, and at the Myonjgi University in Korea, which has a department of Baduk studies, and they have one course on computer Go.

Monty Newborn: Please, the question?

Ken Freidenbach: So my question is: would any of these experts be willing to guess how long it'll be before we have a Go program that can beat an international grand master?

Monty Newborn: We'll go down the table quickly.

Ken Freidenbach: And then the other question would be has anyone looked at the mathematical Go endgames -- nightmares for professional Go players -- where they fight over the last point on the board and they've come up with an amazing array of numerical quantities that have very nonnumeric properties and what does this tell us about adding together things like territory and position?

Monty Newborn: Okay, I think the second question is too complex to go into at this point, so I'm going to excuse the panel on it. But if we run down the table quickly on a number, maybe we could get an opinion on the game of Go. Would anybody like to? John?

John McCarthy: Actually, the second question is the easiest to answer. <Laughter> Namely, Elwyn Berlekamp at Berkley wrote a whole book, essentially on Go endgames, and it turned out to be a pretty elaborate mathematical theory. Now, as to the first question, when will computers beat humans at Go, in my opinion it requires a new idea. And guessing how soon a new idea of suitable magnitude will arise is just wild. What one can't say, at least I will claim you can't say, is that progress in computer speed, or progress in human understanding of the game, is proceeding at some linear rate and all you have to do is divide the distance that you have yet to go by the rate and then you can calculate how long we'll get there. No, you can't do that yet.

David Levy: Can I disagree? I disagree with John. I've actually done this computation and it comes out at 35 years. <Laughter> The reason very simply is that Go programs have been progressing at an average of half a Q a year for the last three years.

Monty Newborn: We've got to move on. We've got 35 years from you.

David Levy: And if you want to see how I calculate it, buy my new book called "Robots Unlimited" which is being published later this year. <Laughter, applause>

Monty Newborn: Any comment, Ed? Murray? A comment? Thirty-five years, we've got plus or minus?

Murray Campbell: I think 15 to 20 years.

Monty Newborn: 15 to 20, so we've got an average of around 20 to 25, 25, something like that.

David Levy: Maybe my son will make a bet with you, Murray.

Monty Newborn: I think Ed is taking down notes to make a wager with David, or something, I'm not sure. Would you like to ask the next question?

John Gaddy: My name is John Gaddy and this question is for David. We were born in the same year and I realized how far you evolved and I'm still down in kindergarten with chess. When you told the story about the Texan, he was better than I am. So I have over the years tried to dabble in chess and I picked up a couple of these games, ____________ and Travel Champion 100 and had fun with them. How would you evaluate these games in the context of chess history, and what game would you recommend today for the consumer to sit and home and try to develop themselves as a chess player?

David Levy: Well, I never would recommend any particular program, because each program has its pros and cons. I would say you need to find a program that can play at roughly the level… that has at least some levels of playing skill roughly at your level because that's the most enjoyable for you. You need programs that have some levels above your level of skill, so that when you beat the current level you can move on upwards. And you just have to look at the features and the playing strengths and choose one that you think suits you. I'm sorry I can't be more specific.

Monty Newborn: Thank you very much, David. Our last question.

Tom Cochran: It it's the last one, I'll try to be quick. I'm Tom Cochran working as a contractor over at NASA Ames doing systems engineering. We're trying to design the next generation spacecraft, and in order to go long distances away from earth, we have to get away from the methodology of turning 20, 30 controllers solving all of our problems for us 'cause the communication length is too long. It's a rather challenging game managing the various subsystems that interact on spacecraft, and I was fascinated by the discussion about how chess programs can be adaptive even though they have major errors in them. I've been saying that the major barrier to human space flight right now is flight qualification of our software and we're looking for five-nines of reliability because we have so many different systems, we're afraid to fly unless we -- you know, or put our friends at risk -- unless we really have ways of verifying that it's true. So we've been stuck with much older methodologies for solving what are essentially very complex problems. So the question is basically: How do we do validation and verification on what's essentially an infinite state space of how the system is going to behave?

Monty Newborn: Maybe Murray would be good on that.

Murray Campbell: I can't say that Deep Blue was ever validated and verified. I'm sure many bugs existed, even during the games with Kasparov. It's just that chess is a complex enough game that many of the bugs never arose in the course of the six games that were played during the match, and in the course of our testing. So it's as you said, it's such a large state space, it's very hard to comprehensively cover that space, and I don't have any great insights from our work that would help in that respect.

Monty Newborn: Thank you, Murray. I'll just conclude with a couple of quick remarks. I've really enjoyed the discussion that we've had. It's always a very lively discussion and when you have such a talented bunch of people on a panel it makes it extremely interesting. This is an incredibly talented bunch. Looking through my own history of events in computer chess, this is possibly the most outstanding bunch of small number of people that I've been involved with. And I just want to thank the panel for participating and spending the time here and I’m sure all of you thank them and would give them a hand.

<Applause>

End