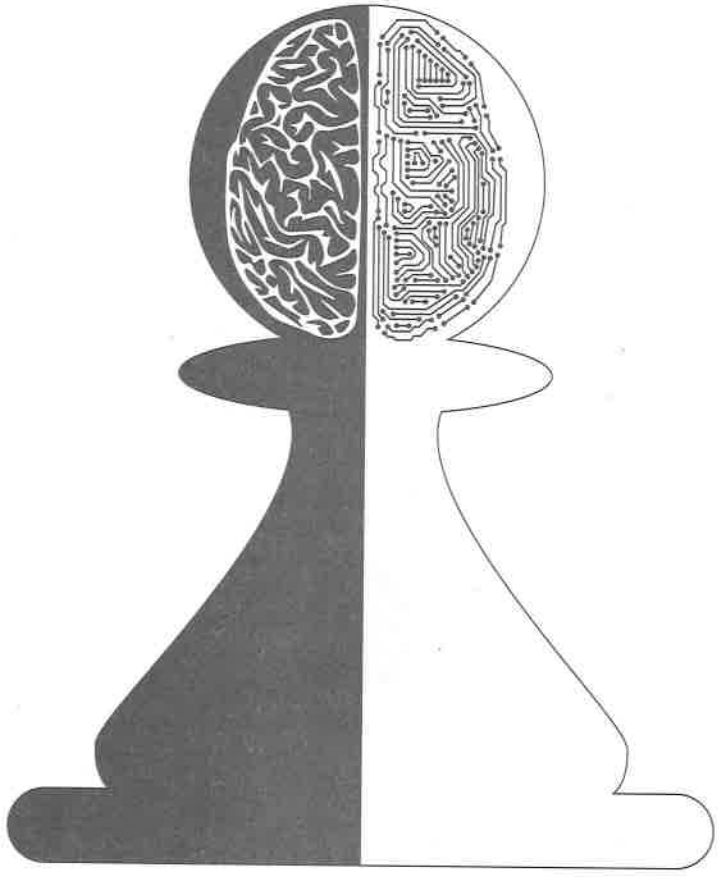
MAN vs MACHINE

Challenging Human Supremacy at Chess



Foreword by Vladimir Kramnik

Karsten Müller & Jonathan Schaeffer

Some quotes have been edited for consistency. This includes changing descriptive chess notation to algebraic notation, as well as standardizing spelling and punctuation.

All chess games given in this book were played using tournament controls (typically 40 moves in 2 hours), except where indicated otherwise.

The authors thank 14th World Chess Champion Vladimir Kramnik for agreeing to write the foreword for this book. We had an engaging hour-long conversation with him – a highlight for both Karsten and Jonathan!

Thank you to Monroe (Monty) Newborn for permission to use his photographs. All ChessBase photographs come courtesy of Frederic Friedel. The photos by Mde Vincente and Gerhard Hund are used under the Creative Commons License. Additional permission from IBM, Carnegie Mellon University, and Semen Karpenko are appreciated.

Foreword

Technology has forever changed the world of chess. Today's player has access to... strong sparring partners whenever you want to play (and they never get tired); encyclopaedic opening books (that really aren't books); comprehensive games collections (that include virtually every game played by every player of note); and perfection in some endgames (that defy human understanding). As a young chess player growing up in Russia and honing my skills, I had access to none of these tools. I had to find willing opponents, annotate printed opening books, transcribe important games, and attempt to discover the mysteries of the endgames. I'm not complaining, just describing the not-so-distant past. Computers have changed so much in chess, as indeed they have transformed so much of the world today.

I was late to adopt computers into my training regime. At the beginning of my career I used computers only for their game databases. In 1995, while helping Garry Kasparov with his world championship match with Anand, I saw how important the use of computer applications were for his training. After working with Garry, I started to use computers quite regularly. However, it was mostly used for blunder checking. We were analyzing on the board and sometimes you could easily just simply blunder something, miss some cheap trick, and just make mistakes. At that time, the playing strength of the programs was quite weak, but still strong enough to embarrass us. Even Kasparov sometimes blundered in his analysis, and the computer was ruthless and impassionate about pointing out the mistakes. This was a humbling experience for a grandmaster! Simply having the ability to check for blunders was in itself a useful tool.

Then came Kasparov's famous matches against Deep Blue. Frankly, I was not taking computers too seriously at that time. Even though I understood that it was not that simple to beat the computer, I was sure that Garry was going to win both matches. Of course I analyzed the games. I found it unsurprising that he won the 1996 match by a score of 4-2 – the games were normal and logical. The 1997 match was dominated by a lot of PR that distracted Garry (but not Deep Blue) and this may have played a role in the final result. But I'm absolutely convinced that Garry was still a much stronger player than Deep Blue. The final result was bad luck on Garry's part, that he lost his nerves at some point. Even so, he was the better player.

My own fights with the machines also started around that time. I don't really know when I lost my first game against a machine. I was not a big fan of playing training games with computers, but I probably lost one such game. My first tournament with a computer participant was probably a rapid chess event in Mainz around 1999. My first classical chess game against a computer was in 2000 in Dortmund. This was a grandmaster round-robin tournament and the computer's participation was controversial. Some players were against including the machine; I didn't mind so much. I won the game in my first classical man-machine encounter.

Then I prepared for my first match against Deep Fritz, which was planned for 2001. The first step in my preparation was to analyze all the 1997 Kasparov-Deep Blue

games, with Fritz running on my laptop. Just a laptop – no special chess chips analyzing 100s of millions of positions per second. To my extreme surprise, Fritz was simply playing better than Deep Blue. I was shocked. I couldn't understand how Garry managed to lose this match. When moves involved deep calculation, Fritz made the same moves as Deep Blue nine times out of ten. When a move choice was based on a positional decision, Fritz usually made a slightly better move than Deep Blue. I was puzzled -- I was expecting Deep Blue to be much stronger. I was even a bit frightened that I was going to play against Fritz but with it using more computing power. After doing this analysis, I was really surprised that Gary managed to lose the match to Deep Blue.

The first match with Deep Fritz actually took place in 2002. It started very well for me with two draws and two wins. I was extremely happy. But then in game five I blundered a piece in basically one move. In game six I was leading by one point and so I sacrificed material. So in the end the match was drawn and the question of whether man or machine was the better player remained open. In my second match with Deep Fritz in 2006 things did not go as well and I ended up losing.

It was already difficult, but still not totally impossible, for a human to beat a top computer program. But within maybe two or three years it became completely impossible. I think by around 2010 there was no chance anymore for the human side. In the history of computer chess, there were three chapters. First the humans were better for a long time. Then the interesting chapter, where man and machine were close in strength, lasted maybe 10 to 15 years. And now, the final chapter, computers are stronger for good.

Computers have changed the game of chess, the world of chess, and even my profession. There are many pluses to what computers have brought to chess. Of most value is that they improved our understanding and appreciation of the game. The minuses are obviously that there is much less opportunity for the human side – less room to be creative. We must not forget that chess is, after all, a game between two humans. Computers may now be stronger than the human World Champion, but this achievement does not change the real value of the game: the pleasure that we humans get from playing one another at this beautiful intellectual game. And that will never change.

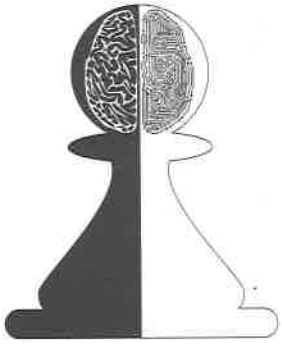
When I chose chess as my profession, I never imagined that one of the legacies of my career would be as a contributor to the field of artificial intelligence research. Both Garry and I put our titles and reputations on the line in the interests of Science. We both had early victories and eventually suffered painful defeats. I have no regrets. I enjoyed the challenge of playing against technology.

Grandmaster Karsten Müller and Professor Jonathan Schaeffer have managed to describe the fascinating history of the unequal fight of man against machine in an entertaining and instructive way. It evoked pleasant and not so pleasant memories of my own fights against the monsters. I hope that their work gives you as much pleasure as it has given me.

Vladimir Kramnik

14th World Chess Champion

Table of Contents



Pre-Game

The courtier stared at the chessboard, searching for a way out of his dilemma. Internally his mind was in turmoil, and externally it was starting to show. Faint hints of moisture could be seen on his forehead, above his knitted brow. He moved restlessly in his chair causing his frilly clothes to make rustling sounds. None of this, of course, bothered his adversary. Indeed, none of this was even noticed by him.

Across the chessboard was a man dressed in Middle Eastern clothes and wearing a turban. The opponent stared down at the board. He was emotionless in his expression, apparently unflappable, completely oblivious to the royalty and senior government officials watching his every move.

The courtier reached out and played a move. In less than a minute, the opponent replied, rather stiffly moving a piece.

"Oh no! I am losing!" thought the courtier in horror. "How is this possible? How can a mere machine, a combination of pulleys and wheels, best me, the creation of God? What wizardry makes this possible?" And then there was a surge of panic. There was no way out of his predicament. "I will be forced to resign in front of the Empress. How can I possibly recover from this public humiliation?"

The year was 1770 and people believed that technology had conquered the human game of chess.

Today's generation of chess players was raised on computers. To many of them, it is hard to imagine what life was like before cell phones, the Internet, Facebook, and Google. Technology now affects virtually every aspect of our daily life. Our love for the game of chess is not immune to the computer revolution. What serious player does not use a computer as a sparring partner or an analysis aid? Even if you are not a grandmaster, just to be able to play online whenever you want, against an opponent from literally anywhere in the world, allows you to indulge in your passion more so than ever before. Technology is helping us collectively to hone our skills through more practice and insightful analysis. Computers have helped discover secrets of the game and increased our appreciation of its beauty.

Although we take for granted that computers play strong chess, that was not always the case. In fact, grandmaster-level computer play did not happen until the late 1980s, and commercially available grandmaster programs arrived around the year 2000. In the fast-paced technology world of today, this seems like an eternity ago!

Today computers play at a level that is often called "super human"; they are stronger than the best human players (sorry, Magnus). The story of how computer chess programs grew from a whimsical idea into a 3300 ELO mega-grandmaster is intimately tied with the story of incredible technological advances that have happened over a 250-year period. The historical record, amazingly, started in the year 1770 when a brilliant engineer built a machine (named the Turk) that fooled people into thinking it played chess. Of course it was an illusion, but the seed of an idea was planted.

This book tells the story of the epic battle of man versus machine for supremacy at chess. From an entertainment-oriented beginning (1770), the foundational ideas

(1830-1952), the first programs capable of playing a complete game of chess (1960s), and the stunning victory of Deep Blue over World Champion Garry Kasparov (1997), the story of the scientific quest to build something many said was impossible is an important part of the history of computing, even the history of mankind.

This book tells the story of a cerebral battle between two remarkable groups of people.

The first are the scientists, who are passionate about developing new technology and pushing it to its limit. For the fledgling field of artificial intelligence research, building a chess-playing machine capable of competing with the strongest human players was one of the original "grand challenge" problems to tackle. Chess, it was felt, was a microcosm of more complex applications; if you could not satisfactorily address chess – a mere game – then what hope was there for harder problems? Fifty years of research, new ideas, improved computing hardware, and competitions against humans were needed to finally achieve this goal. It is an outstanding achievement and is justifiably regarded as a technological milestone.

The second group is the chess players, who are passionate about playing an intellectually stimulating game and improving their abilities. They are keen to demonstrate superiority over any opponent, whether man or machine. To excel at chess requires a rich set of highly refined skills, including imagination, calculation, reasoning, and learning. These skills set human chess players apart from mere calculating machines and, thus, human superiority at chess was unquestioned. Reality was something different and today it is acknowledged that the best chess players in the world are silicon based. Yet, the human achievement is impressive. Despite the onslaught of technology – computers analyzing millions of positions per second – humans remained on top for many years. The challenge coming from computers forced humans to up their game. Today's top players have achieved an unprecedented level of skill.

The motivations of these two sides are not all that different. The scientists did what they did in pursuit of knowledge (discover new ideas), ego (be first; be world champion), and, in some cases, financial gains. The chess players participated because of their competitive spirit (winning matters), ego (winning; be the best), and, in some cases, financial gains.

Man playing machine at chess was a scientific experiment, one that would have not been possible without human contestants. It would have been easy for the strongest chess players to say "no," and deprive the scientists of crucial performance data. Instead they bravely agreed to play. The stakes to them were enormous; a loss could hurt their reputation and pride in front of a global audience. In contrast, the scientists had little at stake; a lost game meant going back to the drawing board, searching for new ideas, or fixing a programming error.

These man-machine encounters have happened before and will happen again. It was only in March 2016 that the world watched in fascination as Lee Sedol, many-time World Champion at the Oriental game of Go, did battle with the computer AlphaGo. In 2011 it was man versus machine at the question-answer game of *Jeopardy!*.

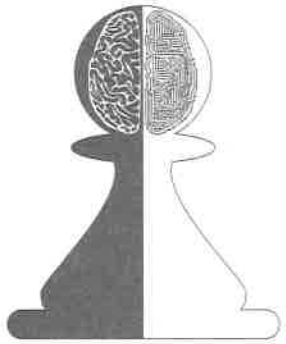
The man-machine battle for supremacy in chess is a landmark in the history of

technology. There are numerous books that document the technical side of this epic story. The scientists, by definition, have a requirement to publish their work. Hence there are literally hundreds of articles documenting the ideas, advances, experiments, and competitive results achieved by chess-playing computers. The human side is not often told. Few chess players are inclined to write about their man-machine encounters, other than annotating the games played.

This book strives to bring the two sides together. It tells the stories of many of the key scientists and chess players that participated in a 50-year research project to advance man's understanding of computing technology.

To enrich the story, we include three types of sidebars throughout the book: technology (explanations of the ideas that advanced the technology for building chess programs), context (quotes from chess players and scientists giving their opinions and predictions), and computer game milestones (showing how the technological advances were affecting other games). This allows the reader to understand the broader context of the computer chess story.

Join us as we recount the epic battle of man versus machine for supremacy at the game of chess.



Opening

1
0000 (1770-1956)

Cast your mind back in time. Can you imagine what the world was like in the year 1900? Automobiles, airplanes, washing machines, and microwave ovens either did not exist or were novelties. What kind of quality of life was that? No televisions, computers, Internet, and cell phones. What could we possibly do for entertainment? The 20th century saw an unprecedented level of innovation, likely only to be superseded by what happens in the 21st century.

With the advent of technology, it was inevitable that the competitive spirit of humankind would come to the fore. After all, if these machines were capable of performing tasks that we could do, then the obvious question to ask was "Who can do it better?" The idea of besting an individual's physical capabilities using a machine goes back a long time. In the 1800s, there is the well-known story of John Henry who matched his skills at pounding steel spikes into rocks with that of a steam-powered drill. Henry won that initial contest, but his days as champ were numbered. Later on, there were examples of human runners racing against horseless carriages. Over short distances, man could win – but only for a little while.

The man-versus-machine battles at strength and endurance activities are in the distant past. But man still battles machine when it comes to other physical skills, such as dexterity. For example, humans can still play football (soccer) and table tennis better than machines.

What about in the intellectual domain? We regard the capabilities of our brains as what differentiates us from other life on our planet. Our ability to reason sets us apart. Yet with the advent of computers, we have to ask the thought-provoking and humbling question as to whether we are so special.

One of the profoundest contributions of the 20th century was the realization that human's intellectual abilities could be realized using a computer. Imagine being in the year 1900 and told that a little black box (with the whimsical name of "computer") could correct spelling mistakes in your writing, do your accounting, and help you with navigation. Hearsay! Yet all of this is commonplace today. These *problems are solved* using data as input to a well-defined formula, and a machine capable of executing that formula using the data.

What about playing a game of chess? Surely that is different. There is no well-defined formula for playing strong chess. Further, there are other intangible elements at play, including imagining, reasoning, learning, searching, applying knowledge, and using psychology. That makes the problem much harder, right?

The research field of artificial intelligence has been working to develop computer programs that exhibit behavior that one would normally call "intelligent." It does not mean that the programs are smart, just that their behavior appears to be smart. All computer programs use a well-defined specification – the software that they

are given – and chess programs are no exception. However, unlike spell checking, accounting, and navigation, the mathematical formulas used for playing chess are not well defined.

Building a chess-playing machine that is stronger than any human chess player was one of the initial "grand challenge" problems for artificial intelligence researchers. For over 60 years, many researchers, commercial developers, and hobbyists around the world have worked on furthering progress on solving this fascinating problem. Progress was slow largely because achieving this goal depended on the invention of numerous other technologies. The DEEP BLUE victory over Garry Kasparov in 1997 would not have happened without amazing technological progress on many fronts over the past century. Some examples include:

- Computers. In 1900, the state of the art was an electro-mechanical machine that could count data on a punched card. A physical realization of a computer as we know it today did not emerge until the late 1940s.
- Engineering of the components needed to build microscopic devices. In the year 1900, who could imagine the precision required to build a computer chip at the nanometer level (one billionth of a meter)?
- Programming languages and tools for developing software. Imagine the wonder of sitting in a classroom in the year 1900 and being told that you could write a textual description of the solution to a problem (the "program") – and a machine would do exactly what you asked it to do!
- Algorithms that can be used for a chess program. What is an easy and/or efficient and/or practical way to use a computer to sort? Or sift through billions of possibilities? Optimize or maximize a computational result? Numerous algorithms (precise step-by-step methods) had to be invented.
- Devices for storing data. Data storage in the year 1900 consisted of paper in filing cabinets. One could manually manage hundreds or perhaps thousands of documents, but beyond that was a challenge. In 1956, the cost of a one-megabyte (one million bytes) disk was roughly $10,000 (US); by 1990 the cost of a one-gigabyte (billion bytes) disk was roughly $5,000; in 2016, a one-terabyte (thousand billion bytes) costs around $25!
- Electronic communication, the ability to send data anywhere in the world in an instant. This was superior to sending physical mail by car, train, and/or ship. Transmitting information that used to take several days (or weeks) now takes a fraction of a second.
- Chess knowledge. A century of studying the game has resulted in a deeper understanding of the theory and practice of strong play. The chess community learned much from studying Lasker, Nimzowitsch, Réti, and so on. Endgame technique has evolved considerably, starting with Rubinstein and Capablanca. Opening analysis has increased the breadth of playable openings and the depth of the analysis; some early names that come to mind include Grünfeld, Pirc, Alekhine, and Najdorf.

All of the above happened in the past century, most of it in the last half of the century.

Building a hardware/software entity capable of playing chess at a level higher than the very best human players is a remarkable feat. It is a testament to the incredible technology that has been built, and the passionate researchers and software developers who turned ideas into reality. It is also a testament to the brave grandmasters that were willing to do battle with the machine, allowing scientists to measure progress towards their goal. In turn, however, the capabilities of the computer were used by the chess players, allowing them to improve their game and advance towards unprecedented levels of human skill at a rich and complex game. This book tells the story of man versus machine for supremacy at chess, an "intellectual game *par excellence*" (Newell, Simon, and Shaw 1958).

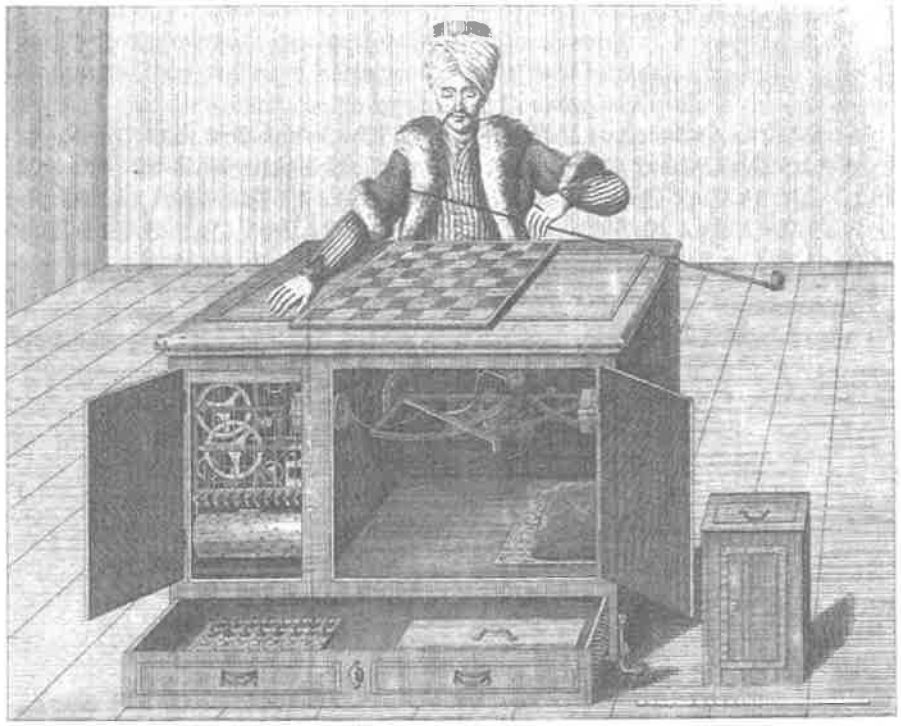
Incredibly, the idea of a chess-playing machine goes back almost 250 years. Baron Wolfgang von Kempelen (1734-1804) was a respected scientist who held favor in the court of the Empress of Austria. In 1770, he made a bold promise to the Empress that he could build an automaton that was more impressive than anything she had ever seen. True to his word, six months later, he delivered a machine that appeared to play chess.

Von Kempelen was a genius. The contraption he conceived of and built was impressive! It consisted of an ornate wood desk with many compartments, some containing a mechanical contraption of numerous interacting wheels of various sizes. The top of the desk had a built-in chessboard, and the pieces could be found in one of the desk's compartments. Seated behind the desk was an automaton that looked like a man dressed in middle-eastern robes. When von Kempelen's creation was running, the wheels would turn giving rise to the impression that something was being calculated. Eventually the machine stopped, and the human-like figure reached out and played a move. Periodically, von Kempelen would approach the machine and wind some dials, planting the idea that the machine used springs for its energy. The machine had no official name, but because of the way the automaton was dressed – in a turban – it was often referred to as the Turk.

The machine was a stunning demonstration of human ingenuity, and everyone who witnessed it play was duly impressed (von Windisch 1783).

> The most daring idea that a mechanism could ever conceive would be without doubt that of a machine which would imitate by more than mere form and movement the masterpiece of all creation. Not only has Mr. von Kempelen conceived such a project, he has executed it, and his chess-player is without any contradiction the most amazing automaton which has ever existed.

The Turk toured Europe to great success and eventually travelled to the United States. During its illustrious career, the machine did battle with many well-known personalities, including Benjamin Franklin. It even went to the most famous mecca for chess in the 1800s, the *Café de la Régence*, to challenge with strongest players in Paris, eventually getting a chance to play the great Philidor. After the game, Philidor apparently claimed that it was "his most fatiguing game of chess ever!" Sadly, no record of the encounter has survived.



The Turk chess "machine."
(Von Windisch 1783)

Perhaps the Turk's most famous opponent was Napoleon Bonaparte (Wairy 1895):

> The automaton was seated in front of a table on which a chessboard was arranged for a game. His Majesty took a chair, and sitting down opposite the automaton, said, laughing: "Come on, comrade; here's to us two." The automaton saluted and made a sign with the hand to the Emperor, as if to bid him begin. The game opened, the Emperor made two or three moves, and intentionally a false one. The automaton bowed, took up the piece and put it back in its place. His Majesty cheated a second time; the automaton saluted again, but confiscated the piece. "That is right," said His Majesty, and cheated the third time. Then the automaton shook its head, and passing its hand over the chessboard, it upset the whole game.

This account of the game's start is likely apocryphal. It is hard to imagine a man of Napoleon's stature (and ego) being toyed with by the machine. It seems more plausible that Napoleon was the one sweeping the pieces off the board. He played three times, losing badly; this was not the kind of treatment he was used to receiving! For the record, here is one of their man versus "machine" encounters.

Napoleon Bonaparte – Turk
Irregular Open Game C23
Schönbrunn Palace, Vienna, 1809

1.e4 e5 2.♕f3 ♘c6 3.♗c4 ♘f6 4.♘e2 ♗c5 5.a3 d6 6.0-0 ♗g4 7.♕d3 ♘h5 8.h3 ♗e2 9.♕e2 ♘f4 10.♕e1 ♘d4 11.♗b3 ♘h3 12.♔h2 ♕h4 13.g3 ♘f3 14.♔g2 ♘e1 15.♖e1 ♕g4 16.d3 ♗f2 17.♖h1 ♕g3 18.♔f1 ♗d4 19.♔e2 ♕g2 20.♔d1 ♕h1 21.♔d2 ♕g2 22.♔e1 ♘g1 23.♘c3 ♗c3 24.b×c3 ♕e2# 0-1

Today, it is easy to see that the machine was an illusion. Von Kempelen cleverly designed his contraption to create the impression that the Turk calculated and then played chess moves. Of course, hidden inside the machine was a strong chess player (the chess master Allgaier, in Napoleon's case). The cleverness of the design was how a human could be inside and avoid detection even when, as part of von Kempelen's showmanship, many of the compartment doors were opened to reveal empty space or mechanical parts. Von Kempelen never revealed his secret, and much effort was devoted in the popular and scientific literature trying to figure out how the Turk worked. Some correctly figured out the mystery; most did not.

The Turk had an amazing career. It died in 1854, the victim of a fire. However, the idea of a chess automaton continued, in the likes of Ajeeb (Hastings 1895 winner Harry Nelson Pillsbury did time in the machine) and Mephisto (World Championship challenger Isidor Gunsberg was the computer for a while – a chess master has to do what needs to be done to pay the bills).

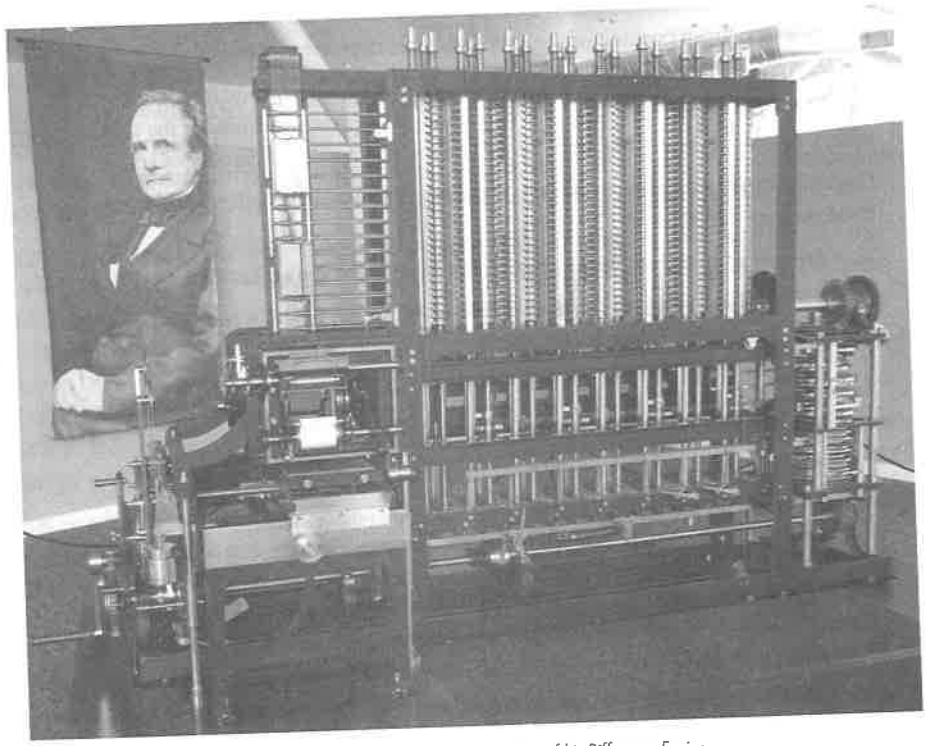
The Turk was the first example of using computers to cheat at chess. Here we had man impersonating a computer, a theme that continued to 1929 (when Ajeeb, oddly enough, also died due to a fire). Later on, of course, as computers became stronger at chess, we have a role reversal: humans misrepresented their playing abilities by using computers to decide on their moves.

While the Turk was a marvelous technological illusion, it actually inspired a technological revolution. While it was giving exhibitions in London in 1819, a young scientist named Charles Babbage came to see it and was intrigued. He returned the following year and played against the machine, losing of course. While Babbage understood that the Turk could not possibly be calculating moves on its own, it led him to imagine what it would take to have a machine do that for real. Thus was the inspiration for the dawn of the computer revolution.

Charles Babbage is rightly considered the father of computing. In the early to mid 1800s he envisioned building a mechanical device to perform calculations. Although the Turk had wheels and cogs rotating in such a way as to create the illusion of computation, Babbage's Difference Engine was the real thing – it used the same technology to actually calculate results. It could compute simple mathematical formulas and was used to accurately and quickly fill in numerical tables.

Babbage's experience with the Turk inspired him, and the idea of building a machine to play chess was at the back of his mind for much of his career. Later on he designed the Analytical Engine that, he believed, satisfied many of the

computational abilities needed to play chess. It included things that we take for granted today: memory, input and output (punched cards), a computing engine, and an instruction set. Although Babbage's research concentrated on the architecture of machines that could compute, his muse, Ada, Countess of Lovelace, thought about algorithms – how to design a set of instructions that a computer could use to carry out a task. While Babbage is the father of computing, Ada Lovelace is the mother of computer programming.



Charles Babbage with a modern recreation of his Difference Engine. (ChessBase)

In his autobiography, Babbage gives his reasoning about why he selected chess as a test-bed for his research and, impressively, the first algorithm for a computer to play a game (Babbage 1864). His thoughts on this matter were probably influenced by his work with Ada Lovelace.

> After much consideration I selected for my test the contrivance of a machine that should be able to play a game of purely intellectual skill successfully; such as [tic-tac-toe], drafts [draughts/checkers], chess, &c.
>
> I endeavoured to ascertain the opinions of persons in every class of life and of all ages whether they thought it required human reason to play games of skill. The almost constant answer was in the affirmative. Some supported this view of the case by observing, that if it were otherwise then an automaton could play such games. A few of those who had acquaintance with mathematical science allowed the possibility of

machinery being capable of such work; but they most stoutly denied the possibility of contriving such machinery on account of the myriads of combinations which even the simplest games included.

On the first part of my inquiry I soon arrived at a demonstration that every game of skill is susceptible of being played by an automaton.

Further consideration showed that if *any position* of the men upon the board were assumed (whether that position were possible or impossible), then if the automaton could make the first move rightly, he must be able to win the game, always supposing that under the given position of the men that conclusion were possible.

Whatever move the automaton made, another move would be made by his adversary. Now this altered state of the board is *one* amongst the *many positions* of the men in which, by the previous paragraph, the automaton was supposed capable of acting.

Hence the question is reduced to that of making the best move under any possible combinations of positions of the men.

Now the several questions the automaton has to consider are of this nature:

1. Is the position of the men, as placed before him on the board, a possible position? That is, one which is consistent with the rules of the game?
2. If so, has Automaton himself already lost the game?
3. If not, then has Automaton won the game?
4. If not, can he win it at the next move? If so, make that move.
5. If not, could his adversary, if he had the move, win the game?
6. If so, Automaton must prevent him if possible.
7. If his adversary cannot win the game at his next move, Automaton must examine whether he can make such a move that, if he were allowed to have two moves in succession, he could at the second move have *two* different ways of winning the game;

and each of these cases failing, Automaton must look forward to three or more successive moves.

Now I have already stated that in the Analytical Engine I had devised mechanical means equivalent to memory, also that I had provided other means equivalent to foresight, and that the Engine itself could act on this foresight.

In consequence of this the whole question of making an automaton play any game depended upon the possibility of the machine being able to represent all the myriads of combinations relating to it. Allowing one hundred moves on each side for the longest game at chess, I found that the combinations involved in the Analytical Engine enormously surpassed any required, even by the game of chess.

As soon as I had arrived at this conclusion I commenced an examination of a game called [tic-tac-toe] usually played by little children. It is the simplest game with which I am acquainted.

Although the algorithm omits important details, at its core is the principle of look-ahead searching.

Babbage was never able to realize his dream of building the Analytic Engine, as he was continually plagued by a lack of funding. Although his design was purely mechanical (rotating wheels with cogs), the breadth of his vision and the depth of his ideas would have profound impact on the computing revolution a century later. Today's modern computer has a high-level architecture not that dissimilar from Babbage's 19th century design. Just as importantly, he envisioned a computing engine that did more than calculate numerical formulas. Without fully understanding the implications, Babbage's ideas were the forerunner of artificial intelligence research.

Fast-forward fifty years. The first real chess-playing machine was built by Leonardo Torres y Quevedo (1852-1936). Inspired by Babbage's work, in the early 1900s he had the idea of constructing a mechanical chess machine (*a la* Babbage) that was run using electronic circuits (a forerunner of modern computing). A machine-driven arm would be used to move a piece. This was impressive technology for its time.

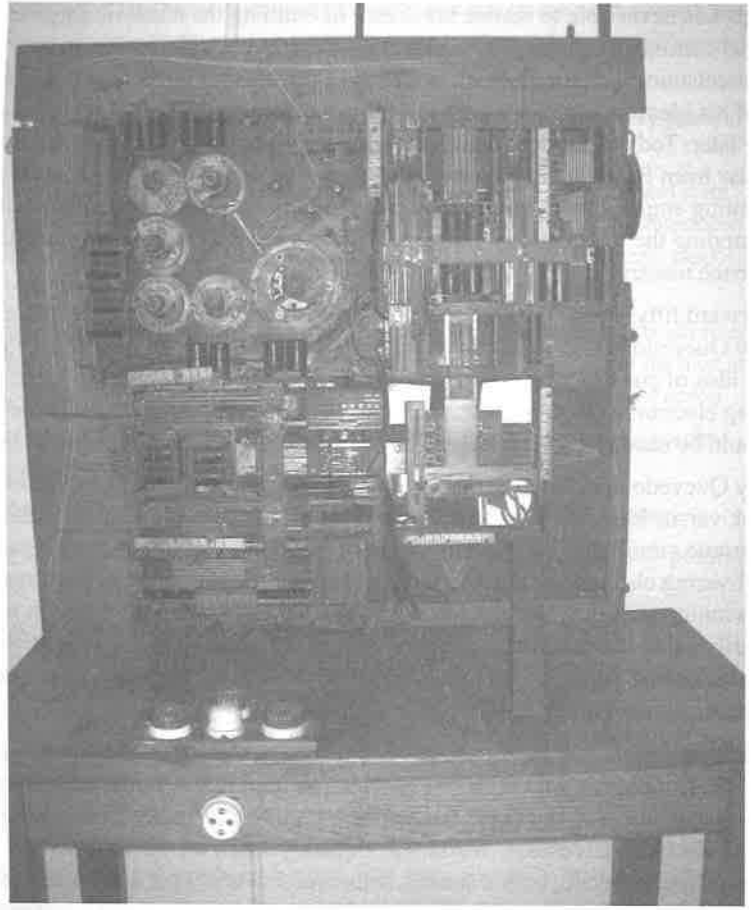
Torres y Quevedo decided to demonstrate his invention using the endgame of king and rook versus king, with the human always playing the weaker side. He designed an electronic circuit that adopted the simple algorithm of boxing in the king and then delivering checkmate. If a win were possible, the program would always play a winning sequence. Its algorithm guaranteed the final result, although not necessarily using the minimal number of moves. As opposed to the TURK with its human fraudsters, Torres y Quevedo's creation was a real automaton, computing its moves without any human input.

The machine was named EL AJEDRECISTA, Spanish for chess player. Although it was built in 1912, it did not gain widespread attention until it was exhibited in Paris in 1914. As happened with the TURK, the machine captured the public's imagination and generated scientific discussion. World War I quickly ended the conversation. In 1920, Leonardo's son, Gonzalo, built a second, improved EL AJEDRECISTA. Of note was that each chess piece had a metal ball at its base, and movement of a magnet underneath the board was used to make a move by rolling a piece to its destination square.

Torres y Quevedo's invention was the first real chess-playing machine. In fact, many regard it as the first computer game in history. Both EL AJEDRECISTAS can be seen today at the *Colegio de Ingenieros de Caminos, Canales y Puertos* in Madrid.

Norbert Wiener (1948)
Renowned MIT mathematician

A chess machine could be built that "might very well be as good a player as the vast majority of the human race."



Leonardo Torres y Quevedo's first chess-playing machine.
(Mde Vincente)

Fred Reinfeld (1948)
Chess author

Fred Reinfeld, introducing the game between Savielly Tartakower and Lajos Steiner (Warsaw, 1935), wrote that: "When Professor Wiener of the Massachusetts Institute of Technology invented a calculating machine which requires only one ten-thousandth of a second for the most complicated computations, he was quoted as saying, 'I defy you to describe a capacity of the human brain which I cannot duplicate with electronic devices.'

"Up to the time these lines were written, the Professor had apparently not yet perfected an electronic device capable of making such chess moves as Tartakower's 20th... The day may yet come, however, when we shall see such books as Robot's 1000 Best Games, or when chess tournaments will have to be postponed because of a steel shortage."

The world saw amazing technological advances in the first half of the 20th century because of and in spite of the global political upheavals. The most profound advance was the invention of the computer. Researchers had known for a long time (even before Babbage) that it was possible to build mechanical devices to automate a series of mathematical calculations. However, the 1930s and in particular the 1940s saw a number of important ideas come together, giving birth to the modern computer. Key to this revolution was the notion of programmability – that the machine would input a series of instructions to execute. Change the instructions, and the machine would exhibit different behaviour. Contrast that with El Ajedrecista, where it could do one and only one task. The notion of a computer was that it would be general purpose and do anything it was instructed to do.

Teams in the United States, Europe, and Great Britain worked independently to build the first computer. Depending on your definition of a computer, one can argue that each was first to the finish line! More importantly, people began to think about what you could make these machines do. The early applications developed for computers were, not surprisingly, military in nature. However, a number of dreamers had grander aspirations for this technology. For example, Konrad Zuse (1910-1995) in Germany, whose Z3 machine in 1941 has a strong claim to being the first programmable computer, developed a programming language and imagined how one might use it to program chess. Alan Turing, who worked on building a special-purpose machine that could crack Germany's secret wartime message encryption technology, had discussions with his scientific colleagues – all chess players – imagining how one might build artificially intelligence machines, including playing chess. But the first to get his ideas out to a wider audience was an American scientist, Claude Shannon.

Claude Shannon (1916-2001) was a researcher at Bell Telephone Laboratories in the 1940s. He is considered the father of information theory, having made profound contributions in a number of important areas of direct applicability to computing. He understood what computers could do and this led him to consider chess as an interesting application. In 1949 he gave a talk on the issues of getting a machine to play chess; this was later elaborated into his pioneering paper "Programming a Computer for Playing Chess." In the paper, he justified the research community's interest in chess with the following discussion (Shannon 1950):

> The chess machine is an ideal one to start with, since: (1) the problem is sharply defined both in allowed operations (the moves) and in the ultimate goal (checkmate); (2) it is neither so simple as to be trivial nor too difficult for satisfactory solution; (3) chess is generally considered to require "thinking" for skillful play; a solution of this problem will force us either to admit the possibility of a mechanized thinking or to further restrict our concept of "thinking"; (4) the discrete structure of chess fits well into the digital nature of modern computers.

If you add one more point – (5) that there are human opponents of varying skill levels that allow one to be able to assess the strength of the chess machine – then you have the introductory paragraph to most of the technical computer chess papers written in the next fifty years!

Shannon's paper touched on all the important aspects of a modern chess program. The contribution in the paper that is most often quoted is the two search strategies that he introduced:

Type A: consider all scenarios a fixed number of moves ahead. Shannon was concerned that a program built on this idea "would be both slow and a weak player." The mathematics supported his claim. If in a typical position a player had 40 moves to choose from, then looking ahead one full move (each side plays a move) would lead to 40x40 = 1,600 possibilities. This is certainly a manageable number. But what if you consider two full moves ahead? 40x40x40x40 = 2,560,000 scenarios. More work, but still doable. Three full moves ahead? 4,096,000,000. This is becoming a problem! This approach would soon be referred to as "brute-force search," usually with a derogatory connotation.

Type B: be smart about the move sequences considered. Shannon wanted to ensure that "the machine does not waste its time in totally pointless variations," thus only examine those lines of play that are "important." Of course, the whole idea hinges on defining what "important" means. How do you decide which move is relevant and which is irrelevant? Were that easy to decide, then chess would not be as intellectually challenging as it is. This approach is often called "selective search" – being selective (smart) about which moves are chosen to invest search effort in.

By so clearly differentiating the two search strategies, Shannon unwittingly set off a debate about the right approach for playing chess. Whereas Type B was a better reflection of how humans played chess, it had the serious disadvantage of trying to define "importance." On the other hand, a Type A approach was simple to program; it just got lost in the mathematical explosion of possibilities. As we shall see the question of the appropriate choice of search strategy dominated the discussion of computer chess for the next three decades.

Shannon concludes his paper with remarks that, with hindsight, are obvious yet profound:

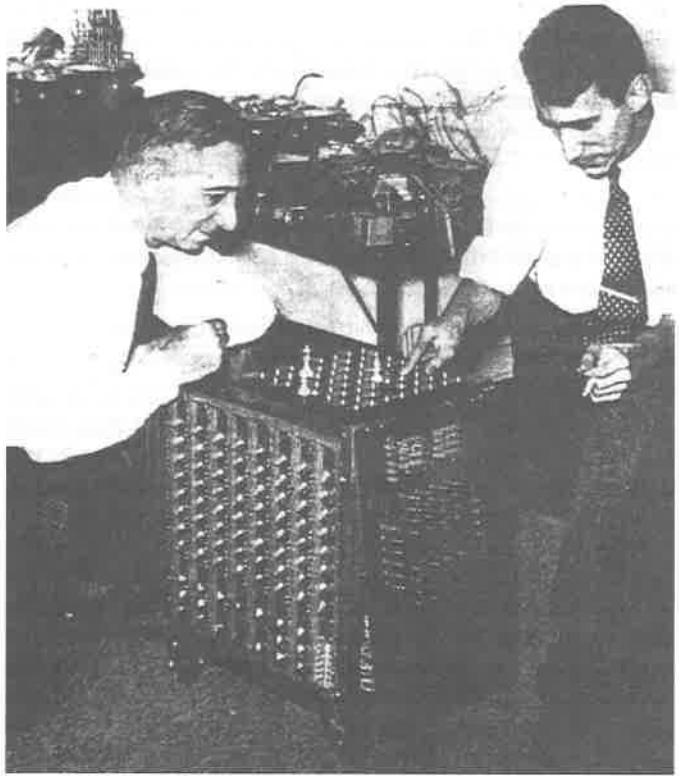
> It is not being suggested that we should design the strategy in our own image. Rather it should be matched to the capacities and weakness of the computer. The computer is strong in speed and accuracy and weak in analytical abilities and recognition. Hence, it should make more use of brutal calculation than humans, but with possible variations increasing by a factor of [a thousand] every move, a little selection goes a long way forward improving blind trial and error.

Consider the human brain and the computer as being two information-processing architectures for intelligence. Each has strengths and weaknesses. Humans are very good at visual analysis, reasoning by analogy, learning, and so on. Computers are very good at doing repetitious tasks, precise calculations, memorizing vast amounts of information, and so on. It is easy to think that the best way to build a strong chess program is to copy the human approach – after all, it works! But the simplicity of the argument does not match the reality of an implementation. What is interesting is that many things that humans are good at are hard for computers to do, and *vice versa*. When trying to solve a problem, such as chess, it is best

to exploit the strengths of the information-processing architecture and avoid the weaknesses. Thus if one wants to write a chess program, take advantage of the computer's ability to do repetitious tasks (look at millions of positions), do precise calculations (position evaluation), and memorization (openings and endgames). In effect, that is what Shannon is saying. Just because humans play chess does not mean it is a model we should copy for computers. Just because birds fly by flapping their wings does not mean we should build airplanes in a similar way.

Computing science is a fast-paced field. Many papers are obsolete within a few years of being published. Shannon's paper is one of these exceptional intellectual feats that stand the test of time. Reading the paper today, almost seven decades after it appeared in print, one cannot help but be impressed at how farsighted he was.

Shannon never wrote a chess program himself. He was a tinkerer and liked to build things. He did build a chess machine that allowed up to six pieces on the board. It is unclear how well (or even if) the machine worked, as it never gave a public demonstration. He was eclectic and inquisitive; he also built a machine to juggle balls.



Claude Shannon (right) demonstrating his chess machine to chess master Edward Lasker. (Monroe Newborn)

Look-ahead Search

When searching ahead, minimax is the fundamental principle used for determining the value of a chess move sequence. Pretend that the two players in a game are given the mathematical names Max (short for Maximum) and Min (abbreviation of Minimum). Max wants to achieve the best that is possible – winning a pawn (+1 in material) is better than losing a knight (-3 in material). Min also wants to achieve the best, but what is good for Min is bad for Max. Thus Min prefers a score of -3 (Max loses a knight; Min wins a knight) to 1 (Max wins a pawn; Min loses a pawn). In minimax search, given a choice of moves, Max always chooses the move to maximize the reward; Min chooses the move that minimizes Max's reward (or, conversely, maximizes Min's reward).

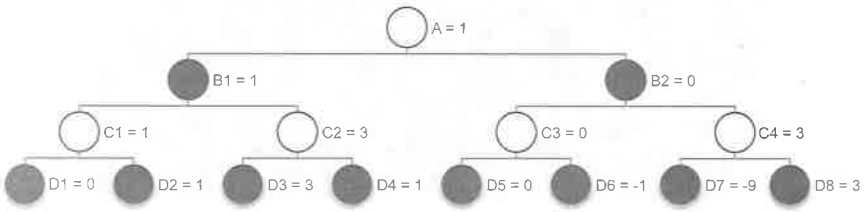
Consider a position, A, with White to move (the Max player). Assume, for simplicity, that both players only have two legal moves in each position. The example uses a simple material evaluation function, where a pawn is worth 1 point and a piece is worth 3.

In the diagram below, the White player has a choice of two moves, one leading to position B1 and the other to B2. Consider B1. Black has a choice of two moves, leading to C1 and C2. Now it is White to play. White could play to D1. Our evaluation of the position says material is equal, so the score is 0. Alternatively, White could try playing to D2. The evaluation is +1, meaning a pawn has been won. Now, given a choice of even material or winning a pawn, which would you choose? The value of position C1 is 1, the maximum of the values from D1 and D2. Similarly, the value of C2 is 3, the maximum of D3 (3) and D4 (1).

What is the value of position B1? It is Black to move, and Black wants White to do as poorly as possible. Given the choice of playing to C1 (1, White is up a pawn) or C2 (3, White is up a knight), Black will choose C1, the minimum of C1 and C2. A similar analysis can be done for position B2.

In our example below, the minimax value for position A is 1 – against Black's best defense White is guaranteed a score of 1 (a pawn ahead), the maximum of B1 and B2.

The above diagram is usually called a search "tree." If you turn the diagram upside down and use your imagination, then it looks like a tree. The starting position (A in this case) is called the root of the tree. The positions at the bottom of the tree, where the evaluation is done, are called leaves of the tree. And, of course, the lines that connect one position to the next represent moves and are called branches.



The above diagram is called a depth-3 search tree. All sequences of 3 moves (White moves; Black replies, White responds to Black's reply) were considered. However,

the word "move" is ambiguous in chess, so computer programmers usually refer to a move by a player as a "ply." Thus the opening sequence 1.e4 e5 2.♘f3 ♘c6 is a game of two moves, but four ply.

After the war, Alan Turing ended up at Manchester University and worked in the fledgling fields of computer hardware and software. But chess-playing machines were never far from his thoughts. As far back as 1948 he wrote that, "One can produce 'paper machines' for playing chess. Playing against such a machine gives a definite feeling that one is pitting one's wits against something alive." This reflected the discussions he had with his wartime colleagues at Bletchley Park (where the German code-breaking work was done), including chess masters Conel Hugh O'Donel (CHOD) Alexander, Harry Golombek, and Stuart Milner-Barry.

Turing's presence at Manchester spurred interest in computer chess. In 1951, Dietrich Prinz wrote a program for solving mate-in-two problems. His program adopted a brute-force approach (Shannon Type A) – every move sequence of three ply (White's move, Black's reply, and White's move) was considered. The program looked for any White move for which every possible Black move White had a checkmating move. In other words, White (Max) could achieve checkmate because Black (Min) could not find anything better than being checkmated. This was the first use of computers to validate the correctness of human chess compositions.

1952: Checkers

Christopher Strachey (Manchester Computing Machine Laboratory) writes the first working checkers program. It plays complete games against human opponents.

Evaluation Function

In minimax search, the positions at the bottom of the tree (leaf nodes) need to be assigned a value. Some values are obvious: checkmate is a very high score, being checkmated is very low (negative), and a draw is probably assessed as a 0. But what about all the other positions? An evaluation function takes a chess position and describes how good or bad the position is by relating it to a single number, usually an integer. Some programs use the convention that a positive number means White has the advantage; negative is in Black's favor. Other programs use positive scores to indicate that the side to move has the advantage. The bigger the number (positive or negative) for one of the players, the larger the advantage or disadvantage. A score of zero usually means that the chances are even (or, in the extreme case, that the position is a draw).

In Shannon's paper, he described a reasonable starting point for a computer being able to assess the desirability of a position:

> The evaluation function…should take into account the "long term" advantages and disadvantages of a position, i.e. effects which may be expected to persist over a number of moves longer than individual variations are calculated. Thus

the evaluation is mainly concerned with positional or strategic considerations rather than combinatorial or tactical ones. Of course there is no sharp line of division; many features of a position are on the borderline. It appears, however, that the following might properly be included in [an evaluation function]:

(1) Material advantage (difference in total material).

(2) Pawn formation:

(a) Backward, isolated and doubled pawns.

(b) Relative control of centre (pawns at e4, d4, c4).

(c) Weakness of pawns near king (e.g. advanced g pawn).

(d) Pawns on opposite colour squares from bishop.

(e) Passed pawns.

(3) Positions of pieces:

(a) Advanced knights (at e5, d5, c5, f5, e6, d6, c6, f6), especially if protected by pawn and free from pawn attack.

(b) Rook on open file, or semi-open file.

(c) Rook on seventh rank.

(d) Doubled rooks

(4) Commitments, attacks and options:

(a) Pieces which are required for guarding functions and, therefore, committed and with limited mobility.

(b) Attacks on pieces which give one player an option of exchanging.

(c) Attacks on squares adjacent to king.

(d) Pins. We mean here immobilizing pins where the pinned piece is of value not greater than the pinning piece; for example, a knight pinned by a bishop.

(5) Mobility.

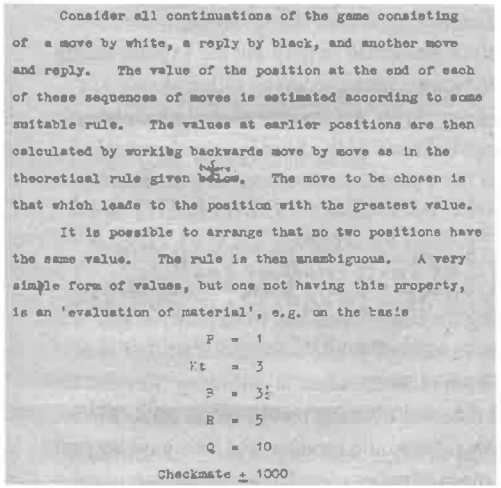
These factors will apply in the middle game: during the opening and endgame different principles must be used. The relative values to be given each of the above quantities is open to considerable debate, and should be determined by some experimental procedure. There are also numerous other factors which may well be worth inclusion. The more violent tactical weapons, such as discovered checks, forks and pins by a piece of lower value are omitted since they are best accounted for by the examination of specific variations.

There really is not all that much difference between today's programs and what Shannon conjectured, with the important exception of additional terms in the evaluation function. This is a further testament to Shannon's amazing foresight.

Prinz's program was conceptually simple to write because it only needed to give one of two values to a position: it was either a checkmate or not – nothing else mattered. But Turing was interested in having a program to play complete games of chess against humans. He wrote the specifications for a chess evaluation function

based on Shannon's ideas. He even tried programming it, but it was a large task for that time and never did get completed. However, in 1952 he decided to test his ideas out – if he could not write the program, then he would simulate it. Only someone anxious to see a chess program in action (and a little bit quirky) would have come up with the idea of having the human emulate the decision-making process of a computer. Basically, he would consider all moves in a position, manually calculate the evaluation function for each of the resulting positions using his algorithm, and then choose the move that led to the highest score. Of course this was a tedious and error-prone process, but Turing was just crazy enough (in a good way) to try it.

Turing tried his thought experiment at least twice. The first was against the wife of one of his students. Unfortunately the game score for this historic encounter has not survived. Turing annotates the second game in a paper published in 1953 but without naming the human opponent (Turing 1953). However, writer Alex Bell later tracked it done to a student named Alick Glennie. The following game represents the first game of "computer" versus human at chess. Whereas the TURK used a human to pretend to be a computer, Turing was a human actually emulating a computer. It was a step forward in technology, albeit one that was not so readily repeated!



Consider all continuations of the game consisting of a move by white, a reply by black, and another move and reply. The value of the position at the end of each of these sequences of moves is estimated according to some suitable rule. The values at earlier positions are then calculated by working backwards move by move as in the theoretical rule given below. The move to be chosen is that which leads to the position with the greatest value.

It is possible to arrange that no two positions have the same value. The rule is then unambiguous. A very simple form of values, but one not having this property, is an 'evaluation of material', e.g. on the basis

P = 1
Kt = 3
B = 3½
R = 5
Q = 10
Checkmate ± 1000

Alan Turing with the edited text of his famous 1953 paper on Computer Games.

Alan Turing computer simulation – Glennie, Alick

Irregular Open Game C42
Manchester, England, 1952

1.e4 e5 2.♘c3 ♘f6 3.d4 ♗b4 4.♘f3 d6 5.♗d2 ♘c6 6.d5 ♘d4 7.h4 ♗g4 8.a4 ♘×f3+ 9.g×f3 ♗h5 10.♗b5+ c6 11.d×c6 0-0 12.c×b7 ♖b8 13.♗a6 ♕a5 14.♕e2 ♘d7 15.♖g1 ♘c5 16.♖g5 ♗g6 17.♗b5 ♘×b7 18.0-0-0 ♘c5 19.♗c6 ♖fc8 20.♗d5 ♗×c3 21.♗×c3 ♕×a4 22.♔d2 ♘e6 23.♖g4

♘d4 24.♕d3 ♘b5 25.♗b3 ♕a6 26.♗c4 ♗h5 27.♖g3 ♕a4 28.♗×b5 ♕×b5 29.♕×d6 ♖d8 0-1

Turing (1953) summarized the game as follows:

> If I were to sum up the weakness of the above system in a few words I would describe it as a caricature of my own play. It was in fact based on an introspective analysis of my thought processes when playing, with considerable simplifications. It makes oversights which are very similar to those which I make myself, and which may in both cases be ascribed to the considerable moves being inappropriately chosen. This fact might be regarded as supporting the glib view which is often expressed, to the effect that 'one cannot programme a machine to play a better game than one plays oneself.'

Alex Bell interviewed Glennie many years after the game was played. Glennie seemed to have no appreciation that he had inadvertently become an important part of artificial intelligence history (Bell 1978).

> As I remember, he persuaded me over lunch to take part in his chess experiment. I just happened to be there and was willing to take part on the spur of the moment. It was played in the afternoon, in his office, a rather bare placed with a small untidy table with paper. We had a chessboard with pieces and Turing had his select rules written on about six sheets of paper somewhat mixed up with other paper.
>
> Laboratory gossip had told me that mechanical chess was one of Turing's interests so there were very few preliminaries before we started to play. He did explain briefly what he wanted. You can see the recorded game. It seemed to go rather slowly and I think I got slightly bored as I was not a keen player and had not played much before or since – I knew a few standard openings but none of the finer points of strategy. I was indeed a weak player: chess was for me a pleasant relaxation for odd moments with other weak players.
>
> During the game Turing was working to his rules and was clearly having difficulty playing to them because they often picked moves which he knew were not the best. He also made a few mistakes in following his rules which had to be backtracked. This would occur when he was pondering the validity of White's last move while I was considering my move. There may have been other mistakes in following the rules that escaped notice – possibly they could be detected from the record of the game. He had a tendency to think he knew the move the rules would produce and then have second thoughts. He would then try to find the piece of paper containing that section of the rules, and to do so would start juggling with all his papers. We were playing on a small table which did not help.
>
> The game took 2 or 3 hours. Turing's reaction to the progress of the play was mixed; exasperation at having to keep to his rules; difficulty in actually doing so; and interest in the experiment and the disasters into which White was falling. Of course, he could see them coming. I remember it as a rather jolly afternoon and I believe Turing must have enjoyed it too – in his way.

Unfortunately Turing died tragically at age 41. In his two decades of his work, he was extremely productive: decoding German war-time messages, defining the boundaries for what was and was not computable (today called the Turing machine), and creating the Turing Test for deciding whether a computer program was exhibiting intelligent behavior. And, of course, playing the first game of "computer" chess.

In 1956, a group of scientists at Los Alamos National Research Laboratory, home of the atomic bomb, were looking for a creative outlet (James Kister, Paul Stein, Stanislaw Ulam, William Walden, and Mark Wells). They had access to state-of-the-art computing technology and wanted to try programing something other than physics calculations. They hit upon the idea of writing a chess program. Unfortunately the computer they used, MANIAC I (Mathematical Analyzer, Numerical Integrator, and Computer), was slow and had a small memory. This limited what they could accomplish with chess. To compensate, they decided to program a simpler version of the game – a 6x6 board without the bishops, no castling, and pawns move only one square at a time (Kister *et al.* 1957). The program, initially called MANIAC, was later referred to as the Los Alamos Chess Program.

The Los Alamos program reportedly only played three games. The first game had the program play itself. The second was against Martin Kruskal, a strong player who later on became a famous mathematician and physicist. Kruskal gave the program queen odds and eventually won. In the final game, the program defeated a lady who learned to play the game a few days before. This is the first known instance of a non-simulated computer program defeating a human opponent at a chess variant.

Los Alamos Chess Program – Unnamed Opponent

Los Alamos, USA, 1956
Played on a 6x6 board. Remove all the bishops.

1.d3 b4 2.♘f3 d4 3.b3 e4 4.♘e1 a4 5.b×a4 ♘×a4 6.♔d2 ♘c3 7.♘×c3 b×c3+ 8.♔d1 f4 9.a3 ♖b6 10.a4 ♖a6 11.a5 ♔d5 12.♕a3 ♕b5 13.♕a2+ ♔e5 14.♖b1 ♖×a5 15.♖×b5 ♖×a2 16.♖b1 ♖a5 17.f3 ♖a4 18.f×e4 c4 19.♘f3+ ♔d6 20.e5+ ♔d5 21.e×f6 ♘c5 22.♕×d4+ ♔c6 23.♘e5 1-0

In 1956, it had been almost 200 years since von Kempelen was inspired to build the first chess automaton. In that time, amazing progress had been made at turning his illusion into reality. The preliminaries were over. All the key technologies were in place:

- Calculating hardware. The first general-purpose computing machine had been built, and year after year they would be improved. Further since the early 1950s, machines were commercially available, albeit at a price that few could afford. Each year the machines would get faster and the prices would drop.
- Data. Computers had internal storage and external storage. Like all computing hardware of that era, the size of the data devices and their speed of access would improve and the cost would drop.
- Software tools. 1956 was the dawn of the software age. The first mainstream compiled programming language – FORTRAN – was about to be released.
- Algorithms. Many of the key ideas needed to for a computer program had been envisioned. Computers were now available to scientists. It remained for an adventurous person to write the first functioning chess program.

- Interest. Numerous groups were interested in artificial intelligence (AI). The Dartmouth AI conference was held in 1956, heralding the start of an aggressive AI research program. Building a program to play a strong game of chess was one of the original challenge problems arising from the Dartmouth discussions.

All the technical ingredients needed to build a chess-playing program were available. Shannon supplied the recipe. Now someone had to create an entrée.

2
1600 (1957-1969)

Writing a chess program in the late 1950s was a challenging task. First, programming languages were in their infancy. FORTRAN was a huge step up from machine language, but the early program translators had a limited feature set, were buggy, and generated code that was slow to execute. Second, the machines had limited memory. A machine with 100,000 bytes of memory was exceptional, but much of this precious resource was given to the operating system and program code. Third, debugging tools were non-existent. When something went wrong, or you suspected something was not right, there was little to help the programmer find the problem. Finally, computers were costly and access was limited. Much of the early efforts in chess programming were hampered by the challenge of getting access to expensive computing hardware. Under these difficult conditions, progress in developing strong chess programs was understandably slow.

Not surprisingly, much of the early work in computer chess was to be done by people who were either an employee of a computer hardware company or worked for a university. Both had limited access to precious computer time (especially in the middle of the night).

IBM was a company who had access to the key ingredient for building a chess program: state-of-the-art computers. All they needed were programmers – a skills' set that was hard to find in those days. So, IBM did what was necessary to find these people (Wall 2014):

> IBM put an ad in the December 1956 issue of *Scientific American* and the *New York Herald Tribune* newspaper seeking anyone interested in computer programming. The ad featured a black knight chess piece, and said that "those who enjoy playing chess or solving puzzles will like this work." One of the applicants that responded to the ad was US chess champion Arthur Bisguier (1929-2017). Bisguier was then hired as an IBM programmer. ... Another applicant was Alex Bernstein, a U.S. Intercollegiate champion. ... Another applicant was Don Schultz, who became president of the United States Chess Federation. He was with IBM from 1957 to 1987.

Alex Bernstein joined IBM and ended up leading a team (Michael de Van Roberts, Timothy Arbuckle, and Martin Belsky) that developed the first program capable of playing a complete game of chess. Soon-to-be grandmaster Arthur Bisguier became the chess advisor for the project.

The program combined Shannon's Type A approach (search four ply ahead) with Type B (be selective and only consider seven "plausible" moves per position). These limits reflected the technology available at the time (Bernstein and de Van Roberts 1958):

> These limits – four half-moves ahead with seven choices at each step – are dictated by the time factor. It takes the machine close to eight minutes to decide on each move

in most cases. If it had to weigh eight plausible moves instead of seven at each level, it would take about 15 minutes for a move. If it carried the examination to one more level ahead, a single move would take some six and a half hours.

It is hard to imagine such search constraints today, where the calculation that Bernstein describes would take a fraction of a second using a modern computer.

In 1958 a movie was made of Bernstein playing a game against his program. The narrator describes this encounter, one of the first chess games ever played by a computer against a human (Education Testing Service 1958):

> To find out how good a game of chess a machine might play, Mr. Bernstein and his collaborators prepared a chess-playing program for the IBM 704, a digital computer that has performed one billion calculations in a single day in computing the orbit of an artificial satellite. The chess-playing program is given to the 704 on a reel of magnetic tape.
>
> On the chessboard itself the moves are made by Mr. Bernstein for both players. As he makes a move, he communicates it to the machine. The machine prints out the position of all the pieces: its own and its opponents, to correspond with the chessboard on every move. In calculating its moves, the machine considers the board square by square. Is the square occupied? By whose man? Is it under attack? Can it be defended? Can it be occupied? All this has taken a long time by computer standards: one-tenth of a second. Now the computer proceeds to select its move. It has about 30 possible moves.
>
> After asking eight preliminary questions about each of them, it selects seven of the 30 for further analysis. It tests each of the seven through four moves ahead, considering its opponent's possible replies and its own possible counter responses in each case. It examines 2,800 positions in eight minutes. Now the machine prints out its move. It elects to take the opponent's knight with its own bishop. Mr. Bernstein takes the machine's bishop with his queen. The move is recorded. But the machine rejects the move as illegal. The difficulty is an incorrect coding, which is corrected.
>
> The game continues with the machine playing methodically and tirelessly. It's never absent-minded and never makes an obvious blunder. In individual moves, it often plays like a master. In a complete game, it can defeat an inexperienced player, but can be outwitted by a good one. This game has gone up to the 21st move. Mr. Bernstein attacks strongly, threatening the machine's knight with his castle. He records the move. The machine's response is a useless pawn move. Its unprotected knight is lost to Mr. Bernstein's castle. The machine recognizes its position as hopeless and resigns.
>
> After losing a game, the machine will still make the same moves again and lose in the same way. Some day, though not soon, Mr. Bernstein feels, a program may be designed that will enable the computer to profit by its own mistakes and improve its chess game on the basis of its experience against human opponents.

Note some of the amazing statements in the text. The computer did one billion calculations in a single day. In contrast, modern computers will do several billion in a second. The chess program analyzed 2,800 positions in eight minutes. That works out to almost six positions per second. These numbers help explain the weak play of the chess programs of that era. How strong would Komodo, Stockfish, or Houdini be if they could only analyze six positions per second?

The following game illustrates the limitations of the program's search algorithm. The opponent is described as "skillful" (Bernstein and de Van Roberts 1958).

Bernstein Chess Program – Unnamed Opponent

Irregular Open Game C96
Undated

1.e4 e5 2.♗c4 b6 3.d3 ♘f6 4.♗g5 ♗b7 5.♗×f6 ♕×f6 6.♘f3 c6 7.0-0 d5 8.e×d5 c×d5 9.♗b5+ ♘c6

This clearly shows the disadvantage of the program's search constraints. 10.♘×e5 wins a pawn as ♕×e5 11.♖e1 loses. The material win is missed either because one of the key moves is not in the seven possibilities considered per position, or after Black's 11th move, say 11... 0-0-0, Black is up a piece – the four-ply limit has been reached!

10.c4 d×c4 11.♗×c6+ ♕×c6 12.d×c4 e4 13.♘g5 ♕g6 14.♘h3 e3 15.f3 ♗c5 16.♖e1 0-0 17.♘c3 e2+

Such a horrendous ending is the result of the search limitations discussed above.

0-1 (in 22 moves)

> **Alex Bernstein and Michael de Van Roberts (1958)**
> **Chess program developers**
>
> Even with much faster computers than any now in existence it will be impractical to consider more than six half moves ahead, investigating eight possible moves at each stage.

Concurrent with Bernstein *et al.*'s work, the team of Allen Newell (1927-1992), John Cliff Shaw (1922-1991),[1] and Herbert Simon (1916-2001) began using chess as a model application for their research in creating human-like approaches to computer problem solving. Newell and Simon worked at Carnegie Tech (later Carnegie Mellon University) and Shaw worked at the RAND Corporation (a non-profit research organization).

Newell and Simon shared a strong belief that artificial intelligence success could be achieved by having computers loosely mimic the problem solving strategies used by humans. Thus a brute-force Shannon Type A search approach was not of interest to them. Simon (1978) expressed his philosophy as such:

> Information processing theories envisage problem solving as involving very selective search through problem spaces that are often immense. Selectivity, based on rules of thumb or "heuristics," tends to guide the search into promising regions so that solutions will generally be found after search of only a tiny part of the total space.

The result of their research was the NSS program, named using the initials of their last names. It used goals to guide the search. Human experience was translated into

heuristics to help inform the program's understanding. Heuristics were used to help select subsets of moves to consider, decide on how deep to analyze a line of play, and to assess a position. Simon provided the chess expertise, while Newell and Shaw integrated this knowledge into the program.

From Newell and Simon's writing, we know that they understood that some of the positions considered by a minimax search were provably irrelevant: the outcome of evaluating those positions could have no effect on the final result of the search. Hence, they should be ignored; they were a waste of computational resources. This idea later became known as the alpha-beta (α-β) algorithm. The historical record is not clear as to whether Newell and Simon understood this algorithm in its full generality to eliminate all possible irrelevant positions.

NSS pioneered the use of a high-level programming language for chess; all efforts to date had used machine/assembly language. If programming chess was not enough of a challenge, Newell, Shaw, and Simon had to design and implement their own programming language! Their language, IPL (Information Processing Language), was running in 1957 and the following year it was used to create the NSS chess program.

The following game has Herbert Simon testing NSS.

NSS – Simon, Herbert

Irregular Closed Game D00
Undated

1.d4 ♘f6 2.♘c3 d5 3.♕d3 b6 4.e4 ♗b7 5.e×d5 ♘×d5 6.♘f3 e6 7.♗e2 ♗e7 8.♗e3 0-0 9.0-0 ♘d7 10.♖fe1 c5 11.♖ad1 ♕c7 12.♘×d5 ♗×d5 13.a4 ♖ac8 14.♕c3 ♗f6 15.♗b5 ♗×f3 16.g×f3 ♖fd8 17.♗×d7 ♕×d7 18.b3 c×d4 19.♕d2 ♕c6 20.♗f4 ♕×c2 21.♕×c2 ♖×c2 22.♖c1 ♖dc8 23.♖cd1 ♖c8c3 24.b4 ♖×f3 25.♗g3 d3 26.♖c1 ♗g5 27.♖×c2 d×c2 28.♗e5 c1♕ 29.♖×c1 ♗×c1 0-1

In 1958, NSS achieved a milestone by defeating a human player at chess. History was made – never mind that the opponent was taught the rules of chess shortly before the game was played. However, as quickly as NSS came on the scene and achieved an important milestone, it just as quickly disappeared. Progress in realizing Newell and Simon's artificial intelligence objectives were slow and the researchers moved on to other applications.

There is no Nobel Prize in Computer Science. To address that concern, in 1966 the Association of Computing Machinery (ACM) created the Turing Award, in recognition of the pioneering work of Alan Turing. The 1975 winners were Newell and Simon, in part for their artificial intelligence research that was applied to chess. Herbert Simon had an amazing career, as he also won the Nobel Prize for Economics (1978) and the Outstanding Lifetime Contributions to Psychology prize (1993).

Herbert Simon (seated) and Allen Newell.
(Carnegie Mellon University)

Herbert Simon (1957)
Scientist and Nobel Laureate

Within 10 years a digital computer will be the world's chess champion unless the rules bar it from competition.

Next on the scene was MIT undergraduate student Alan Kotok (1941-2006). He took an undergraduate programming course from MIT junior professor John McCarthy (1927-2011) in 1959. Fellow classmate Elwyn Berlekamp recounts that course (Berlekamp 2016):

> It may have been one of the first courses in programming that was taught anywhere that I was aware of, in the United States at least. It was the spring of 1959 and it was a freshman elective for which a bunch of us signed up. And McCarthy… got people doing various projects and four of us got together and decided to write a program to play chess.
>
> *Question: Who were the four?*
>
> Me., Kotok, [Charles] Niessen, and [Michael] Lieberman.
>
> We divided this project up… I remember I got very excited about and did a piece that related to searching for very quick mates… [The program] certainly made legal moves which itself was something of an achievement… It might be ranked a beginner. All the programmers could easily beat it.
>
> So, we turned it in and McCarthy gave all four of us "A"s for the project. We all went

> our separate ways except for Kotok who really became attached to this and did several upgrades to the program. Although it started as a project in which all of us were more-or-less equal contributors, it ended with a program that was largely Kotok's, I'd say 90%.

Kotok and new collaborators Paul Abrahams, Robert Wagner, and B.F. Wells continued to refine the program for a few years. It was largely written in FORTRAN, which would become the programming language of choice for chess programs for most of the next two decades. The resulting effort is often referred to as the Kotok chess program in part because of Kotok's major role in developing it, but also because he documented it for his B.Sc. thesis.

McCarthy was in the Newell and Simon camp when it came to artificial intelligence, so it is not surprising that the Kotok program used a Shannon Type B search strategy. The selectivity was achieved much like Bernstein's effort, by limiting the number of moves considered in a position. Of historical importance was the elimination of positions from the search that had no impact on the final result. John McCarthy introduced this idea to the project, but it is unclear whether this was an independent idea of his or inspired by Newell and Simon. However there is no question about the name of the idea – "alpha-beta" came from McCarthy. In his 1962 B.Sc. thesis, Kotok tries to describe the algorithms (Kotok 1962):

> The program was tested late in the spring of 1961. The [IBM] 709 took about 5 to 20 minutes per move, depending on the complexity of the situation. Although the machine did not do too badly, we noted that it was looking at many irrelevant positions. We therefore attempted to find a method of pruning the move tree, without discarding good as well as bad moves.
>
> Prof. McCarthy proposed a heuristic for this purpose, called "alpha-beta." It operates as follows: Alpha is a number representing the value of the best position which white can reach, using a pessimistic evaluation. Beta represents the best position white can reach, using an optimistic evaluation, due to the fact that black can hold him to this position. Under normal circumstances, alpha starts at -infinity, and beta at +infinity. At each level, optimistic and pessimistic evaluations are made, and compared to alpha and beta in the following way. If a white move is optimistically less than alpha, it is discarded, since a better alternative exists elsewhere. Likewise, if a white move pessimistically is better than beta, it too is discarded, since black had a better alternative previously; furthermore we revert two levels since no other white moves are worth considering at that position. The reverse strategy is applied for black.

Can you understand this? From this description it is hard to figure out what is really happening. Further the alpha-beta idea is described as a heuristic whereas it is in fact stronger than a heuristic – it is a provably correct way of reducing the search effort. Thus, although Newell/Simon and McCarthy had the right idea, it is unlikely either group understood the full potential of what would soon become known as the alpha-beta (α-β) algorithm.

The program only played a few partial games – ugly chess by both sides – before Kotok graduated. He concluded "From our analysis of the results, we have found that in its present state, the program is comparable to an amateur with about 100 games experience" (Kotok 1962). A doubtful claim, to say the least.

1962: Checkers

Arthur Samuel's checkers program (IBM) plays a six-game exhibition match against Robert Nealey, a future Connecticut State Champion. Nealey wins the match by a score of two wins to one with three draws. This is the first time a computer defeats an opponent of creditable strength in an official competition.

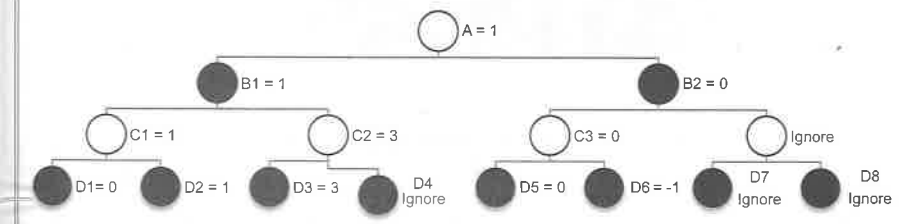
Alpha-Beta Algorithm

The alpha-beta search algorithm is an enhancement to Minimax search. By keeping track of two numbers, the best score that each player can achieve, large parts of the search tree can be proven to be irrelevant to the final result.

Alpha-beta search can be thought of in terms of an obvious principle: once you know you can do no better, then stop. Let us return to our two chess friends, Max and Min. As before, consider that each position has 40 moves to consider and that a position is evaluated solely by material. Max considers the first move and it wins a pawn. On to the second move, which wins nothing. So far so good. Now the third move leads to checkmate. Wonderful! What do we do now? There is no point in doing more analysis here; we know the result. The remaining 37 moves can be ignored.

This idea can be generalized. Consider a position with Min to move. Assume that Min analyzes the first move and finds that he loses a pawn. Clearly Min wants to do better; a line of play that loses a knight is irrelevant. Min tries a move. Max analyzes his first move and it wins a knight. Now, you might think that Max should keep searching, hoping to win more than a knight. In fact, there is no point to doing any more work. From Min's point of view, this line of play is irrelevant. By playing this move, Min loses at least a knight and possibly more. Min will always prefer to take the move that loses the pawn to the move that loses the knight (or more).

The idea is illustrated in the following search tree. At position B1 Min (Black) knows that the first move leads to the loss of a pawn (C1). Can he do better with the second move? At position C2, Max (White) wins a knight by playing to D3. Now we know that further improving this score is irrelevant to Min (Black) at B1 – Min will always choose to lose a pawn over a knight. Hence, position D4 can be ignored. Using the tree analogy, such branches (moves) in the search are said to be "pruned."



In this example, three of the eight leaf nodes do not need to be considered (D4, D7, and D8), a saving of 37.5%. Can you do better than that? Pretend that every position has 40 possible moves and that a search program wants to look ahead

10 ply. The number of leaf nodes in this search tree is the staggering 10^{10} = 10,485,760,000,000,000. If one is fortunate enough to eliminate the maximum possible leaf nodes, then the search tree shrivels down to the reasonable size of 10^5 = 102,400,000. In practice, one cannot obtain the maximum possible savings. However, modern programs usually search within a small factor of the minimum search tree size.

The alpha-beta algorithm is amazing. Searching 10 ply with Minimax in a reasonable amount of computer time is not possible. If your chess program could analyze 100 million positions in a second, then this Minimax search would take over 1,000 days to produce a result. Add in the alpha-beta enhancement and, voilà, the seemingly impossible becomes easy! The search takes one second.

Who invented the alpha-beta algorithm? There are four competing claims, but there is not enough information available to identify a "winner." Newell, Shaw, and Simon certainly discussed the notion of eliminating provably irrelevant positions from the search tree. Alan Kotok mentions the algorithm in his 1962 thesis and credits it to John McCarthy. A 1961 short paper by D.J. Edwards and T.P. Hart demonstrates "The α-β Heuristic." Also, Arthur Samuel (1901-1990) had a variant of the idea in his 1962 checkers program. In 1963, Russian researcher Alexander Brudno (1918-2009) published a paper describing and analyzing the algorithm, which he says he discovered a few years earlier. However, in the early 1960s the Cold War was in full force, and Brudno's work was not known in the West until many years later. A formalization and thorough analysis of the algorithm did not appear until 1975.

Alan Kotok (seated) programming the PDP 6 computer. Computing pioneer Gordon Bell looks on. (ComputerHistory.org)

Alan Kotok graduated and went on to an illustrious computing science career. In 1962 John McCarthy moved to Stanford University where he became the leader of their fledgling artificial intelligence research program. The Kotok program came with him, and McCarthy tinkered with improving it. This new version is often called the Kotok-McCarthy program in the literature.

P.H. Clarke (1963)
Chess author

Will the world champion in 2000 be a man or a machine? This was the subject for discussion on Moscow television recently by Grandmasters [Vasily] Smyslov, [David] Bronstein, and [Yuri] Averbakh. As might be expected, no agreement was reached. Averbakh held that in about 35 years' time scientists will be able to design a machine perfect enough to compete with masters and even grandmasters and that it will have many advantages over man. Smyslov was of a different opinion. He compared chess with music, asserting that just as a mechanical composer could not rival human fantasy, so a machine could not play better chess than a man.

Bronstein took a middle line, surmising that by the year 2000 there would be separate championships for men and machines. I wonder who will be right.

Israel A. Horowitz and Philip L. Rothenberg (1963)
Chess authors

That a richly endowed robot will one day be able to play a highly skillful game of chess leaves no room for doubt. On the other hand, in the absence of a fantastic super-speed electronic brain, the chess championship of the world is likely to be retained by humans for centuries to come.

In 1965, McCarthy visited Moscow and found out about a chess program being developed at the Institute for Theoretical and Experimental Physics (ITEP). The team consisted of Georgy Adelson-Velsky (1922-2014), Vladimir Arlazarov, Anatoly Uskov, and Alexander Zhivotovsky, with chess expertise from former World Champion Mikhail Botvinnik (1911-1995) and master Alexander Bitman. ITEP Director Alexander Kronrod and John McCarthy agreed to a four-game correspondence match between their programs. Given the tensions between the two countries, the match became more than just a scientific experiment; it became a competition between the USSR and the USA.

1966: Checkers

Arthur Samuel's checkers program (IBM) played four-game exhibition matches against World Champion Walter Hellman and his challenger Derek Oldbury. The grandmasters won all eight games rather easily. This was the first time that a world champion publicly played a computer program.

The match began in November 1966 and lasted nine months. Every few days a move would be telegraphed to the opposing side; access to computing resources being the limiting factor. The Soviet program won by a score of two wins and two draws. At the time it was not appreciated that the result was to portend the future of computer chess. First, the Soviet program (which had no formal name) used

a Type A search whereas the Kotok-McCarthy program used a Type B. Science had its first data point suggesting which approach was better. Second, the Soviet program played two games with a search limitation of three ply; both were draws. Two games were played with a five-ply limit; both were wins. Thus there was the first evidence of a strong correlation between computational resources and chess performance.

As an historical footnote, it is sad to report that Dr. Kronrod was demoted at ITEP due to complaints that too much valuable computing time was being squandered on chess. In contrast, John McCarthy won the Turing Award in 1971, in part for his work on computer chess.

Georgy Adelson-Velsky (left) and John McCarthy playing chess. Perhaps this game was the human version of the USSR versus USA match. (Semen Karpenko)

It is not surprising that with Kotok's and McCarthy's work on computer chess at MIT getting some scientific exposure that a new leader would emerge. Undergraduate student Richard Greenblatt (1962) recounts his start in chess and computer chess:

> I got involved in computer chess after I visited the Stanford Artificial Intelligence Laboratory during a trip to the Fall Joint Computer Conference in November 1966, which was held in San Francisco. I had played chess in grade school, and used to play against university students at the Student Union of the University of Missouri in Columbia, Missouri, where I grew up; but I had never played in a tournament and did not play significantly after I came to MIT as a freshman in the fall of 1962.
>
> At that time there was an ongoing match between the Kotok program (that had moved to Stanford with Prof. John McCarthy) and a Russian program. Examining some computer printouts, it was immediately evident to me that the standard of play and analysis were very low, and I returned to [MIT in] Boston and resolved to do better. After one month, I had the computer playing chess, and the following [January] we decided to enter a human tournament, feeling then, as I still do, that human competition is the best way to measure progress in computer chess, particularly from the AI point of view.

Unlike previous efforts, Greenblatt was able to quickly build a complete program (weeks, instead of many months or years). He had the advantage of Kotok's work as a model, as it was completely published in his thesis.

The program was given the rather esoteric name MacHack VI, the amalgamation of three components of the work. First, Greenblatt did his work under the umbrella of the MIT research program called Project MAC (Multiple Access Computing to some; Machine-Aided Cognition to the artificial intelligence researchers). Second, what he was doing with his software development was "Hack"ing, a term that had its origins in the early 1960s MIT computer culture. Greenblatt was one of the new generation of sophisticated computer experts. The word had none of the negative connotations that it sometimes has today. Finally, he wrote his program to run on the Digital Equipment Corporation (DEC) computer PDP 6 (VI). Hence the formal name MacHack VI, although usually referred to simply as MacHack.

The combination of a more powerful computer, tuned program code, and the alpha-beta algorithm allowed MacHack to search deeper and to consider more moves in a position than previous attempts. The standard program search was 5-ply deep. Like Kotok's program, at each ply it only considered plausible moves: 15 moves at the first ply, 15 at the second, 9 at the third, 9 at the fourth, and 7 at the fifth (Greenblatt *et al.* 1967). The extra breadth reduced the probability of making a catastrophic blunder because of missing a plausible move. Leaf nodes values included the result of a quick check for any obvious tactics (such as following all capture sequences), often called a quiescence search.

The MacHack program pioneered two important computer chess ideas. The first was the use of an opening book. Why compute the opening moves every game when there were plenty of books available giving the main lines of play for each of the openings? Larry Kaufman (then a master, and later a grandmaster) and Alan Baisley created the opening book. Greenblatt recalls the opening book effort: "By later standards it wasn't so big, but at the time it was pretty good sized. I don't know I think it was probably 8,000 or 10,000 moves in there" (Greenblatt 2005).

The second idea was the use of a transposition table. During a search, the same position can arise multiple times. The first time a position is analyzed, remember the result. If you encounter the same position again later on in the search, you may be able to use the result of the previous computation. It is an obvious observation, but no one had yet done it. Transposition tables, named as such because the same position could arise as a result of move transpositions, had the potential to eliminate duplicate parts of the search.

Quiescence Search

An important consideration in a chess program is when to stop the analysis of a line of play and evaluate the resulting position. The ideal position to assess is one that is quiet, or *quiescent*. Shannon already figured this out in his amazing paper:

> A very important point about the simple type of evaluation function given above (and general principles of chess) is that they can only be applied in relatively quiescent positions. For example, in an exchange of queens White plays, say, ♕×♕ (×=captures) and Black will reply while White is, for a moment, a queen ahead, since Black will immediately recover it. More generally it is meaningless to calculate an evaluation function of the general type given above during the course of a combination or a series of exchanges.

There are several obvious rules the one might use for deciding if a position is not quiescent: the side to move is in check, the side to move can give check, the side to move can immediately recapture a piece, a valuable piece is *en prise*, and multiple pieces are attacked (e.g., a fork). In effect, the program is trying to resolve what looks like a forced sequence of moves. Part of the art/science of building a chess program is deciding on the rules for determining if a position is quiescent.

Modern chess programs will search until some criterion is met (e.g., a prescribed search depth is reached). Before doing the evaluation, a quiescence search will be done. This search is restricted to only the moves that help resolve quiescence. The goal is to extend the analysis until a quiet position is reached – only quiet positions should be evaluated.

Greenblatt was ambitious and he was able to get MacHack entered into the monthly Boylston Chess Club Tournament in Boston. On January 21, 1967, in the first round, MacHack VI, using the pseudonym of "Robert Q" was paired against Carl Wagner. Wagner, with a 2190 rating, had no problem winning this historic game.

Wagner, Carl (2190) – Robert Q (unrated)

Irregular Opening A00
Boylston Chess Club Tournament (1), 1967

1.g3 e5 2.♘f3 e4 3.♘d4 ♗c5 4.♘b3 ♗b6 5.♗g2 ♘f6 6.c4 d6 7.♘c3 ♗e6 8.d3 e×d3 9.♗×b7 ♘bd7 10.e×d3 ♖b8 11.♗g2 0-0 12.0-0 ♗g4 13.♕c2 ♖e8 14.d4 c5 15.♗e3 c×d4 16.♘×d4 ♘e5 17.h3 ♗d7 18.b3 ♗c5 19.♖ad1 ♕c8 20.♔h2 ♘g6 21.♗g5 ♖e5 22.♗×f6 g×f6 23.♘e4 f5 24.♘f6+ ♔g7 25.♘×d7 ♕×d7 26.♘c6 ♖be8 27.♘×e5 ♖×e5 28.♕c3 f6 29.♖d3 ♖e2 30.♖d2 ♖×d2 31.♕×d2 ♘e5 32.♖d1 ♕c7 33.♗d5 ♔g6 34.b4 ♗b6 35.♕c2 ♘c6 36.♗e6 ♘d4 37.♖×d4 ♗×d4 38.♕×f5+ ♔g7 39.♕g4+ ♔h6 40.♕×d4 ♕e7 41.♕h4+ ♔g6 42.♗f5+ ♔f7 43.♕×h7+ ♔f8 44.♕h8+ ♔f7 45.♕a8 ♕c7 46.♕d5+ ♔g7 47.♔g2 ♕e7 48.h4 ♔h6 49.g4 ♔g7 50.h5 ♕e2 51.h6+ ♔f8 52.h7 ♕×f2+ 53.♔×f2 ♔e7 54.h8♕ a6 55.♕e6# 1-0

But there was a positive note at the end of the event (Krakauer 2010):

> Although we entered our first five-game tournament with high hopes, the program lost its first four games. In the fifth, though, it arrived at the endgame in decent shape, and the programmers decided to offer the opponent a draw. The opponent, an elderly man who perhaps imagined that the computer would be very strong in the endgame, accepted. In fact, the program was rather bad in its endgame play at the time, and the opponent might well have prevailed had he persisted. Richard Greenblatt noted, in a recent e-mail message, "The opponent was an older guy who was a local chess legend and rated about 1400. He was also a bit of a promoter, so although the game appeared legit, we were a bit leery about 'counting' it too much."

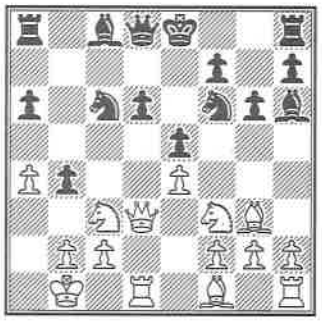
The program finished with a score of one draw and four losses, achieving a United States Chess Federation (USCF) rating of 1239. This was sufficient to win the Class D championship of the tournament, and Robert Q (aka MacHack VI) received a trophy. The era of man-machine competition was officially under way!

Greenblatt's appetite was whetted and a few months later MacHack VI competed in the Massachusetts State Championship. It was amazing what a few program enhancements and bug fixes could do.

MacHack VI (unrated) – Unnamed Opponent (1510)

Irregular Sicilian B21
Massachusetts State Championship (2), 1967

1.e4 c5 2.d4 c×d4 3.♕×d4 ♘c6 4.♕d3 ♘f6 5.♘c3 g6 6.♘f3 d6 7.♗f4 e5 8.♗g3 a6 9.0-0-0 b5 10.a4 ♗h6 11.♔b1 b4



12.♕×d6 ♗d7 13.♗h4 ♗g7 14.♘d5 ♘×e4 15.♘c7+ ♕×c7 16.♕×c7 ♘c5 17.♕d6 ♗f8 18.♕d5 ♖c8 19.♘×e5 ♗e6 20.♕×c6+ ♖×c6 21.♖d8#

History was made!

Lawrence Krakauer was a witness to the exciting days of MacHack's tournament adventures. Here he recalls a game that illustrates the emotional side of watching your programming creation compete (Krakauer 2010):

> In the particular game I'm describing, the computer was doing quite well, and it looked as if it would win. On the other hand, it could be weak in the endgame, so we knew that it might not pull it off. A master-level player working with the team was observing the game with us. I think it may have been Larry Kaufman, a student at

MIT who was a national master, but I'm not sure of that. Suddenly, he said something like, "Wow, look at this!," and he pointed out a mating combination for the program – the game was won! Except, of course, there was the question of whether or not the program would "see" the same possibility he had spotted. After all, of all the people following the game, only he had seen it.

We ran over to the line printer, and looked at the plausible move list, which had already been printed. The initial move of the mating combination was on the list, of course, since *all* moves are on the list. But it was ranked dead last - the Plausible Move Generator module of the program considered it to be absolutely the least-likely move for the computer to choose. This was not surprising, since it was a queen sacrifice. We knew that if the computer looked ahead a few ply... it would have been likely to see the mating combination. But of course, you can't see it if you don't look, and why waste time looking at a move that will result in the immediate capture of your queen?

Oh, well. How could we expect the program to find a tricky mating combination that only one observer had noticed, and he was a master-level player?

We waited for the program's move, but the program seemed to be taking longer than usual to do its analysis. Finally, the line printer chattered, and we knew that at the same time, the move was being typed out at the tournament site. It was the queen sacrifice, and the program's board evaluation, the large positive number that we considered to be "plus infinity," indicated that the program had seen the forced mate, and thus "knew" that the game was definitely won. That, in fact, accounted for the extra time the program had taken to announce its move. Upon seeing what looked like a mating combination, the program had evaluated all possible responses of the opponent at each level of the look-ahead, to be certain that the checkmate could not possibly be avoided.

We were all delighted, of course, but Greenblatt's delight was tempered by puzzlement. We could understand how the program had seen the mate once it had looked in depth at the queen sacrifice, but why had it evaluated that move at all, the lowest of the low in terms of its likelihood of success? If I'm recalling correctly, Greenblatt actually pulled out the thick source code listing, and started looking through the program to figure out what might have happened.

And then it hit him. There's an old chess adage, "When in doubt, check." The queen sacrifice was a checking move, and the program had been written to evaluate *ALL* checking moves, no matter how dubious they seemed. Nobody had particularly noticed that the queen move was a check. I mean, we saw it, obviously, but it's hardly important that the queen checks the opponent when the queen is going to be immediately captured. But it was that incidental attribute of the move that had caused it to be evaluated by the look-ahead module, which had exposed the mate.

The phone rang. It was one of our representatives at the tournament, all upset. The computer had been winning, he wailed, and now it's throwing away the game by giving up its queen! I was glad to know that I wasn't the only one who hadn't seen the mating combination.

Watch and learn, we told him, watch and learn.

MacHack's tournament career was short. Greenblatt (1992) summarized it by saying that, "MacHack went on to play in about half a dozen human chess tournaments. Its best results were drawing an 1880 player and beating a 1720 player. Its best performance rating in a tournament was 1820 and I believe its official USCF rating was 1523." In recognition of the program's trailblazing success, it was made an honorary member of the USCF.

Transposition Tables

Consider a search where the line of play 1.d4 d5 2.♘f3 is followed by a five-ply analysis. Later on in the same search, the program considers the sequence 1.♘f3 d5 2.d4. The second line of play is said to transpose into the first. If the five-ply result of the first piece of analysis has been saved somewhere, then when the second move sequence occurs, the transposition can be detected and the five-ply value that has already been calculated can be used.

The idea of a transposition table is intuitively obvious. Whenever you search a position, save the position and the search result. When you come across a new position, check to see if it has been previously searched and, if so, you may be able to reuse the result. Note that just because a transposition has been detected does not automatically mean one can throw away the repeated position. It might be that the first time the position is searched to, say, a depth of five, but when the transposed position arises, a search to depth seven is required. In this case the previous result cannot be used.

It turns out that the analysis done by chess programs is full of transpositions. The simple trick of not repeating a calculation can result in a large reduction in the search effort. Depending on properties of the position being analyzed and the depth of search, it is possible to see scenarios where the program runs hundreds of times faster. In complex middlegame positions, the savings are much smaller but still substantial.

Transposition tables are sometimes called hash tables, a name that describes the implementation rather than the idea. Hashing is a computer science algorithm that takes a chess position and "hashes" it into a number. This number is used to specify the table row in which the position is to be stored. The search typically proceeds as follows:

- Take the current position and turn it into a number N.
- Look in table entry N to see if the position is present.
- If present use the result to possibly end further search of this line of play.
- If not present, search this position and at the end of the search store the position and the result into table entry N.

Clearly the larger the transposition/hash table, the more information that can be saved, and the greater the likelihood that transpositions can be detected. Most chess programs will fill all of available memory with the table.

We conclude MacHack's tournament record with a game showing that Greenblatt's fear of the endgame was well founded. In the following game, MacHack squanders

a winning position. Note that the program is referred to as MacHack VII, a reflection of the newer and faster computer (PDP 10) that the program was using.

Haley, Philip – MacHack VII

Reti Opening A08
Labor Day Open (3), Toronto, 31.08.1969

1.♘f3 ♘f6 2.g3 d5 3.♗g2 c5 4.d3 ♘c6 5.♗f4 ♗f5 6.0-0 ♕b6 7.♘c3 ♕×b2 8.♕d2 d4 9.♘e4 ♘×e4 10.d×e4 ♗×e4 11.♖fb1 ♕c3 12.♕×c3 d×c3 13.♖×b7 ♘b4 14.♖b8+ ♖×b8 15.♗×b8 ♘×c2 16.♖c1 ♘d4 17.♘×d4 ♗×g2 18.♘b5 ♗h3 19.♗×a7 ♗d7 20.♘c7+ ♔d8 21.♘d5 c4 22.♘×c3 e5 23.♘e4 ♗b5 24.♖b1 ♗c6 25.♘c3 ♗d7 26.a4 ♗f5 27.e4 ♗c8 28.a5 ♗a6 29.♖b6 ♔c8 30.♘d5 f6 31.♖c6+ ♔d7 32.♖×a6 ♔c8 33.♖b6 ♔d7 34.♖b8 ♔c6 35.♗e3 c3 36.♘×c3 1-0 (in 47 moves)

Despite MacHack's successes, objectively the program's play was no better than Class C.

MacHack's most famous game was not played in a tournament. In 1965, MIT professor Hubert Dreyfus published an article titled "Alchemy and Artificial Intelligence." As the title suggests, it was a scathing critique of the progress made and the potential for success of artificial intelligence research. Dreyfus singled out the work of Newell, Shaw, and Simon. He is particularly barbed in his comments about their progress, predictions, and reporting of results (Dreyfus 1965):

> The chess-playing story is more involved and might serve as a study of the production of intellectual smog in this area. …
>
> In fact, in its few recorded games, the NSS program played poor but legal chess, and in its last official bout (October 1960) was beaten in 35 moves by a ten-year old novice. Fact, however, had ceased to be relevant. Newell, Shaw, and Simon's claims concerning their still bugged program had launched the chess machine into the realm of scientific mythology. …
>
> While their program was losing its five or six poor games – and the myth they had engendered was holding its own against masters in the middle game – Newell, Shaw, and Simon kept silent. When they speak again, three years later, they do not report their difficulties and disappointments … [and] gives the impression that the [within 10 years] chess prediction is almost realized. With such progress, the chess championship may be claimed at any moment. Indeed a Russian cyberneticist, upon hearing of Simon's 10-year estimate, called it "conservative." And Fred Gruenberger at RAND has suggested that a world champion is not enough – that we should aim for "a program which plays better than any man could." This output of confusion makes one think of the French mythical beast which is supposed to secrete the fog necessary for its own respiration.

And this is only the introduction of the paper! Dreyfus had made the astute observation that the field of artificial intelligence, not just computer chess, suffered from unrealistic expectations and unfulfilled predictions. Not surprisingly, this was a message that many did not want to hear, and the article attracted few friends but

many fierce opponents. Dreyfus later expanded on his ideas in the widely read book *What Computers Can't Do*.

Dreyfus was a bit unlucky in that within a year of writing his paper, Greenblatt produced a reasonably strong chess program. And, since Dreyfus enjoyed playing chess, there was but one way to settle the score: on the chessboard. Richard Greenblatt recounts MacHack's most famous game (Greenblatt 2005):

> Well, there was a guy at MIT in those days named Hubert Dreyfus, who was a prominent critic of artificial intelligence, and made some statements of the form, you know, computers will never be any good for chess, and so forth. And, of course, he was, again, very romanticized. He was not a strong chess player. However, he thought he was, or I guess he knew he wasn't world class, but he thought he was a lot better than he was. So anyway, I had this chess program and basically Jerry Sussman, who's a professor at MIT now, … brought over Dreyfus and said, well, how would you like to have a friendly game or something. Dreyfus said, oh, sure. And sure enough, Dreyfus sat down and got beat. So this immediately got quite a bit of publicity.

Dreyfus, Hubert – MacHack VI

Italian Game C50
Dreyfus match, MIT, 1967

1.e4 e5 2.♘f3 ♘c6 3.♗c4 ♘f6 4.♘c3 ♗c5 5.d3 0-0 6.♘g5 ♘a5 7.♗d5 c6 8.♗b3 ♘×b3 9.c×b3 h6 10.♘h3 d5 11.e×d5 ♗g4 12.f3 ♗×h3 13.g×h3 ♘×d5 14.♘×d5 ♕×d5 15.♗d2 ♕×d3 16.b4 ♗e7 17.♖g1 e4 18.f×e4 ♗h4+ 19.♖g3 ♗×g3+ 20.h×g3 ♕×g3+ 0-1 (in 37 moves)

Although the game was not well played (especially by Dreyfus), all that really mattered was the result. Herbert Simon (1967) was particularly sarcastic in his response to the game:

> What are the facts? A man who exhibited great zest in writing that a "ten-year old novice" had beaten a particular chess program was himself beaten, and beaten roundly, by MacHack. Neither fact by itself proves much about the present or future of chess programs, but the two facts may interest and arouse emotions in persons already passionately committed to conclusions (pro or con) on these mutters. To protest amused comment on the MacHack victory shows either a desire to apply the rules of rhetoric asymmetrically, or such deep emotional involvement as to cause blindness to the asymmetry. You should recognize that some of those who are bitten by your sharp-toothed prose are likely, in their human weakness, to bite back; for though you have considerable skill in polemic, you have no patent on it.
>
> The discussion of the philosophy and status of artificial intelligence would benefit from de-escalation. Since you have contributed some of the most vivid prose on the subject, may I be so bold as to suggest that you could well begin the cooling – a recovery of your sense of humor being a good first step. You see, the real humor in the Dreyfus-MacHack game, as any chess player who plays it over will tell you, is not that you were beaten, the humor is that the Greenblatt program exhibited in this game many of the same human failings that you did (failing to see obvious impending

> mates, for example), and still clobbered you by the skin of its teeth. It was a real cliffhanger, in which one fringe unconsciousness was outdone by another. MacHack behaved not like an "omniscient computer" (to quote you out of context), but like a frail and sometimes desperate humanoid even, shall we say, as you and I.

Academic debates are usually not so spirited! In the end, it would be fair to say that both sides agreed to disagree. Dreyfus was a pariah to much of the AI community for many years. However, with hindsight, the AI community begrudgingly will admit that much of what Dreyfus wrote was correct.

In 1992, Richard Greenblatt reflected back on his pioneering work in computer chess (Greenblatt 1992):

> We say a system is wedged if there exists a binding, a clashing deep within its bowels, that prevents progress that you would otherwise expect. …
>
> Looking back, I believe the field of computer chess was wedged when I got involved in it 26 years ago. It was not merely the state of ignorance, although that was great, but a certain "romantic" ideal, among philosophers and mathematicians, that was inhibiting progress. Most of these individuals were not strong chess-players, but some were. Former World Champion Botvinnik wrote a book claiming to be about computer chess, which devolved into a discussion of his famous combination against Capablanca. Thus, we may say, the philosophers, the mathematicians and the chess grandmasters of that time were all more or less equally wedged.

Much like Alan Kotok. Richard Greenblatt was given the opportunity to turn his work on computer chess into his B.Sc. thesis. Unlike Kotok, Greenblatt never got around to it and did not get academic credit for his groundbreaking work. Of course, what he lacked in a degree, he more than made up for in international recognition for his work.

MacHack was developed for the PDP series of computer, a line of products that became immensely popular. The program was made freely available to PDP users. To that point in time, all chess programs developed had a user community of one – the team that developed the program. MacHack could now be played by thousands of people.

Greenblatt's work was a major milestone in the history of computer chess. He pioneered the participation of computers in human tournaments. He invented new ideas that would improve a chess program's performance. He advanced the state of the art in arguably the biggest leap forward in computer chess history.

Mikhail Botvinnik (1968)
Former World Chess Champion

> I forecast an unprecedented period of popularity for the game. When an electronic machine has started playing chess and played it successfully this will be such a momentous event that every schoolboy will want to know about it. In world history, it will perhaps fall not far short in importance of the discovery of fire.
>
> The young will have to study not only computer technique and programming but also chess itself. And then when a hundred times more young people study chess, when many of them devote their lives to it, then we shall have a real chance of getting a new generation of [Mikhail] Tals and [Boris] Spasskys.

While Greenblatt's program generated considerable media and research community interest, little was heard from the Soviet Union. The ITEP program remained hidden behind the Iron Curtain; progress, if any, was a secret. Nevertheless, others took up the challenge. Another Russian chess program appeared in the late 1960s. Not much is known about it, but in 1968 it played a game against the readers of *The Ural Weekly* (*Uralsky Rabochny*) newspaper. Each week the program played a move, and the readers would vote on a response (majority rules). Although the program's play is weak, it is noteworthy that elite grandmaster Lev Polugaevsky annotated the game for the magazine *Chess in the USSR* (*Shakhmanty v USSR*). A selection of his annotations is included below (Polugaevsky 1968).

Readers of *The Ural Worker* newspaper – Russian Program

Nimzowitsch Defense B00
Ural region, USSR, 1968

1.e4 ♘c6

A move suggested by Nimzowitsch. It is not very popular and has almost disappeared from tournaments, but the computer has its own 'theoretical taste,' which does not coincide with the conclusions of contemporary chess theory.

2.d4 d5 3.♘c3 d×e4 4.d5 ♘e5 5.♗f4 ♘g6 6.♗g3 f5?

It seems that a computer also has human weaknesses – it can be just as greedy as a human being.

7.♗b5+ ♗d7 8.♘h3!

8...c6!

A natural move, but since it was made by a computer it deserves an exclamation mark. This move bears witness to the great possibilities of the electronic chess player. Evidently the computer is able to assess the position correctly. Black's Achilles' heel is the square e6 and the computer correctly decides not to allow

the exchange of his white-squared bishop, which is the only piece defending that square.

9.♗c4 ♕b6 10.♕d2 ♕c5

The computer is alert. It avoids the trap prepared by the humans: 10...0-0-0 11.♘a4 and the queen has nowhere to go. The computer also refuses the 'Greek gift' – the pawn on b2: 10...♕×b2 11.♖b1 ♕a3 12.♖×b7 with an overwhelming advantage for White. Who could say after this move that the computer thinks in a primitive way?

11.d×c6 ♗×c6 12.♗e6! ♘h6

What would a chess player have played in this position? He would have chosen the lesser evil: 12.... ♖d8 13.♗f7+ ♔×f7 14.♕×d8 h6, but the computer cannot part with the exchange. We should note however that the computer's combinative ability is not too bad: it saw the piquant variation: 12.... h6 13.0-0-0 ♘f6 14.♗c7 and then 15.♕d8+.

13.0-0-0 ♘e5 14.♘g5 ♘h6g4 15.f3 g6

It has to give up the knight. The fight is over, but the computer (like some chess players) does not like resigning in time.

16.f×g4 ♗g7 17.♗×e5 ♕×e5

This leads to an attractive finish. ... Could the computer have seen the final combination? Perhaps, but even a computer is entitled to count on his opponent's mistakes...

18.♕d8+ ♖×d8 19.♗f7+ 1-0

Such experiments of pitting a chess player against a voting audience had been done before. However, this was a first for computer chess.

Horizon Effect

What happens when a program is told to look ahead a fixed number of moves? Trouble. The horizon effect is the colorful name given by Hans Berliner to the phenomenon of a computer playing an obviously bad move because of its limited search horizon of vision. Consider the following position and assume that Black is only searching three ply ahead.

Black is in check but has a massive material advantage. The correct sequence is 1...♔g8 2.♗×h8 ♔×h8 – but Black loses a queen for a bishop! This is a terrible result, hence the program searches for something better (within the three-ply horizon).

1...c3 The program sees that c3 2.♗×c3+ ♔g8 gives up only a pawn. This is a much better result than the line above. The material advantage is preserved.

2.♗×c3+ d4 Again, a three-ply search only loses a pawn.

3.♗×d4+ e5 And another pawn is lost. Black is still "winning."

4.♗×e5+ ♔g8 5.♗×h8 and White wins.

Having a fixed depth at which to stop searching clearly is wrong. Hence, considerable effort was devoted to identifying when to extend the search. For example, a common heuristic used is to never stop searching in a position where a piece is *en prise*. That would have solved the problem shown above but, of course, more such rules are needed.

In 1957, Herbert Simon predicted that chess programs would be world-championship caliber by 1967. That milestone came and went without much note, except perhaps by Hubert Dreyfus. But a new prediction soon came forward.

David Levy was a young master chess player with an interest in computing science and artificial intelligence. The confidence of youth met up with the over-optimism of research (Levy 2005):

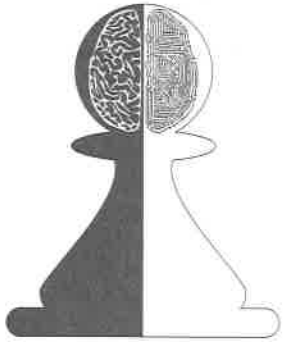
> In August 1968 John [McCarthy] and I started a bet that became a milestone in computer chess history. We were at a cocktail party in Edinburgh during one of the machine intelligence workshops organized by Donald Michie who was founder and head of the first AI university department in Britain. During 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 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.
>
> 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?" 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 that I would lose a match to a computer program within 10 years. Later the bet grew bigger. The following year, [MIT researcher] Seymour Papert and [scientist and

computer chess program developer] Edward Kozdrowicki joined the list of opponents and the final amount at stake when we ended the bet was £1,250. But I had never felt that I was going to be in any trouble.

HAL 9000 (1968)
Computer

I'm sorry Frank, I think you missed it. Queen to bishop three, bishop takes queen, knight takes bishop, mate.

At the end of the 1960s, academic interest in building chess-playing computer programs was high. In part this was a consequence of the media and academic attention that Greenblatt's work attracted. The result was several North American computer chess efforts being launched. This work needed a catalyst to increase the efforts invested in the project, attract sponsors, bring in research grants, and advance artificial intelligence research. All this came together in 1970.



Middlegame

3
2000 (1970-1978)

The publicity generated by MacHack helped increase the level of interest in creating chess-playing programs. Tony Marsland, a graduate student at the University of Washington, enjoyed playing chess and this interest motivated him write a chess program. After graduating, he went to Bell Telephone Labs in New Jersey where he continued tinkering with his program in his spare time. He came up with an idea for helping to popularize computer chess research at the upcoming Association for Computing Machinery (ACM) conference (Marsland 2007):

> … I wrote to Monty Newborn, who was working at Columbia University in Manhattan and was an organizer for the upcoming ACM Fall Joint Computer Conference, suggesting that we provide some kind of a Computer Chess Exhibit. I had in mind a demonstration of computer vs. human play. Instead, Monty came up with a better idea of a computer chess tournament and we met with Keith Gorlen and David Slate (Northwestern University) in a Howard Johnson's cafe on the Garden State Parkway and hammered out a proposal that Monty took to the ACM for their blessing…

And so was born the 1970 ACM Computer Chess Tournament. This event was meant to generate publicity for computer chess, help foster and support research in this area, facilitate the exchange of ideas, and benchmark the progress of developing strong chess-playing programs. In the end, the event was a tremendous success and became an annual event through to 1994. It was soon renamed as the North American Computer Chess Championship. The 1970 event was the start of a 25-year experiment that documented the gradual improvement in the playing abilities of chess computers.

The first tournament attracted six entrants. Of interest in the lineup was Hans Berliner, then a Ph.D. student at Carnegie Mellon University. He was a strong over-the-board chess master, and was the World Correspondence Chess Championship from 1965-1968. In 1956 he won the Eastern States Open, ahead of a promising junior player named Bobby Fischer. Berliner was the first strong chess player to write a chess program – in this case J. Biit (Just Because It Is There) was the first program he had ever written!

MacHack was noticeably absent from the lineup. Greenblatt explains why his program did not participate (Van den Herik and Greenblatt, 1992):

> Basically I was not particularly excited by the idea of computer-vs-computer chess. That plus the fact that I was busy at that time I think are the two reasons. I felt then and I still feel now to a great extent that it is better for the field if anybody can go to the local tournament and play any time when ready. The whole thing, where there is an event once a year, and you come in and play 4 or 5 games, is not a particularly positive situation. But on the other hand I also understand that from the point of view of sponsorship and people's interest and so forth, maybe that helps promote the game and promote computer chess.

Max Euwe (1970)
Former World Chess Champion

The question is not merely whether a computer can be taught to play chess, but whether a computer can replace human perception to any great extent. If it is possible to arrive at an answer using chess as an example, a great contribution will have been made to the understanding of how the mind functions.

The first game to finish achieved one of the tournament's goals immediately – publicity. Programming errors resulted in the Marsland CP quickly succumbing to J. Biit (Marsland 2007):

> The first ACM …Computer Chess Championship took place in New York. Meanwhile I was busy driving across the continent (probably I was in North Dakota when the first round started). However, I had arranged with my local sponsors [to have someone operate the program for me]. I am sure he would have had a happier time had [the Marsland CP] performed better, but at least we recognized the value to the advertising world of a *New York Times* headline like "Computer Loses in King-sized Blunder"! Any mention of computer chess in the [*New York Times*] was better than none, I guess.

The participants quickly realized another goal: the exchange of ideas. David Levy, who started in 1971 to be the guest commentator at the ACM events, observed this first hand (Levy, 2005):

> And one of the things that was very noticeable to me very quickly was the friendly atmosphere at the tournaments, in which the programmers would chat to each other while the games were in progress and between rounds. And they would get ideas from each other. So that after each tournament, the programmers would go away not only with more knowledge about their own programs, but with knowledge about how other people were doing things. And this, in my view, was the main factor in increasing the strength of programs steadily year on year. It was just an acquisition of important knowledge by most of the people in the field. So I think the importance of these tournaments cannot be underestimated in the whole history of the progress of computer chess.

The tournament was won by Chess 3.0, developed by a team of students at Northwestern University. The program's win was decisive, not only by winning all three of its games but, more importantly, the quality of its play was noticeably above that of the other entries. This program, often called Chess x.y to avoid having to remember their numbering scheme, was to dominate the annual computer chess tournament for a decade.

Program	Authors	Score
CHESS 3.0	Larry Atkin, Keith Gorlen, David Slate	3-0
DALY CP	Chris Daly, Ken King	2-1
COKO III	Dennis Cooper, Ed Kozdrowicki	1½-1½
J. BIIT	Hans Berliner	1½-1½
SCHACH	Franklin Ceruti, Rolf Smith	1-2
MARSLAND CP	Tony Marsland	0-3

CHESS 3.0 wins the 1st ACM Chess Championship, 1970.
Left to right: Monroe Newborn, Larry Matsa (ACM President), David Slate, Larry Atkin (CHESS 3.0), and Ben Mittman (Northwestern University). (Monroe Newborn)

In 1968 undergraduate students Larry Atkin and Keith Gorlen wrote a chess program. Physics graduate student and 2050 USCF rated player David Slate heard of this effort and wrote his own program. In 1969, the two teams joined forces with the resulting effort named CHESS 2.0. In 1970 Gorlen left Northwestern University and the CHESS team (although he stayed in touch and occasionally made contributions).

Robert Fischer (1972)
Former World Chess Champion

Up till now they've only had computer scientists developing such programs, and they won't get anywhere until they actually involve some good chess players.

David Bronstein (1973)
Grandmaster

Whatever you might say and whatever I might say, a machine which can play chess with people is one of the most marvelous wonders of our 20th century!

There was nothing unusual about the early versions of the program. Yet it showed complete dominance in winning the first three ACM tournaments. Larry Atkin and David Slate (1977) described CHESS 3.6 as:

> ...the last in a series of evolutionary changes to our original chess program, written in 1968-1969, and it faithfully carried most of the original design deficiencies. CHESS 3.6 was, like the dinosaur, a species about to become extinct. Basically, a Shannon Type B program, it had a depth-first [alpha-beta algorithm], more-or-less fixed depth tree search. A primitive position evaluation function scored the endpoints and also doubled as a plausible move generator earlier in the tree by selecting the "best *n*" moves for further exploration. Rudimentary as they were, CHESS 3.6's evaluation and tree search were just adequate to make "reasonable-looking" moves most of the time and not hang pieces to one- or two-move threats. Apparently this was enough to play low class C chess and, for a while, to beat other programs.

Samuel Reshevsky (1973)
Grandmaster

Until you can engage a grandmaster of high repute, the computer will never get anywhere.

In 1973, faced with the prospect of trying to make incremental improvements to the program's code that had become increasingly messy over the years, Slate and Atkin opted for a complete rewrite and a new tree-searching strategy. This change to the search algorithm, although cosmetically simple and suggested by Claude Shannon almost 25 years previously, had profound ramifications for the future. Their reasoning went as such (Slate and Atkin, 1977):

> ... CHESS 3.6 had a plausible move generator based on its evaluation function. ... At first we were going to implement a similar scheme in CHESS 4.0. However, with only a month or two remaining before the [1973 ACM] tournament, we changed our minds. Although our plausible-move generator sounded plausible enough, and differed not very much from methods employed in several other chess programs, we had built up profound dissatisfaction with it over the years. A suggestion by Peter Frey triggered some thoughts on this matter, and as a result we dumped selective searching in favor of full-width searching, ostensibly a more primitive algorithm.[2] In CHESS 4.5, all legal moves are searched to the same depth. Beyond that depth, only a limited "quiescence" search of captures and some checks is conducted. ...
>
> The principal motivation for switching to full-width searching was a desire for simplicity. Simplicity was important to ease the testing and debugging that had to be crammed into a short period of time. The easiest way to avoid all the complexities of generating plausible moves is to do away with the plausible-move generator. The trouble with plausible-move generators is that they have to be very clever to avoid discarding, at ply 1, good moves whose merit even a meager 5-ply search would discover. This is true for both tactical and strategic moves. Thus a move that appears "quiet" (to a naïve plausible-move generator) at ply 1 may pose a threat at ply 3 and

> *win outright at ply 5. With* CHESS *3.6, and other Shannon Type B programs, whether the right move is played often depends on whether, by sheer accident, that move is inadvertently included in the "best n," n being about 8 or so moves, at the base of the tree. ...*

The notion of considering all moves had, of course, been discussed by researchers going back to Shannon (his Type A approach). Usually this idea was met with derision since it was clearly not how humans approached game-tree searching. Further, the combinatorial explosion of possible scenarios seemed to make the idea impractical. Slate and Atkin were willing to try it. They did not have to wait long to get feedback on the idea (Slate and Atkin, 1977):

> The implementation of full-width searching had immediate beneficial results. At last we had a program whose behavior we could explain simply. When it searched to 5 ply it found everything within that range, including both tactical combinations and positional maneuvers, some of which were obscure and ingenious.
>
> Besides simplicity, a full-width search rewards its creators with "peace of mind." The following daydream (or nightmare) illustrates the psychological hazards associated with the standard "best *n*" tree-search approach.
>
> Imagine yourself at a computer chess tournament. In a complicated position your program has the opportunity to shine by finding the right continuation or to embarrass you by making a blunder. You are reduced to a mere agent of the machine – communicating moves between it and its opponent and reporting the time on request. While anxiously waiting for the machine's decision, you speculate about what move it will make. It is difficult to infer the program's thinking processes. By combining an estimate of the machine's ability with an analysis of the structural features of the position you decide that:
>
> 1. The program will very likely make the right move.
> 2. The program will very likely make the wrong move.
> 3. The machine's move will depend on seemingly irrelevant factors that are difficult to estimate.
>
> In Cases 1 and 2 the suspense is relieved – one has peace of mind. Case 3, however, is hard on the nerves. Often one likes to be surprised by one's program, but not in positions where there is something straightforward to be done. A full-width search sharply reduces the number of incidents of Case 3 by eliminating the "seemingly irrelevant factor" of whether a tactically crucial move at a low ply level happens to lie just within the best-*n* group or outside of it.

From the debugging point of view, things were enormously simplified. If one did an *n*-ply search and did not find the winning combination, all you had to do was play out the move sequence to see if it should be found in *n* ply or less. If so, then this probably indicated a programming problem. As computer chess programmers discovered over and over, sometimes there was bug in their code, but equally likely was their inability to do chess analysis (and count ply)!

Another innovation in the CHESS series of programs was iterative search. For its decision-making process, the program would search to a fixed depth. This depth

limit was set at the start of the search and, for many programs, was the same depth for all searches. However, some searches are easy (there is one obvious best move) and others are more difficult (many promising moves to choose from). The former might result in a 5-ply search that takes a few seconds, while the latter might be many minutes (using 1970s computer hardware). How do you choose a search depth that reflects the amount of effort required?

The idea of iterative search is to keep increasing the search depth until sufficient time has been used. The Northwestern program would do a 1-ply search. Once completed and assuming they still had more time in which to make a move, they would repeat the search but this time to a depth of 2 ply. Again, if the search completes and there is more time available, try doing 3 ply, and so on. Thus the final search depth does not have to be set in advance; the program keeps going until a time limit is reached.

Another important idea used by the Northwestern team was "bit boards." The idea is to use one computer bit for each square on the board – 64 bits of information, or 8 bytes. Bit 0 might represent the square a1; bit 1, a2; bit 3, a3; and so on with bit 63 representing h8. By manipulating the bit board with logic operations – Boolean and, or, not, exclusive-or, and shift – new information about the current position could be efficiently computed. For example, consider generating all the legal moves for the white pawns. If one takes a bit board representing all the white pawn locations (e.g., a2, bit 1, and c2, bit 17), shift the bits to the left one position (effectively adding one to each), and then logically "and"ing it with a bit board of all the empty locations (turning bits off for locations that are occupied), then one has a new bit board showing all the legal one-square-forward moves by white pawns.

Bit boards can be used to quickly compute many properties on the chessboard. Some of the more popular uses of bit boards include move generation (what are the set of legal moves), king safety (determining the squares near the king that are under attack), and evaluating pawn structure (finding doubled, isolated, or backward pawns).

Iterative Deepening

The idea of iterative deepening (ID or iterative search) is to repeat a search over and over again, each time increasing the search depth. This is a counter-intuitive idea since the only thing that matters is the result of the final completed search; the earlier searches represent repeated/wasted effort. So what is the appeal of iterative deepening?

There are four important observations one can make about the importance of iterative deepening:

(1) The early search depths are irrelevant. Assume that adding one to the search depth increases the search time by a factor of 5 – e.g., searching to depth 5 takes 10 seconds, but to depth 6 takes 50. The vast majority of the time is spent in the last iteration. In this example, the time taken by the searches to depths 1, 2, 3, 4, and 5 is dwarfed by the cost of the 6-ply search.

(2) Time control. Given a fixed amount of time, ID allows the program to find the maximum search depth achievable. Most programs use ID to decide when to

stop searching. For example, in the scenario above, should the program search to depth 7 if it has to make a move in at most a total of two minutes? If depth 6 takes 50 seconds and it might take a factor of 5 to reach depth 7, then there is no point in starting the search – it is unlikely to complete in the time required.

(3) Move ordering. The alpha-beta algorithm is most efficient (builds the smallest search trees) when it considers the best move first. In older programs, a plausible move generator would be used to order the moves; more often than naught the best move was not in the #1 position. But ID solves this. The results from the 1-ply search are used to order the moves for the 2-ply search, and so on. In most cases, the best move from an n-ply search is the best move for the n+1-ply search. Thus, even though ID does additional (smaller) searches, the information gleaned from this work usually makes the overall search effort less than had ID not been used!

(4) Results from an earlier search can be used in a later search. Whenever a position is searched, the best move found can be saved in the transposition table. When that position is reached again, possibly on a subsequent iteration (larger search depth), the best move can be retrieved and searched first – it was best previously, so it has a good chance of still being best.

Iterative deepening helps the program manage time and also improve search efficiency. Further it is simple to implement. It is a winner all around!

The name iterative deepening appears to have been first suggested by Jim Gillogly, the author of the TECH chess program, a many-time competitor in the ACM tournaments.

Slate and Atkin (1977) succinctly capture the frustration they felt in developing their series of chess programs in the 1970s: "The lack of programming tools has plagued the whole field of computer chess. With the proper tool one might accomplish in a day a job that had been put off for years." The truth is that 40 years later, their comment is still valid!

The above program enhancements (added at various times throughout the 1970s) and the later move to a fast computer (a Control Data Cyber machine, one of the fastest commercial computers of that era) allowed the CHESS X.Y programs to stay consistently ahead of the rest of the competition at the ACM tournaments::

1970: CHESS 3.0, first (3 wins, 0 draws, 0 losses)

1971: CHESS 3.5, first (3 wins, 0 draws, 0 losses)

1972: CHESS 3.6, first (3 wins, 0 draws, 0 losses)

1973: CHESS 4.0, first (3 wins, 1 draw, 0 losses)

1974: CHESS 4.2, second (3 wins, 0 draws, 1 loss – to RIBBIT)

1975: CHESS 4.4, first (4 wins, 0 draws, 0 losses)

1976: CHESS 4.5, first (4 wins, 0 draws, 0 losses)

1977: CHESS 4.6, first, tied with DUCHESS (3 wins, 1 draw, 0 losses)

1978: CHESS 4.7, second (3 wins, 0 draws, 1 loss – to BELLE)

1979: CHESS 4.9, first (3 wins, 1 draw, 0 losses)

It is remarkable how consistently well their program played over the first 10 North American Computer Chess Championships (1970-1979). Given the small number of games in each event, the closeness of the competition in terms of playing strength, and the presence of programming bugs, their dominance is a testament to Slate's and Atkin's innovative ideas, careful programming, and attention to details.

Killer Heuristic

Pretend in some position White plays the move ♕f5 and it is refuted by Black's fork ♘d6. ♘d6 is said to be the "killer" move for ♕f5. Now assume sometime later in the search, in a different position, White plays ♕f5. The program has not seen this position before and does not know what move is best for Black. The idea behind the killer heuristic is the following reasoning – ♘d6 refuted ♕f5 before, so maybe it will do it again. Hence, the program tries ♘d6 first, if it is legal in the position.

The killer heuristic is just that, a heuristic. It knows that ♘d6 worked before and hopes that it will work again. The heuristic does not take into account any of the surrounding context. For example, in the first instance, ♘d6 may have been a checking move, resulting in a win of the queen. In the second instance, ♘d6 may be a bad move because the square is attacked by a pawn.

Regardless, given a position with no information as to what move to try, some knowledge is better than no knowledge. Slate and Atkin (1977) characterize the idea as "basically an inexpensive attempt, based on superstition, to find a quick refutation move." They did not invent the idea, but they helped popularize it.

The program was also successful in human tournaments. In 1974 CHESS 4.0 played in an event held at Northwestern University. In a field of 50 players, it scored 4½ out of 6 with a performance rating of 1736. By the end of 1975 it had a USCF rating of 1572. In other words, it was not yet clear whether the Northwestern program has stronger than MACHACK.

In 1976, all doubt was removed. CHESS 4.5 played in the Class B (1600-1800 ratings) section of the Paul Masson tournament in California. Peter Frey (1978) recounts that the program:

> ...played against 5 human opponents with USCF ratings between 1693 and 1784. The program had a perfect 5-0 score. Nobody was more surprised at this outcome than the authors, David Slate and Larry Atkin. They have consistently maintained that the program is about C class in strength. One of the human opponents at the Paul Masson tournament remarked that the program was the "strongest 1572 player that he had ever seen."

David Slate (left) and Larry Atkin at the 1975 North American Computer Chess Championship (Monroe Newborn)

Chess 4.5 – Chu, Herbert (1784)

Owen's Defense B00
Paul Masson ACC California (5), 25.07.1976

1.e4 e6 2.d4 b6 3.♘f3 ♗b7 4.♘c3 ♗b4 5.♗d3 ♘f6 6.♗g5 h6 7.♗×f6 ♕×f6 8.0-0 g5?! 9.♕d2?

White just completes the development and misses the very strong 9.♘b5! ♕d8 (9...♘a6 10.c3 ♗e7 11.♘d2 ♕g7 12.♘c4) 10.c3 ♗e7 11.a4 g4 12.♘d2 ♖g8 13.f4 a6 14.♘a3 Shchekachev-Bacrot, Calatrava 2006, in both cases with a strong initiative.

9...♘c6 10.a3?! ♗e7 11.♘b5?! Now this comes too late.

11...♗d8?! After the natural 11...0-0-0, Black is much better as ...g4 is coming.

12.e5?! This does not feel right, but Black is probably still slightly better anyway.

12...♕g7 13.♕e3 a6 14.♘c3 g4 15.♘e1 ♗g5 16.f4

So far Black's strategy has worked well, but now comes the crucial phase.

16...♗e7?! 16...g×f3 17.♕×f3 0-0-0 is even better as 18.♕×f7? runs into 18...♕×f7 19.♖×f7 ♘×d4−+.

17.f5?

White is not ready for this advance yet. After 17.♘e4 d5 18.e×d6 c×d6 19.c3, he is slightly better as Black's king has problems finding a safe harbor.

17...e×f5? Both miss the very strong 17...♕g5!, after which Black is clearly better.

18.♗×f5

18...♘×d4?

This runs into a powerful shot. The normal 18...0-0-0= was called for.

19.♗×d7+! ♔×d7 20.♕×d4+ ♔c8 21.♘d3 ♔b8?

After this slow move, the computer gives Black no chance and controls the game. 21...♖d8 22.♕f4 ♖d7 was the last chance to fight.

22.♕d7 ♕g5?

This just gives up a pawn. 22...g3 23.h3 and 22...♕f8 were more tenacious.

23.♖×f7 ♕e3+?! 23...g3 offers more resistance.

24.♔h1 ♗d8 25.♕×g4 25.♖e1!?+−.

25...♔a7 26.e6 ♗g5?! 27.♖e1 ♖hg8 28.♖×e3 ♗×e3 29.♖g7 ♖af8 30.h4 1-0

With a 1950 performance rating, Chess 4.5 now had a USCF rating of 1822. Why the sudden improvement? As much as the program creators would love to say it was the result of brilliant ideas, thorough testing, and careful programming, the answer in this case was simpler. No doubt all of the above contributed, but the real reason was the move to a Control Data Corporation (CDC) 170 computer. This was one of the fastest commercial computers in the world, and allowed the program to do an unprecedented amount of searching for each move decision. The days of 5-ply searches under tournament conditions were coming to an end. The faster computer and better search algorithms allowed 6-ply in the middlegame and deeper in endgames. Deeper searching clearly enabled stronger play. But what was the value of an extra ply of search? It would be a few years before that question was answered.

The following year Chess 4.5 played in the Minnesota Open Championship. Its impressive score of five wins and one loss included the first tournament victory over a player with a 2000+ rating.

Chess 4.5 – Fenner, Charles (2016)

Sicilian Defense B42
84th Minnesota Open (2), 19.02.1977

1.e4 c5 2.♘f3 e6 3.d4 c×d4 4.♘×d4 a6 5.c4 ♘f6 6.♗d3 ♕c7 7.0-0 ♗c5 8.♘b3 ♗a7 9.♘c3 ♘c6 10.♗g5 ♘e5 11.♗×f6 g×f6 12.♕e2 d6 13.♔h1 ♗d7 14.f4 ♘×d3 15.♕×d3 0-0-0 16.♖ad1 ♗c6?!

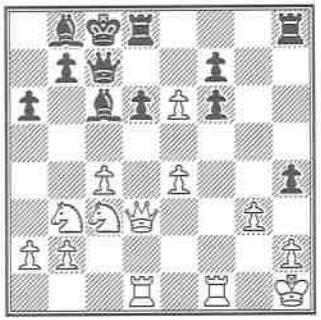
The beginning of a wrong plan. 16...h5 is more in the spirit of the position as 17.♕×d6 ♕×d6 18.♖×d6 h4 19.h3 ♗c6 20.♖×d8+ ♖×d8 gives Black compensation for the pawn.

17.f5 ♗b8?! 18.g3?!

Slightly weakening. 18.♕h3 is more precise.

18...h5 19.f×e6 h4?

19...f×e6 20.♘d4 ♗d7 limits the damage.



20.♖×f6?

Very greedy. 20.♘d5 wins, e.g., 20...♗×d5 21.c×d5 ♗a7 22.♖c1 ♗c5 23.♘d4+–.

20...h×g3?!

This is too ambitious. After 20...f×e6 21.♘d4 (21.♖×e6 is met by 21...d5) 21...♕g7, Black is not worse.

21.♕×g3 Objectively 21.♖×f7! is better, but very messy, e.g., 21...♕b6 22.♖d2 ♖dg8 23.♕d4 and White is for choice.

21...♖dg8?

Based on a miscalculation. After 21...f×e6 22.♖×e6 ♕f7, Black has more than enough compensation for the two pawns.

22.e×f7! The computer shows tactical alertness.

22...♕×f7 22...♖×g3?! 23.f8♕+ ♖×f8? runs into 24.♖×f8+ ♔d7 25.♖f7++–.

23.♖×f7 ♖×g3 24.♘d5 Fenner offers a draw, which is declined.

24...♗e8?

Desperation. 24...♖gh3 is necessary.

25.♘b6+ ♔d8 26.♖×b7 ♗c6 27.♖×b8+ ♔c7 28.♖c8+ ♖×c8 29.h×g3 ♗×e4+ 30.♔g1 ♖h8 31.♘d5+ ♔c6 32.♘a5+ 1-0 A very good game by the machine for those times.

The only loss was to a player rated 2175. With a 2271 performance rating, the chess and computer-chess worlds were stunned to realize that master-rated chess programs were not that far off.

The highlight of 1978 was the Twin Cities Open in Minneapolis. Chess 4.5 blitzed the field, winning all five games. The program's USCF rating climbed past the magical barrier of 2000. With a 2014 rating, Chess 4.5 held the official title of Expert.

And then there was speed (blitz) chess. In March 1977, Chess 4.5 scored 2 points out of 4 in an exhibition blitz match against David Levy, a 2300+ performance rating. This was followed up in September with an exhibition game in London against Grandmaster Michael Stean (2485). User interfaces were not as user friendly then as they are today, so Levy acted as the conduit for transferring the computer's moves from the screen to the chessboard. Stean was given five minutes for the game; the program had five seconds per move (because of the user interface issue) and would forfeit if the game did not end by move 60.

Chess 4.6 – Stean, Michael (2485)

Owen's Defense B00
Blitz game London, 18.09.1977

1.e4 b6 2.d4 ♗b7 3.♘c3 c5?!

Experimental. 3...e6 is the main move.

4.d×c5

A good choice under the circumstances. 4.d5 scores better in human games.

4...b×c5 5.♗e3 d6 6.♗b5+ ♘d7 7.♘f3 e6 8.0-0 a6 9.♗×d7+?

A mistake that reduces White's potential. After 9.♗a4, White's lead in development is very dangerous.

9...♕×d7 10.♕d3 ♘e7 11.♖ad1

Stean: "The damned computer has one of my pawns." But he does find a defense.

11...♖d8 12.♕c4?

This and the next moves show that the computer has no concrete idea of how to play the position. 12.♘d2 ♘c6 13.♗f4 ♗e7 14.♕g3 0-0 15.♘b3 applies much more pressure.

12...♘g6 13.♖fe1? 13.♗×c5? is met by 13...♕c6–+.

13...♗e7 14.♕b3? White should try to halve the bishop pair with 14.♗g5.

14...♕c6 15.♔h1?

An odd move. White should not waste time and continue the initiative with 15.♘d2 but Black is better also in this case.

15...0-0 16.♗g5 ♗a8 16...♗×g5 17.♘×g5 h6 18.♘f3 ♖b8 is more precise.

17.♗×e7 ♘×e7 18.a4 ♖b8 19.♕a2 ♖b4 20.b3 White's queen looks very odd now.

20...f5 21.♘g5 f×e4 22.♘c×e4

22...♖×f2?

Flashy, greedy and wrong. After 22...d5 23.♘g3 ♖f6 24.c3 ♖g4 Black has a strong attack coming up.

23.♖×d6?

Also too greedy. Stopping the attack first with 23.♔g1 is called for.

23...♕×d6?

It is better to sacrifice the exchange or a pawn with 23...♕c7 24.♖dd1 (24.♘×f2 ♕×d6 25.c4 ♘f5 26.♖×e6 ♕f8) 24...♖×e4 25.♘×e4 ♖f4 with compensation in both cases as a result of White's offside queen.

24.♘×d6 ♖×g2?

This is "easily" parried by the machine. 24...h6 25.♘ge4 ♗×e4 26.♘×e4 ♖×e4 27.♕b1 ♖g4 28.♖g1 ♖e4 29.♕d1 ♘f5 is much more active and gives practical drawing chances.

25.♘ge4 ♖g4 25...♖×e4 26.♘×e4 ♖g4 27.♕a3 ♗×e4+ 28.♖×e4 ♖×e4 29.♕×c5 +−

26.c4 ♘f5 27.h3 Stean: "This computer is a genius."

27...♘g3+?! 27...♘×d6 28.h×g4 ♘×e4 offers more resistance.

28.♔h2 Stean: "Help."

28...♖×e4 29.♕f2 h6 29...♖×e1? 30.♕f7+ ♔h8 31.♕f8#

30.♘×e4 ♘×e4 31.♕f3

The computer knows no fear and sees no ghosts.

31...♖b8 32.♖×e4 ♖f8 33.♕g4 ♗×e4 34.♕×e6+ ♔h8 35.♕×e4 ♖f6 36.♕e5 ♖b6 37.♕×c5 ♖×b3 38.♕c8+ ♔h7 39.♕×a6 1-0

This was the first time that a grandmaster had lost to a computer chess program. It was a watershed year for Chess 4.6, defeating GM Robert Hübner and IMs Hans Berliner, Lawrence Day, and Zvonko Vranešić at speed chess.

Another milestone was reached when Chess 4.6 won against United States Champion Walter Browne in a game played in a 44-board simultaneous exhibition. With another grandmaster scalp to the computer's credit, it was now just a matter of time before grandmasters would lose at tournament time controls.

Then there was the idea of creating a man-machine chess-playing team. David Levy recounts of an interesting game played at the 1979 ACM tournament:

> The game against Slate playing in combination with [Chess 4.9], that was the first example of what is now called advanced chess. And it was interesting because it was an innovation at the time to have a chess master playing against a program together with a human chess player. The idea was that both the program and Slate were weaker than me, but the idea was to see whether together they could make a formidable pair. I don't remember the game itself, but I remember that it was quite easy for me to win. But what's interesting is that about 20 years after that, Kasparov came out with the idea of, as it's now called, advanced chess, with a strong grandmaster plus a chess program, against another strong grandmaster with a chess program.

Meanwhile, on the other side of the planet, the Russians had not abandoned their interest in computer chess. Mikhail Botvinnik continued to report that he was working on a program, but little of substance on what he was doing reached the West. However, the ITEP program made famous by the 1966-1967 USSR-USA match was still of interest to Russian scientists. In 1971 Arlazarov and Uskov worked on a successor program, and were joined by an experienced programmer, Mikhail Donskoy (1948-2009). Now working at the Institute of Control Sciences in Moscow, the team felt confident enough in their program that in 1972 they braved exposing it to the public. A two game correspondence match was played between the program and the readers of *Komsomolskaya Pravda*, the youth newspaper for the All-Union Leninist Young Communist League. Moves were played once a week, with the move receiving the most votes from the readers being the one

selected. A journalist at the newspaper suggested a name for the program – KAISSA after Caissa, the goddess of chess. Donskoy (undated) recounts:

> The match took almost a year – from January to November – and ended in victory for [the] people with a score of 1.5 to 0.5. Those who remember the hot summer of 1972 envied the authors of KAISSA who spent a couple of days a week in an air-conditioned machine room – the coolest place in Moscow.

KAISSA – Readers of *Komsomolskaya Pravda*

Sicilian Defense Rossolimo Variation B50
Correspondence match (1), 1972

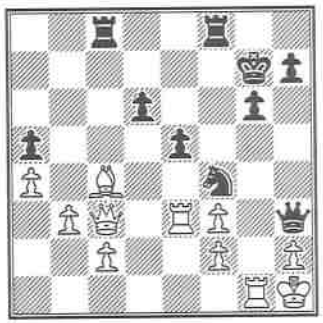
1.e4 c5 2.♘c3 ♘c6 3.♘f3 d6 4.♗b5 ♗d7 5.0-0 g6 6.d4 c×d4 7.♗×c6 d×c3 8.♗×b7 ♖b8 9.♗d5 ♗g7 9...c×b2? runs into 10.♗×b2 ♖×b2 11.♕d4 and both rooks are attacked.

10.b3 ♘f6 11.♗e3 ♕c7 12.♕d4 a5 13.♗c4 0-0 14.♖ae1 ♗c6 15.e5 ♗×f3 16.e×d6 e×d6 17.g×f3 ♘h5 18.♕d3 ♗e5 19.♗d4 ♔g7 19...♗×d4!? 20.♕×d4 ♖fd8 21.a4 d5 22.♗b5 ♘g7 is slightly more precise.

20.♖e3 f6 21.♖fe1 ♘f4 22.♕×c3 ♖bc8 23.a4? White does not have time for this. After 23.♕d2, Black has compensation for the pawn but not more.

23...♕d7 24.♗×e5? This only opens the f-file for Black's attack. 24.♔h1 limits the damage.

24...f×e5 25.♔h1 ♕h3 26.♖g1



26...♘d5? A tactical miscalculation. After 26...♖f5, Black's attack crashes through sooner or later, e.g., 27.♖g3 ♕h6 28.♕a1 ♖h5 29.♕g1 d5 30.♗d3 ♘×d3 31.♖×d3 ♖×c2−+.

27.♕×a5 ♖c5 28.♕a7+ ♖c7 29.♕a5 ♖c5 30.♕a7+ ♖f7?! 31.♕×c5 d×c5 32.♗×d5 ♖f4 33.♖×e5 ♖×f3?! 34.♗×f3 Modern engines want to continue here with 34.♖e7+!? ♔f8 35.♖×h7 ♕×h7 36.♗×f3 when only White can play for a win.
34...♕×f3+ 35.♖g2 ½-½

Readers of *Komsomolskaya Pravda* – KAISSA

Nimzo-Larsen Opening A01
Correspondence match (2), 1972

The second game showed the value of finding a good opening setup. The computer plays too slow and does not manage to solve the problems of the light-squared bishop and gets crushed.

1.b3 e5 2.♗b2 ♘c6 3.c4 f6 4.♘c3 ♗b4 5.♘d5 ♘ge7 6.a3 ♗d6 7.g3 0-0 8.♗g2 ♘g6 9.e3 f5 10.♘e2 ♖e8 11.♕c2 e4 12.d3 e×d3 13.♕×d3 ♖f8 14.f4 ♗e7 15.h4 h6 16.h5 ♘h8 17.e4 d6 18.0-0-0 ♖f7 19.♘×e7+ ♕×e7 20.♘c3 ♗e6 21.♘d5 ♕d7 22.♘e3 f×e4 23.♗×e4 ♘e7 24.♗×b7 ♖b8 25.♗e4 ♘f5 26.♘d5 a5 27.g4 ♘e7 28.♘×e7+ ♖×e7 29.g5 h×g5 30.f5 ♘f7 31.f×e6 ♕×e6 32.♗d5 ♕e3+ 33.♕×e3 ♖×e3 34.♖df1 1-0

Not much else was known about KAISSA until it made its competitive debut in 1974. From discussions with the programmers, some important innovations were in Kaissa:

- Sophisticated time management, including thinking on the opponent's time;
- Extensive use of bit boards, probably predating the CHESS X.Y work;
- Use of the null-move search; and
- Searching using "human-like" reasoning.

Null-move searching was an important idea, whose real strength was not to be realized for another 15 years. The basic idea is for one side to make an illegal move in chess – to pass. Forfeiting the move allows the opponent the opportunity to realize whatever threat they may have in the position. Thus a null-move search is seen as providing a worst-case scenario. The reasoning goes like this: "If I do not move and my opponent then wins a knight, then with my move I better find a way to parry the threat to win my knight." In other words, a null-move search can give the program valuable information about the opponent's threats. Of course, the above reasoning breaks down in a *zugzwang* position.

The human-like reasoning, called the Method of Analogies, was especially intriguing to the research community. The idea was to use analogies to eliminate parts of the search. Consider a position where a sequence of moves allows White to win a knight. Suppose later on in the search a similar position is reached – it is identical except for a pawn having moved one square forward. The question is whether White can play the same sequence of moves to win a knight. The Method of Analogies analyzes all the preconditions needed for a combination to work. If an irrelevant move does not change any of the preconditions, then the original combination should still work.

Clearly there was innovative computer chess research going on in the Soviet Union. Other than the two correspondence games above, the West had no inkling as to how strong the KAISSA program might be. How do you get an unknown program in Russia to compete with the top programs in the rest of the world? David Levy (2005) recounts what happened:

I remember very well in 1973 when we had the ACM tournament in Atlanta, and after the tournament was over, you [Monty Newborn] and Ben Mittman and I were in the bar at the Hyatt Regency, and Ben said, "Gee, guys, this is such great fun. What can we do next?" And I thought for a moment, and I said, "Why don't we have a world championship?" And so we started talking about it, and then I explained to the two of you that how FIDE organizes its world championship every three years, which it did at that time. And we all agreed that would be a lot of fun. And so we started thinking about where we could hold it, and who would sponsor it. I can't remember whether it was you or Ben mentioned that the following year there was going to be an IFIP [International Federation for Information Processing] Congress in Sweden, in Stockholm. And so somehow contact was made with IFIP, and they liked the idea. So they put up the money for sponsoring the championships in Stockholm, and we all went off to Stockholm. That was very interesting, because we not only had most of the top programs from the ACM tournament, we also had some European programs who were able to get there more easily. One of those was KAISSA, the Russian program, about which we knew nothing before it entered for Stockholm.

The first World Computer Chess Championship (1974) had 13 entries, representing the United States (3 programs), Great Britain (3), Canada (2), Soviet Union (1), Austria (1), Norway (1), Switzerland (1), and Hungary (1). KAISSA won the tournament, winning all four games. The pre-tournament favorite, CHESS 4.0, was handed a defeat in round 2 by CHAOS (Ira Ruben, Fred Swartz, Joe Winograd, Victor Berman, William Toikka). CHESS 4.0, CHAOS and RIBBIT tied for second place with 3 points. After the event, a friendly game was played between CHESS 4.0 and KAISSA; the Northwestern program was unable to convert a winning position and drew the game.

Mikhail Donskoy (KAISSA) at the 1974 World Computer Chess Championship (Monroe Newborn)

CHESS 4.0 with its Type A search approach lost a beautiful game to CHAOS, a Type B searcher. In the following position, CHAOS played ♘×e6, a move that is obvious to masters but fraught with danger for a computer program that cannot search deep enough to see the full consequences. Using positional factors to compensate for the lost material, CHAOS found the right move. At the time, ♘×e6 was called the most beautiful move by a computer chess program.

The KAISSA team returned triumphant to Moscow. With them they carried the prize for first place (Donskoy undated):

> I was awarded the "Caissa" gold (in the sense of pure gold) medal of the world champion among chess programs, [and] then deposited [it] in the [Institute of Control Sciences]. In the years of perestroika, its trace was lost in the museums of various chess clubs, where she was transferred without the consent of the authors' team members.

The next year, the Russian computer-chess community achieved another milestone – helping a grandmaster win an adjourned position (Levy 1988):

> Certain standard endgames have been programmed in such a way as to allow perfect or near-perfect play by computers. This work started in the Soviet Union, with a routine to play the endgame of king, queen and g-pawn (or b-pawn) against king and queen. International Grandmaster David Bronstein, who was once a challenger for the World Championship, actually reached this endgame in the Soviet Union in 1975. During the adjournment he telephoned the programmers who looked up their database and told Bronstein how to play. When the game was resumed, Bronstein followed the program's recommendation and eventually won (the opponent deviated from the expected line of play). After the game it was discovered that the program actually had an error in the key variation, overlooking a stalemate possibility, but I understand that this mistake was immediately corrected.

In 1977, KAISSA returned to competition at the Second World Computer Chess Championship in Toronto against a field of 16 competitors. This time there was role reversal between KAISSA and CHESS 4.6. CHESS 4.6 won the event with four wins and KAISSA tied for second place with DUCHESS (Eric Jensen, Tom Truscott, Bruce Wright) after an early loss to DUCHESS. As in 1974, after the event was over there was a friendly game between CHESS 4.6 and KAISSA. This time the American program won. With its world championship title, track record in North American computer chess events, and strong play in human events, CHESS X.Y was clearly the

strongest program of its day.

One position at the world championship caused quite a stir. The following position is from the game DUCHESS versus KAISSA. DUCHESS had just played 34.♕a8+ and to the surprise and amusement of the spectators, KAISSA relied with the incomprehensible 34...♖e8. Clearly this was an enormous blunder, the result of a programming error. Even the chess experts in the audience had a good laugh, including former World Champion Mikhail Botvinnik. It was only after the game that it became known that ♖e8 was forced – the obvious ♔g7 is refuted by 35.♕f8+ leading to mate. This illustrates the strength of the computer's "brute-force approach" to search; by considering all moves, nothing is missed within the search depth. In this case, the human heuristic approach failed to consider the non-obvious move ♕f8.[3]

KAISSA participated in the 1980 World Computer Chess Championship but finished in the middle of the pack. By then the team members had moved on to other projects.

Monroe Newborn (1977)
Computer chess developer and event organizer

Masters used to come to computer chess tournaments to laugh. Now they come to watch. Soon they will come to learn.

Personality

Often a chess player – man or machine – is characterized by their playing style. For example, Mikhail Tal excited audiences with his aggressive attacking approach to the game, while Tigran Petrosian was admired for his subtle positional play. Computer chess programs also have styles, as Shannon noted in his famous paper:

> It is interesting that the "style" of play of the machine can be changed very easily by altering some of the coefficients and numerical factors involved in the evaluation function and the other programs. By placing high values on positional weaknesses, etc., a positional-type player results. By more intensive examination of forced variations it becomes a combination player. Furthermore, the strength of the play can be easily adjusted by changing the depth of calculation and by omitting or adding terms to the evaluation function.

> Thus by changing as little as a single number in a program, one could transform a Tal into a Petrosian, or a *patzer* into a grandmaster. In other words, computer programs have the advantage of being able to take on multiple personalities – and without needing psychiatric help.

During these exciting days of computer chess interest, little was heard from Richard Greenblatt. MACHACK was available on the PDP computers, and was still arguably the most widely played chess program in the world. However, Greenblatt had moved on to other projects, most notably designing and building a special-purpose computer to run programs efficiently that were written in the Lisp programming language (a John McCarthy creation). Eventually this work would lead to Greenblatt creating a company to commercialize this work.

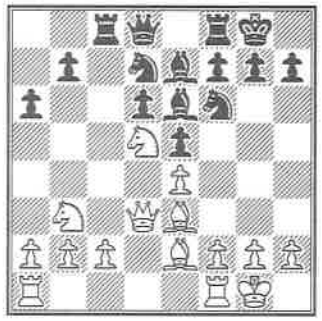
MACHACK continued to attract attention. In 1977, it was not the strongest program but, in many circles, was better known than CHESS X.Y. This reputation led to an unexpected encounter. In 1977, former World Champion Bobby Fischer (1943-2008) showed up at MIT wanting to play MACHACK. Since defeating Boris Spassky for the world title in 1972, Fischer had played no public games. In 1975 he refused to defend his title against Anatoly Karpov. And then, surprise, he wanted to play games against a computer.

Not much is known about the circumstances under which the games were played. Greenblatt was not present. No pictures were taken. The program's logs of the game have not been published. Fischer submitted the game scores to a fledgling publication, the *Computer Chess Newsletter*, without comments.

MACHACK – Fischer, Robert

Sicilian Defense B92
Exhibition match (1), MIT, 1977

1.e4 c5 2.♘f3 d6 3.d4 c×d4 4.♘×d4 ♘f6 5.♘c3 a6 6.♗e2 e5 7.♘b3 ♗e7 8.♗e3 0-0 9.♕d3 ♗e6 10.0-0 ♘bd7 11.♘d5 ♖c8



12.♘×e7+?

A big mistake probably caused by overestimating the pair of bishops. But now Black's dynamics get out of control. 12.c4 is the main move to keep stability, which

also scores quite good for White.

12...♕×e7 13.f3?!

13.♘d2 d5 14.e×d5 ♘×d5 15.c4 ♘×e3 16.♕×e3 is not as weakening.

13...d5 14.♘d2 ♕b4 14...d4!?

15.♘b3? Black initiative gets very dangerous now.

15.♕b3, to trade queens, eases the defensive task, e.g., 15...♕×b3 16.c×b3 d4 17.♗g5 ♖c2 18.♖ab1 h6 19.♗d3 ♖c6 20.♗×f6 g×f6 21.f4 and White is only slightly worse.

15...d×e4 16.♕d1?! ♘d5

17.♗a7?

A typical computer mistake in those days. The bishop will either not see the light of day again or White will have to invest more material to free it. 17.c3 ♕e7 18.♕d2 limits the damage.

17...b6 18.c3 ♕e7 19.f×e4 ♘e3 20.♕d3 ♘×f1 21.♕×a6 21.♖×f1 ♖c7 22.♕×a6 ♖a8−+

21...♘e3 22.♗×b6 ♕g5 23.g3 ♖a8 24.♗a7 h5 25.♕b7 h4 26.♔f2 h×g3+ 27.h×g3 f5 28.e×f5 ♖×f5+ 29.♔e1 ♖af8 30.♔d2 ♘c4+ 31.♔c2 ♕g6 32.♕e4 ♘d6 33.♕c6 ♖f2+ 34.♔d1 ♗g4 35.♗×f2 ♕d3+ 36.♔c1 ♗×e2 37.♘d2 ♖×f2 38.♕×d7 ♖f1+ 39.♘×f1 ♕d1# 0-1

One can only imagine what went through Fischer's mind as he saw the numerous weak moves played by the program. Two more games were played that day, each an easy win for the former World Champion.

In 1978, MacHack played a correspondence game (one move per week) against the readers of a German computer magazine, *Computerwoche* (Computer Week). The game was even until MacHack self-destructed, the result of not searching deep enough.

CW-Leser – MacHack

Petroff Defense C42
Schach dem Computer, Computerwoche, 1978

1.e4 e5 2.♘f3 ♘f6 3.♘×e5 d6 4.♘f3 ♘×e4 5.d4 d5 6.♗d3 ♗e7 7.h3 0-0 8.0-0 ♘c6 9.♘c3 ♘×c3 10.b×c3 ♗e6 11.♗f4 ♖e8 12.♖e1 ♕d7 13.♖b1 ♖ab8 14.♘g5 ♗f5 15.♕h5 ♗×g5 16.♕×g5 ♖×e1+ 17.♖×e1 ♗e6 18.♕g3 ♖c8 19.♗h6 g6 20.♕h4 f5 21.♕f6 ♘×d4 22.c×d4 c6 1-0

Greenblatt realized early on that a faster computer meant deeper searches, and deeper searches meant stronger play. He knew what he could achieve on a commercially available computer, but he wondered what might be possible on a computer that was specially designed for computer chess. In the late 1970s MIT professor Edward Fredkin and Greenblatt worked on developing computer hardware that was specifically designed to facilitate the needs of a chess-playing program. Their system, CHEOPS (CHEss-Orientated Processing System), was in some sense a throwback to the El Ajedrecista approach. The hardware included an 8x8 board, allowing for fast parallel generation of legal moves. Unfortunately the machine could only be as fast as its slowest component – the position evaluation function – that was done largely in software. This went against Amdahl's Law, a well-known computing limit to performance. Assume that a chess search takes 100 seconds, and that 60% of the time is spent doing position evaluation. If you reduce the 40% component to zero – say by building a special-purpose computer – you still are limited by the 60% that is not any faster. Your 100 seconds becomes 60, 1.67 times faster, but no better. And so it was with CHEOPS. Although a working prototype of the machine was completed, it did not play many games before the project was abandoned.

Meanwhile, 1978 was fast approaching, the deadline for David Levy's bet. It was pretty clear that he would win – none of the programs were playing close to 2000 ELO in 1977. Levy dutifully began to play two-game matches against the leading computer program "threats." First up was Chess 4.5 in April 1977. Levy easily won the first game thereby clinching at least a draw in the match; the second game was not played. Next up was Kaissa in December 1977. An easy win in game one negated the need for the second game.

Levy, David (2320) – Kaissa

English Opening A20
Montreal, 17.12.1977

1.d3 ♘c6 2.c4 e5 3.g3 ♗c5 4.♗g2 ♘f6 5.♘c3 0-0 6.e3 ♕e7 7.♘ge2 ♗b6 8.0-0 d6 9.a3 ♗g4 10.b4 ♕e6 11.♘d5 ♗×e2 12.♕×e2 ♘g4 13.♗d2 ♖ab1 14.a4 a6 15.a5 ♗a7 16.b5 a×b5 17.c×b5 e4 18.b×c6 ♕×d5 19.♕×g4 ♕×d3 20.♖fd1 f5 21.♕g5 b×c6 22.♗f1 ♕b3 23.♖dc1 h6 24.♕g6 ♕b7 25.a6 ♕c8 26.♖c3 ♖f6 27.♕h5 ♕d7 28.♖ac1 ♗c5 29.♖c1×c5 d×c5 30.♖×f6 g×f6 31.♕g6+ ♕g7 32.♕×f5 (adjudicated 1-0)

Then it was MacHack's turn in August 1978, this time using CHEOPS. Same result: a Levy win in game one and a won match.

MacHack/CHEOPS – Levy, David (2320)

Sicilian Dragon B71

Cambridge, Massachusetts, 23.08.1978

1.e4 c5 2.♘f3 d6 3.d4 c×d4 4.♘×d4 ♘f6 5.♘c3 g6 6.f4 ♗g7 7.e5 ♘h5 8.♗b5+ ♗d7 9.e6 f×e6 10.♘×e6 ♗×c3+ 11.b×c3 ♕c8 The end of MacHack's book. **12.♕d4 ♘f6 13.♕c4 ♘c6 14.♘d4 ♘×d4 15.c×d4 ♕×c4 16.♗×c4 ♗f5 17.♗b5+ ♔f7 18.♗c4+ d5 19.♗d3 ♖fc1 20.0-0 ♖c7 21.♖b1 ♖ac8 22.♗e3 ♘e4 23.♖f3 ♘d6 24.♖b2 b6 25.a4 ♗×d3 26.c×d3 ♖c3 27.♖h3 h5 28.♗d2 ♖c2 29.♖×c2 ♖×c2 30.♗e1 ♘f5** (0-1 in 43 moves)

Immediately after playing MacHack, Levy went to Toronto for one final match. The opponent was Chess 4.7, a stronger opponent than Chess 4.5 that he played the year before. The faster computer that it used was already paying dividends in human tournaments, and it was realistic to expect the program to be playing at a 2000 USCF level. Still, Levy was rated 2320 ELO, a large gap. The apparent difference in strength had to be tempered by Levy's lack of play in the past few years. Like most strong chess players, he had quickly discovered the life of a chess professional was financially challenging, and he had moved on to other pursuits (including a prolific career writing chess books).

On August 26, 1978, the 6-game match between David Levy and Chess 4.7 began. At stake was pride and the value of the infamous bet. Levy recounts the playing conditions (Levy 2005):

> The match was played in a soundproof glass booth at the Canadian National Exhibition, which is a big exhibition held every year in Toronto. And I had to wear a tuxedo, which is not normal for me. In the 19th century, grandmasters used to wear tuxedos when they played important tournaments, and the early part of the 20th century, so it was sort of nice. And I was playing against the Slate and Atkin program, Chess 4.7, running on a CDC Cyber computer, a very powerful computer located in Minneapolis. When I turned up to play the first game, and sat down, I was expecting David Slate or Larry Atkin to be sitting opposite me making the moves. Instead of which, they wheeled in this really attractive young lady. They clearly decided that they were going to distract me. And she sat there smiling at me the whole time. It was really quite difficult. I had to sort of do what I do in human tournaments, and put my hands like this [shielding his eyes] and look down at the board when I was thinking. And then between moves when I was relaxing, I was sitting there, and she just sat there smiling at me. It wasn't the easiest circumstance under which to play.

Clearly the Chess 4.7 team had adopted an interesting psychological strategy. But Levy also had prepared a strategy, one that he felt could exploit the known limitations of his opponent:

> But I worked out my strategy beforehand, and I developed this sort of anti-computer strategy. And in those days the strategy was very successful, because programs could see a certain distance ahead, but they couldn't see very far ahead. So what you had to be careful of is, you had to be careful of very short-term tactical tricks. You had to just check that there wasn't some sequence of three or five, or maybe even seven moves, that the program could win material, or do something really unpleasant to you.

> And once you got past that, if you could accomplish that safely, then you just have to develop a long-term strategy for the game. And so I developed a very sort of super long-term strategy. I made moves that appeared to have no point at the time, but they were moves which I knew if the game developed as I expected it would, would have a point much later on. But so far into the future that the program couldn't understand it. And the result of that was that the program basically had no idea what was going on. It was just playing the current position with no regard for long-term strategy. Every now and again, it would make a move that created a slight weakness in its position, or move the piece from one part of the board away to where it wasn't defending some area that I wanted to go into much later.

This idea of devising an anti-computer strategy was quite popular for several decades. Even today, with super-human chess-playing programs, work continues on devising ways to exploit perceived weaknesses in the machine's play. All is fair in love and war.

If Levy expected an easy match, he quickly had a rude awakening:

> Well, the first game was almost a disaster. Because I did it [making moves that appeared to have no point at the time] too much, and I was in serious trouble, and I very nearly lost the first game. But I managed to draw.

Levy, David (2320) – Chess 4.7

Reti Opening A07

Toronto match (1), 26.08.1978

1.g3 d5 2.♗g2 e5 3.d3 ♘f6 4.♘f3 ♘c6 5.0-0 ♗d7?!

Very strange for the human eye.

6.b3?!

6.d4 e4 7.♘e5 ♗d6 8.♘×d7 ♕×d7 9.c4 d×c4 10.♘c3 ♕f5 11.♕a4 is more in the spirit of the position.

6...♗c5 7.♗b2 ♕e7 8.a3?!

This allows Black to take the initiative.

8...e4! 9.♘e1 0-0 10.d4 ♗d6

11.e3?

This slow move weakens the light squares. 11.c4 is called for.

11...♘g4 12.h3?

Inviting the following strong strike. But good advice is hard to give.

12...♘×e3!

A correct sacrifice.

13.f×e3 ♕g5 14.g4?

14.♘f3 ♕×e3+ 15.♔h2 e×f3 16.♖×f3 ♕h6 17.♘c3 limits the damage, but Black is of course on top after 17...♗g4.

14...♕×e3+

14...f5!?.

15.♖f2

15...♗g3?!

This should also win in the long run. But the direct attack, 15...f5, pays much higher dividends, e.g., 16.♗c1 ♕g3 17.g5 f4 18.♘d2 ♗×h3 19.♘f1 ♕×g5 20.♘h2 ♕g3−+.

16.♕e2 ♕×f2+ 17.♕×f2 ♗×f2+ 18.♔×f2 f5!

Opening inroads against White's king.

19.g×f5

19.g5 f4 20.♘c3 ♖f5 is also very grim.

19...♘e7 20.c4 ♖×f5+ 21.♔g1 c6 22.♘c3 ♖h5 23.♔h2 ♖f8 24.♘d1 ♘g6 25.♖c1 ♗×h3 26.♗×h3 ♖f1 27.♘g2

27.♘e3 ♖f2+ 28.♔g1 ♖×b2−+.

27...♖f3 28.c×d5 ♖h×h3+ 29.♔g1 c×d5 30.♖c8+ ♘f8

30...♖f8!?, to try to exchange White's active rook, is also very strong.

31.♗c3 ♖d3

31...♔f7!? 32.♖c7+ ♔f6 33.♖×b7 ♘e6 was easier.

32.♘de3 ♖h×e3 33.♘×e3 ♖×e3 34.♗b4 ♖f3 35.♖d8 h6 36.♖×d5

36...♖×b3?!

This greedy capture has no priority. The knight should be activated by 36...♘e6 37.♖d7 ♖×b3 38.d5 ♘d4 39.d6 e3, which wins relatively easily. In the following endgame phase, CHESS 4.7 underestimates the importance of activity in general. Modern programs would doubtlessly win it but in the game Levy even gets a chance to win it.

37.♖d8 ♖f3 38.♖a8 g5?! 39.d5?!

39...h5?!

This makes the technical task much more difficult by giving up winning potential which should be preserved by 39...a6!? 40.♖e8 b6 41.♖×e4 a5 42.♗d6 ♘g6−+.

40.d6

40.♖×a7 b6 41.♖e7 ♘g6 42.♖×e4 ♔f7 should also win in the long run.

40...♔g7 41.♖×a7 ♖f7 42.♖a5 ♔f6 43.♗c3+ ♔g6 44.♖e5 ♖f3 45.♗b4

45...♖f4?

Black's rook should help its own pawns with 45...♖e3, with excellent winning chances.

46.♖e7 ♖f7 47.♖×e4

47...♖d7

47...♘d7 is more logical from a human point of view, as the knight is a better blockader than the rook. But White's activity most probably saves him here as well.

48.♖e7 h4 49.♔g2 g4 50.♔h2 b6 51.♔g2 ♖d8 52.a4 ♘d7 53.a5 ♘f6 54.a×b6 ♘d5 55.b7

55...♘×e7?

This greedy capture is probably a result of the horizon effect as the *zwischenschach* 55...h3+ draws, e.g., 56.♔g3 ♘×e7 57.d×e7 ♖h8 58.♗c3 ♖g8 59.♗a5 ♖h8 60.♗d8 h2 61.b8♕ h1♕ 62.♕d6+=.

56.d×e7 ♖h8

57.♗d6?

Levy rushes to win the rook but misses winning the game with 57.♗c3 ♖g8 (57... h3+ 58.♔h1 +−) 58.♗a5 h3+ 59.♔h1 ♖b8 60.♗c7 ♖e8 61.♗d8 +− .

57...♔f6 58.b8♕ ♖×b8 59.♗×b8 ♔×e7 60.♗f4 ♔f6 61.♗d2 ♔g6 62.♗e1 ♔g5 63.♗f2 ♔h5 64.♗e1 ½-½

Despite the close call, Levy stuck to his strategy and it paid off – he won games two and three:

> But pretty soon in the match, my strategy worked, and I was winning game after game very comfortably. I remember in particular one of the games where I played an English opening, but I played it as though I was playing a Sicilian defense but with white. ... I knew exactly what the long-term strategy was. The long-term strategy in the Sicilian is that you play to reach an endgame; an endgame with a particular kind of formation, which if you can get there in the Sicilian defense, you stand pretty well. So with the extra move, you would stand even better. That was my long-term goal, and I pushed the program off the board.

Levy, David (2320) – Chess 4.7

English Opening A00

Toronto match (3), 28.08.1978

1.c4 ♘f6 2.a3 ♘c6 3.♘c3 d5 4.c×d5 ♘×d5 5.d3 ♘×c3 6.b×c3 e5 7.g3 ♗e7 8.♗g2 ♕d6?! 9.♘f3 ♗e6 10.0-0 0-0 11.♕a4 ♕c5?! 12.♗d2 b5?!

Bad from a positional point of view. It just weakens too many squares.

13.♕c2 f6 14.♖fb1 ♖ad8?!

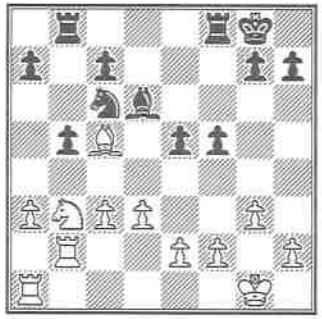
15.♕b2?!

Levy misses or intentionally avoids the tactical possibility 15.♗e3 ♘d4 16.♕d1 ♘×f3+ 17.♗×f3 ♕×c3 18.♖×b5 c6 19.♖bb1 ±.

15...♖b8 16.♗e3 ♕d6 17.♘d2 ♗d5 18.♗×d5+ ♕×d5 19.♕b3!?

Levy knows that the endgame is a weakness of the program and heads for it.

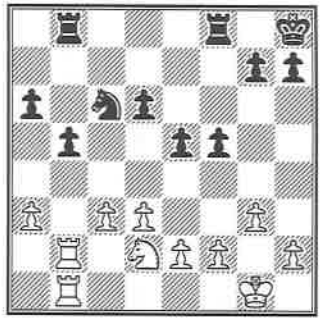
19...♕×b3 20.♘×b3 f5 21.♗c5 ♗d6 22.♖b2



22...♔h8?

The wrong direction, as an endgame is not a middlegame. 22...♔f7 is called for.

23.♖ab1 a6 24.♗×d6 c×d6 25.♘d2



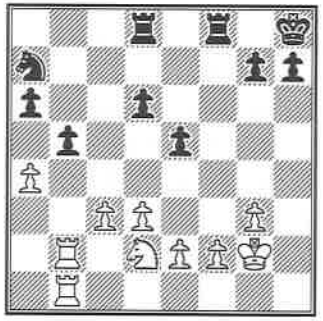
In the early days of computer chess, the endgame was a big problem for the machines because of the importance of long-term planning and thinking in patterns. Chess 4.7 plays the following ending badly.

25...f4?

This just weakens the e4-square and wastes precious time. From a human point of view, it is clear that 25...♔g8 should be played.

26.♔g2 f×g3?! 27.h×g3 ♖bd8?! 28.a4! ♘a7?!

28...b×a4! 29.♖b6 ♘e7 30.♘e4 ±



29.♘e4?

Opening the queenside directly with 29.c4 makes better use of White's better mobilization, e.g., 29...b×a4 (29...b×c4 30.♘×c4 ♘c8 31.♖b8 ♖fe8 32.♖a8 d5 33.♘e3 +−) 30.♖b6 h6 31.♖×a6 ♖f7 32.♖bb6 +−.

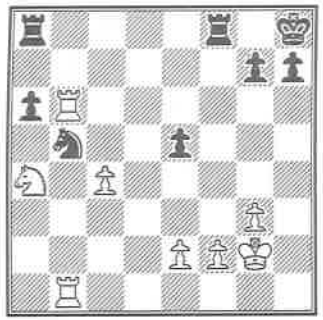
29...b×a4?!

29...d5 30.♘c5 ♖f6 31.♖a1 ♔g8 32.a×b5 ♘×b5 33.c4 d×c4 34.d×c4 ♘c7 is better but White remains for choice of course.

30.♖b6 d5 31.♘c5 ♘b5 32.♘×a4 ♖a8?

Passive. 32...♖c8 33.♖×a6 ♘×c3 34.♘×c3 ♖×c3 35.♖e6 e4 36.d×e4 d×e4 37.♖×e4 ♖c7 offers much better practical chances.

33.c4 d×c4 34.d×c4



34...♘d4?

CHESS 4.7 underestimates the danger posed by White's passed c-pawn. 34...♘c7 is necessary.

35.e3 ♘f3?! 36.c5 ♘g5 37.c6 ♘e4?!

37...♘e6 38.♖c1 ♖f7 39.♖b7 ♔g8 40.♘b6+–

38.c7 ♖×f2+ 39.♔g1 ♖ff8 40.♖b8 h5 41.♖×a8 ♖×a8 42.♖b8+ 1-0

With 2½ points, he needed only a draw in the remaining three games to win his bet. Things had gone so well for him in the previous two games that he became complacent and, well, that is when bad things usually happen.

> … I was feeling pretty confident. And then I decided to take a chance. I'd been very successful with my strategy. I felt very confident that I could use that strategy probably to win every game in the rest of the match if I needed to. So I decided to take the chance and see what would happen if I played very sharp tactical chess. I played a really unsound opening, and the position got very sharp, and the program just killed me.

CHESS 4.7 – Levy, David (2320)

Latvian Gambit C40
Toronto match (4), 29.08.1978

1.e4 e5 2.♘f3 f5 3.e×f5 e4 4.♘e5 ♘f6 5.♘g4

A rare sideline but maybe not bad. 5.♗e2 is the main line.

5...d5 6.♘×f6+ ♕×f6 7.♕h5+ ♕f7 8.♕×f7+ ♔×f7 9.♘c3 c6 10.d3 e×d3 11.♗×d3 ♘d7 12.♗f4 ♘c5 13.g4 ♘×d3+ 14.c×d3 ♗c5

15.0-0?

This castles into an attack. 15.h3 h5 16.♔e2 is a better location for the king in this endgame.

15...h5 16.♘a4?

Misplacing the knight. 16.♗e3 ♗d6 17.h3 is the lesser evil.

16...♗d4 17.♗e3 ♗e5 18.d4 ♗d6 19.h3 b6?

Too slow. After the direct 19...g6, Black is clearly better because of his bishops and his attack.

20.♖fe1?

It is better to try to escape from the danger zone with 20.♔g2, but king safety was a difficult issue for the computers in those days.

20...♗d7 21.♘c3?!

21.♔g2 is still preferable.

21...h×g4 22.h×g4 ♖h4 23.f3 ♖ah8 24.♔f1?! ♗g3?

24...♖h3 hits the Achilles' heel f3 directly and is much stronger. Black has a winning attack.

25.♖e2?

CHESS 4.7 underestimates Black's coming initiative. It had to exchange attacking potential with 25.♗f2 ♗×f2 26.♔×f2 ♖h2+ 27.♔g3 ♖×b2 28.♖e2 and White is not worse.

25...♗c8 26.♔g2 ♗d6?

The wrong order of moves. After 26...♖h3 27.♗g1, the bishop can retreat with 27...♗f4 which gives Black a strong initiative.

27.♗g1 ♖h3

28.♖ae1?

Probably a more or less automatic choice for CHESS 4.7 as rooks belong on open files and the a1-rook was undeveloped. But the exchange sacrifice 28.♖e5!! is much stronger. But such sacrifices posed large problems for the programs for a very long time. But nowadays the programmers seem to have found ways to deal with this problem as the modern programs find it very quickly.

28...♖g3+ 29.♔f2 ♖hh3 30.♖e3

30...♗a6?

Levy miscalculates. After the direct 30...♗f4! 31.♖e7+ ♔f6, White faces a very unpleasant defensive task.

31.♘e2 ♗×e2 32.♖1×e2 c5!

Now it is too late for 32...♗f4? in view of 33.♖e7+ ♔g8 34.♖e8+ ♔h7 35.♖2e7 ♗g5 36.♖×a7 ♗f6 37.♖aa8 g6 38.f×g6+ ♔×g6 39.♖e6 ♖×f3+ 40.♔e2 ♖fg3 41.♖f8±.

33.f4! ♖×e3

33...♗×f4? runs into 34.♖×g3 ♖×g3 35.♗h2 ♖×g4 36.♔f3+−.

34.♖×e3

34...♖h4!

A very difficult decision as 34...♖×e3 35.♔×e3 b5! also gives practical drawing chances. Against a computer, it is probably preferable as long-term planning plays a large role in the resulting bishop ending. With the rooks, the computer has more short-term goals.

35.♔g3 ♖h1 36.♗f2 ♖d1 37.♖a3 c×d4 38.♖×a7+

38...♔f8?

38...♔e8 was the last real chance to fight as ♖d7 must be stopped.

39.♖d7 ♖d3+ 40.♔g2 ♗c5

40...♗×f4 41.♖×d5 ♗e3 42.♗×e3 ♖×e3 43.♔f2 ♖e4 (43...♖h3 44.♖b5+−) 44.♔f3 ♖e3+ 45.♔f4 ♖e2 46.♖b5 d3 47.♖×b6 d2 48.♖d6+−.

41.♖×d5 ♖d2 42.b4 ♗×b4 43.♖d8+?!

The direct 43.♖×d4 is better technique and transposes to the game.

43...♔f7 44.♖d7+

44...♔f8?

44...♗e7 45.♔f3 ♔e8 46.♖×d4 ♖×a2 47.♖d5 should be winning for White but it offers much more resistance than the game continuation.

45.♖×d4 ♖b2 46.♔f3 ♗c5?! 47.♖d8+ ♔e7

47...♔f7 48.♗×c5 b×c5 49.g5 ♖×a2 50.g6+ ♔e7 51.♖g8 ♔f6 52.♖f8+ +−.

48.♗h4+ ♔f7 49.g5 g6 50.♖d7+ ♔f8 51.f×g6 ♖×a2 52.f5 ♖a3+ 53.♔g4 ♖a4+ 54.♔h5 ♖d4 55.♖c7 ♗e7 1-0

This was the first time that an International Master had lost to a computer program under tournament conditions. Now the match was much more interesting to the chess world and to the media.

> So after that game I was still leading 2½ to 1½. So we played four of the games, and there were two more games to come. So then I said, "Okay, David, this is time to take it seriously." So I sort of buckled down, and in the fifth game I pushed it off the board again with my long-term strategy and the match was over, because I'd scored 3½ points.

The match was over; Levy had an unassailable lead. He later reflected on his opponent's ability:

> … How strong was the program? It was a lot weaker than me. I would say it was at least 200 points weaker than me. It was good enough that if I'd made a stupid mistake it would've beaten me. Or playing a very risky strategy, as I did in game four, it could beat me. But as long as I was careful, it had no chance.

Levy's historic 1968 bet had comes to an end: man triumphed over machine (Levy and Newborn 1980):

> Thus ended an era in the annals of computer chess. I had proved that my 1968 assessment had been correct, but on the other hand my opponent in this match was very, very much stronger than I had believed possible when I started the bet.
>
> Then there was the matter of collecting his money:
>
> When sending me his cheque for £250 Professor John McCarthy expressed a sentiment [with] which I concurred – he said that had I lost to a brute-force program he would not have felt that the science of Artificial Intelligence was responsible for my defeat. McCarthy, Michie and Papert all paid promptly and with good sportsmanship, just as I would have done had I lost the bet. Only Edward Kozdrowicki did not.

Having won the match, albeit with a closer score than he anticipated in 1968, Levy remained confident. Perhaps another bet?