S|S|S

Social Studies of Science
42(1) 5–30

DOI: 10.1177/0306312711424596
sss.sagepub.com
SAGE

Is chess the drosophila of artificial intelligence? A social history of an algorithm

Nathan Ensmenger
School of Information, University of Texas at Austin, Austin, TX, USA

Abstract

Since the mid 1960s, researchers in computer science have famously referred to chess as the 'drosophila' of artificial intelligence (AI). What they seem to mean by this is that chess, like the common fruit fly, is an accessible, familiar, and relatively simple experimental technology that nonetheless can be used productively to produce valid knowledge about other, more complex systems. But for historians of science and technology, the analogy between chess and drosophila assumes a larger significance. As Robert Kohler has ably described, the decision to adopt drosophila as the organism of choice for genetics research had far-reaching implications for the development of 20th century biology. In a similar manner, the decision to focus on chess as the measure of both human and computer intelligence had important and unintended consequences for AI research. This paper explores the emergence of chess as an experimental technology, its significance in the developing research practices of the AI community, and the unique ways in which the decision to focus on chess shaped the program of AI research in the decade of the 1970s. More broadly, it attempts to open up the virtual black box of computer software – and of computer games in particular – to the scrutiny of historical and sociological analysis.

Keywords

artificial intelligence, computing, drosophila, experimental technology

In 1965, the Russian mathematician Alexander Kronrod, when asked to justify the expensive computer time he was using to play correspondence chess at the Soviet Institute of Theoretical and Experimental Physics, gave an explanation both prescient and prophetic: it was essential that he, as a premier researcher in the burgeoning new discipline of artificial intelligence (AI), be allowed to devote computer time to chess because 'chess was the drosophila of

Corresponding author:
Nathan Ensmenger, School of Information, University of Texas at Austin, 1616 Guadalupe Street, Suite 2.202, Austin, TX 78701, USA.
Email: nathan@ischool.utexas.edu

artificial intelligence'.[1] What exactly Kronrod meant by this dramatic pronouncement is not entirely clear. In 1965 there was hardly much of a field of AI – the term itself had been invented less than a decade earlier – and there were few computers available at this time capable of playing anything resembling real chess. The M-20 computer at Kronrod's institute was one of the few that could. But Kronrod's assertion that chess was, indeed, the drosophila of AI quickly became part of the foundational lore of the discipline. The analogy was first made in print by the Nobel-prize winning economist Herbert Simon in 1973, and by the end of the decade the metaphor was appearing consistently in the literature. More recently, the 'chess as drosophila' metaphor has been extended even further to encompass both AI and cognitive science (Rasskin-Gutman, 2009; Simon and Chase, 1973).[2]

Regardless of how true Kronrod's grand claim about the centrality of chess to AI research might have been in 1965, within a few decades it had become undeniable reality. By all of the measures of contemporary scientific practice, computer chess has proven to be an enormously productive experimental technology. Hundreds of academic papers have been written about computer chess, thousands of working chess programs have been developed, and millions of computer chess matches have been played. It is a rare discussion of AI, whether historical, philosophical, or technical, that does not eventually come around to chess-playing computers (Collins, 2010; Hofstadter, 2005; Searle, 1999). Chess figures prominently in the iconography of the discipline, as a quick tour of the book jackets of its major literature will readily testify (Ekbia, 2008; Nilsson, 1998; Russell and Norvig, 2009). The 1997 victory of the IBM Deep Blue computer over Garry Kasparov continues to be celebrated as one of the pivotal moments in the history of modern computing. In the same way that *Drosophila melanogaster* dominates the history (and, to a lesser degree, the practice) of the genetic sciences, chess dominates AI (Bramer, 1978; Coles, 1994; Franchi, 2005; Franchi et al., 2005; McCarthy, 1997; Rasskin-Gutman, 2009; Robinson, 1979; Ross, 2006; Schaeffer and Donskoy, 1989; Wells and Reed, 2005).

But what exactly does it mean to suggest that chess is the drosophila of AI? The specific meaning of the analogy has never been more than superficially elaborated. What most practitioners seem to mean by claiming chess as the drosophila of AI is simply that computer chess, like drosophila, represented a relatively simple system that nevertheless could be used to explore larger, more complex phenomena. Herbert Simon, for example, described chess as a standardized experimental 'test-bed' that could be used to explore various hypotheses (Simon et al., 1992). In this reductionist interpretation of the history of the genetic sciences, *Drosophila melanogaster*, like Mendel's peas or Darwin's finches, was significant largely in its role as a controlled microcosm in which to develop the more sophisticated techniques to solve more difficult and significant problems. Similarly, the choice of chess was, for Simon and his fellow computer scientists, solely a function of its intrinsic technical characteristics: chess was the ideal experimental technology for AI because it was both simple enough to be able to formalize mathematically and yet complicated enough to be theoretically interesting (Newell et al., 1958; Shannon, 1950).

It would be a poor historian of science indeed, however, who failed to see in any evocation of drosophila the opportunity to ask questions of deeper analytical significance. As the work of Robert Kohler and others remind us, the choice of an experimental organism (or in this case, technology) is never an epistemologically neutral decision (Burian, 1993; Kohler, 1994; Waters, 2004). Not only are such decisions often driven by practical as well as intellectual factors – the ease with which the human-friendly drosophila adapted itself

to the lifecycle and ecosystem of the laboratory meant that other, less opportunistic organisms, such as neurospora, were relegated to the sidelines – but they often have long-term implications for the research agenda of a discipline that are unexpected and perhaps even undesirable. The widespread adoption of drosophila as the experimental organism of choice for early 20th century genetics research, for example, meant that certain research agendas, such as transmission genetics, became dominant, while others, such as embryology, were neglected (Mitman and Fausto-Sterling, 1992). In a similar manner, the success of computer chess, and in particular an approach to computer chess based on deep-tree searching and the minimax algorithm, came to dominate mid 20th century approaches to AI research, overshadowing other problem domains and techniques. Unlike drosophila, however, and despite its apparent productivity as an experimental technology, computer chess ultimately produced little in terms of fundamental theoretical insights.

This paper explores the role of computer chess in defining the identity and research agenda of AI over the course of the previous half-century. The central argument is that the decision to focus on chess as a representative measure of both human and computer intelligence had important and unintended consequences for the discipline. In choosing chess over its various alternatives, AI researchers were able to tap into a long tradition of popular chess culture, with its corresponding technical and theoretical literature, international networks of enthusiasts and competitions, and well-developed protocols for documenting, sharing, and analyzing data. Yet the brute-force computational techniques that proved most suitable for winning computer chess tournaments distracted researchers from more generalizable and theoretically productive avenues of AI research. The rise to dominance of minimax algorithm-based techniques in particular transformed chess from a quintessentially human intellectual activity into an exercise in deep searching and fast alpha–beta pruning (Marsland, 1991). As a result, computers got much better at chess, but increasingly no one much cared.

The point of this paper is not to give a definitive answer to the question of whether or not chess was the drosophila of AI. Like most such analogies, this one holds true in certain circumstances but in others breaks down. To a certain degree, what matters historically is not so much the fundamental legitimacy of the comparison, but how practitioners have made use of it. AI researchers themselves first proposed the relationship between chess and drosophila, and they mobilized this claim early and often in the self-construction of their discipline. If for no other reason, computer chess is significant in the history of AI because AI researchers believe it to be significant. But the real similarities between chess and drosophila are also illuminating, as they suggest new approaches for thinking about the computer sciences in terms of the larger history of the experimental sciences.

The history of computer chess has immediate relevance to historians of computing, cognitive science, operations research, and decision theory, as well as to scholars interested in the more general question of how experimental technologies shape scientific practice. But this paper also has a larger, more theoretical agenda: it represents an attempt to situate the history of computer software – as an object of historical inquiry quite distinct from the history of the computer itself – within the larger context of the history of science.

Despite numerous calls from historians of computing for further study of the history of software, software thus far has proven remarkably resistant to historical analysis (Campbell-Kelly, 2007; Ensmenger, 2009; Hashagen et al., 2002; Jesiek, 2006; Mahoney, 2008).

In part, this is a reflection of the inherently amorphous nature of software: unlike the computer itself, which is obviously and tangibly technological, software is generally invisible, ethereal, and ephemeral. In many cases, it exists only as a unique – and temporary – arrangement of digital bits buried deeply within a tiny microprocessor. Certain aspects of software, such as sorting algorithms, can be generalized and formalized as mathematical abstractions, while others remain inescapably local and specific, subject to the particular constraints imposed by corporate culture, informal industry standards, or government regulations. In this sense, software sits uncomfortably at the intersection of science, engineering, and business. Software is where the technology of computing meets social relationships, organizational politics, and personal agendas (Ensmenger, 2010). As a result, software is difficult to situate historiographically. To the extent that the history of science has engaged with the history of software, it has treated it as the intellectual history of computer science. But in the real world, in order to transform ideas into action, software must necessarily become embodied: even the simplest algorithms, when translated from Platonic ideals into the specific forms required to operate specific computers, in specific socio-technical environments, become clearly constructed technological artifacts. Software encompasses not only computers, codes, algorithms, and ideas, but also people, practices, and networks of interaction. In this sense, software development is perhaps the ultimate expression of what the sociologist John Law has called 'heterogeneous engineering' (Ensmenger, 2009; Law, 1987).

And so this paper also represents an attempt to think seriously about software as a material artifact; as a technology embedded in systems of practice and networks of exchange. More specifically, it uses the history of the minimax algorithm, the computational equivalent to the drosophila chromosome, to open at least partially the black box of software to the light of historical and sociological inquiry. Buried deeply within most chess-playing computer programs, rendered largely invisible by the literal black-box of a silicon-encased microprocessor, the minimax algorithm can nevertheless be exposed to the scrutiny of historical and sociological analysis. By revealing the social history of even the most seemingly straightforward applications of computer software (one of the many virtues of chess, after all, is that it has unambiguous rules and well-established measures of success), this paper hopes to exemplify a new approach to the history of software. Computer chess in this respect will serve not as the drosophila of AI, but as the drosophila of the *history* of AI.

From Mechanical Turk to virtual fruit fly

The story of computer chess begins long before the invention of the first electronic computers. The historical origins of the chess-playing machine – and of metaphysical speculations about the relationship between mechanical chess and human intelligence – stretch back well into the 18th century (Lohr, 2007; Standage, 2002). In 1770 the Hungarian engineer Wolfgang von Kempelen constructed, as entertainment for the Empress Maria Theresa, an Automaton Chess Player. Kempelen's automaton, also known as the Mechanical Turk (the humanoid portion of the machine was dressed in robes and a turban), could not only play a strong game of chess against a human opponent, but could also solve certain mathematical chess puzzles, such as the so-called Knight's Tour.[3]

For 84 years the Mechanical Turk was exhibited throughout Europe, playing (and defeating) such illustrious figures as Benjamin Franklin and Napoleon Bonaparte (and

this despite the fact that Napoleon cheated). In 1783 it played a close game against François-André Danican Philidor, widely considered by contemporaries to be the world's best human player. In 1818 an exhibition of the Turk, then owned by the aspiring mechanician and musician Johann Mälzel, was visited in London by a youthful Charles Babbage, who was reportedly entranced. After playing two games against the Turk, Babbage was not only inspired to acquire his own automaton (albeit a dancer, not a chess player), but also to sketch out his own plans for a chess-playing machine (*New York Times*, 1875; Schaffer, 1996). He would later argue that chess was one of the compelling applications for his (never-constructed) Analytical Engine (Babbage, 1864). In addition to inspiring Babbage, the Turk also spawned numerous contemporary imitators, including the chess-playing machines known as Ajeeb ('The Egyptian') and Mephisto (Jay, 2000).

The Mechanical Turk was not, in fact, an actual chess-playing machine, but rather an elaborate hoax involving a hidden human player, an articulated mechanical arm, and a series of magnetic linkages. But despite frequent attempts to uncover its secrets – including an 1836 article by Edgar Allen Poe in which Poe transformed (some say plagiarized) a contemporary exposé of Mälzel's Automaton Chess Player and turned it into the model for the modern detective novel – the Mechanical Turk largely retained its mysterious appeal (Panek, 1976; Poe, 1836). It was not until the mid 20th century that a definitive account of its inner workings was made public. In any case, the possibility, at least, of mechanical chess remained a source of continual fascination, and was treated with serious attention by engineers, futurists, philosophers, mathematicians, and cyberneticians (Ashby, 1952). In 1914, the Spanish engineer Leonardo Torres y Quevedo built the first actual chess-playing machine, which was capable of king and rook against king endgames without any human intervention (Randell, 1982).

The long-standing public fascination with chess-playing automata provides some context for understanding the popular appeal of the chess-playing computer, and explains in part why computer chess has played such a prominent role in the public presentation of AI research. Automata in general problematized the boundary between the organic and the artificial, and chess automata in particular raised questions about the distinctiveness of human cognitive activities (Riskin, 2007; Sussman, 1999; Voskuhl, 2007). Chess was, in this context, not just any game: the traditional province of kings (and scholars), chess had long been recognized as the pinnacle of human intellectual accomplishment, requiring both deliberate, carefully cultivated learning and strategy, as well as bold, creative, and courageous flights of inspired brilliance. As such, chess-playing ability was widely considered to be a strong indicator of more general intelligence. In fact, a broad range of thinkers, from Goethe to Franklin, had made chess a metaphor for war, romance, politics, commerce, sports, and just about every other complex human cognitive and social activity (Kasparov, 2007; Rasskin-Gutman, 2009; Shenk, 2006; Steiner, 1971).

Because chess was historically regarded as such an essentially human endeavor, the ability of machines to play chess seemed to have fundamental metaphysical implications. If a machine could emulate what was widely considered the summit of human intelligence, namely the abstract reasoning associated with chess-playing ability, then was it not possible that the essence, and not just the appearance, of humanity could eventually be reproduced mechanically (Guterl, 1996; Husbands et al., 2008)? The frequent portrayal of the first electronic computers as 'giant brains' made the connection between mind and machine, exemplified by chess-playing computers, all the more obvious (Berkeley, 1949;

von Neumann, 1958; Yood, 2003). AI researchers would play explicitly on the broader symbolic significance of chess. As Herbert Simon would famously declare in his 1973 defense of chess as drosophila, since chess was 'the intellectual game par excellence', by devising a successful machine, 'one would seem to have penetrated to the core of human intellectual endeavor' (Simon and Chase, 1973).

The specific origins of computer chess (as opposed to mechanical chess) are often traced back to the mathematician Alan Turing. As early as 1946, Turing imagined a chess-playing computer as one possible example of a 'thinking' machine (Turing, 1946), and there is evidence that by 1948 he and his colleagues at the National Physical Laboratory were discussing in very tangible terms a potential chess-playing machine.[4] In 1953 Turing would write the first chess-playing program (on paper, as no machine yet existed that could actually run his program), and much of Turing's later speculations on the possibility of 'machine intelligence' revolved around an imagined chess-playing computer (Turing, 1950, 1953).

But it was the mathematician Claude Shannon who wrote the very first article ever published on the art of programming a computer to play chess. Like many subsequent computer theorists, Shannon believed that chess was the ideal experimental technology for AI because it was a) 'sharply defined both in allowed operations (the moves) and in the ultimate goal (checkmate)' and b) 'neither so simple as to be trivial nor too difficult for satisfactory solution' (Shannon, 1950). The discrete nature of chess – the fact that the positions on a chessboard could be easily described in terms of a simple 8×8 grid – also meant that it was particularly compatible with the digital nature of modern electronic computing. More significantly, however, since 'chess is generally considered to require "thinking"', Shannon argued, 'a solution to this problem will either force us to admit the possibility of mechanized thinking or to further restrict our concept of "thinking"' (Shannon, 1950). Like many of his contemporaries, Shannon considered chess mastery to be an indicator of more general human intelligence. It seemed to follow therefore that a computer that could play chess was de facto intelligent – or at least capable of simulating a close approximation of intelligence. (This neat side-stepping of long-standing metaphysical discussions about the nature of the mind was characteristic of many AI researchers in this period.)

After briefly describing how a computer might represent internally the configuration of positions on a chessboard, Shannon proposed several approaches to teaching it to play. In theory, it would be possible to play a perfect game of chess, or to construct a machine to do so, simply by following a relatively straightforward algorithm. Because chess is a finite game, with a finite number of positions, each of which allows for a finite number of moves, and because the rules of chess guarantee that every game must eventually end in a win, draw, or loss, all of the possible combinations of moves and counter-moves (each of which is technically known as a 'ply') can be laid out in advance as a branching tree of decision points. By working backward from the end-points of this decision tree, the optimal move for any given position could readily be computed. Other than the flip of the coin that determined which player moves first, there are no random elements to a chess match. Perfect information is available to both players at every step of the process. For Shannon, therefore, chess was an entirely deterministic game: in theory, once a comprehensive decision tree was constructed, the outcome of any (and indeed every) possible game could be calculated in advance. The 'problem' of chess could be then be considered to have been solved conclusively.

In practice, however, the perfect game of chess, while theoretically computable, is effectively unattainable. The numbers involved in constructing even a partially complete decision tree for chess quickly become astronomical – and intractable. Given an average of 42 moves (84 plies) per game, with an average of 38 legal moves to consider per ply, the typical master's level chess match would require a total of 38^{84} (roughly 10^{134}) positions to be evaluated. Just to put this number (10^{134}) into perspective, if every atom in the universe (10^{75} of them) were a chess computer operating at the speed of Deep Blue (10^{6} moves per second), there still would not have been enough time since the Big Bang (10^{18} seconds) to consider each of these combinations. Compared with such large numbers, even the exponential growth in computer power promised by Moore's Law ultimately proves insufficient. The full decision tree simply has too many branches to evaluate. There is not, and will never be, a comprehensive computational solution to chess.

Faced with the impossibility of calculating the combinatoric possibilities of an entire chess game, computer chess programs must necessarily evaluate only a more limited number of moves. Shannon himself proposed two potential solutions for 'pruning' the decision tree. The most obvious solution was to reduce the total number of moves that a computer was required to 'look ahead'. This would make the overall decision tree to be evaluated smaller and more manageable, and therefore more amenable to straightforward computational approaches. Shannon called this approach a 'Type-A' solution, and considered it to be a brute-force method that did not accurately reflect the ways in which human beings played chess. He much preferred a 'Type-B' solution that used sophisticated heuristics to trim the decision tree by privileging certain branches over others. Like human grandmasters, Type-B solutions would focus only on the most promising lines of analysis, and would recognize in patterns of positions more general principles of play that would reflect a more truly intelligent approach to the problem of chess.

The sharp distinction Shannon drew between these two very different approaches –Type-A and Type-B – anticipated a debate that would soon emerge within the discipline about the relationship between artificial and natural intelligence. The lines of debate were drawn along a number of different axes, some philosophical, others pragmatic, but the central dilemma hinged around the question of whether it was necessary for AI to simulate (and therefore understand) natural intelligence, or whether it was enough simply to replicate its functionality (Collins, 1990; Dreyfus, 1992; Searle, 1999). In other words, was it important that intelligent machines 'think' like humans, or was it sufficient that their behavior appeared to be intelligent? Or to put it in terms of computer chess, does a computer that plays chess, no matter how skillfully, ever truly 'understand' chess, and does it matter one way or another? While this might seem at first to be a purely metaphysical distinction, the implications for both the computing and the cognitive sciences are significant: at stake are a set of fundamental distinctions between the class of problems whose knowledge domains are explicit (and therefore potentially comprehensible to a computer) and those whose knowledge domains are tacit (and therefore difficult, if not impossible, to automate) (Brooks, 1990; Collins, 2010).

But while Shannon clearly favored the mimetic, Type-B approach to machine intelligence, his paper described in detail only a functional Type-A solution, a brute force approach built around the so-called minimax algorithm. Although Shannon believed that his, like all other Type-A solutions, was doomed to be 'both slow and a weak player', in practice his

minimax approach proved unexpectedly powerful, and durable. Because the minimax algorithm was relatively simple and easy to understand, it could be quickly and readily implemented on a broad range of computer machinery. Its performance scaled linearly with improvements in the underlying hardware. It was an algorithm that was amenable to tinkering and remarkably resilient to programming errors. And, perhaps most importantly, minimax played a pretty decent game of chess. As a result, the minimax approach disseminated quickly and widely throughout the emerging computing community. By the end of the 1960s, minimax so dominated computer chess competitions that all other approaches were effectively abandoned. By providing a well-defined solution (minimax-based computer chess) to a poorly defined problem (machine intelligence), Shannon's concrete and authoritative answer foreclosed for the time being any discussion about the underlying nature of more fundamental questions.

How a computer sees a chess game

To understand the significance of the minimax algorithm in the subsequent history of AI, it is necessary first to understand how a computer 'sees' a chess game.

To begin with, the computer must be programmed with a numeric representation of the positions on the board. Fortunately for the early computer chess researchers, by the beginning of the 20th century there were already well-established systems for describing and recording chess games. The most widely used was the algebraic chess notation, which used a unique letter–number pair to identify each square of a chessboard. In algebraic notation, vertical files were labeled a–h, and horizontal ranks 1–8. Individual pieces were labeled with an uppercase letter (in English, for example, K for king, Q for queen, B for bishop, and so on), and moves were designated by a combination of piece and position. The notation Be5, for example, indicated that a move by a bishop to file e and rank 5. Which bishop was moved depended on context and the column in which the move was noted (left column is white, right column black).

Although most computers did not use algebraic chess notation as their internal representation of the board, the fact that most human players were already familiar with symbolic representations of chess made the transition between real and virtual chessboards much easier. The specific internal implementation mattered little; what was important was that the widespread use of notational systems meant that computer chess researchers had available to them an enormous amount of data – historical matches, opening books, end-game solutions, puzzles, and post-game analysis – already translated into convenient, machine-readable form. The use of chess notation to transform ephemeral local performances into structured data allowed for the accumulation and codification of chess knowledge.[5] The widespread adoption of systems of chess notation also made it possible to play chess at a distance: the combination of notational systems, communications networks, and protocols for correspondence chess provided a context for the first man-versus-computer chess tournaments, since the practice of playing chess against an unseen (and unknowable) opponent had already been well developed.

In any case, once a computer has been provided with an internal representation of a chessboard and a system for determining which legal moves are available to a given piece in a given position, the process of constructing a decision tree can begin.

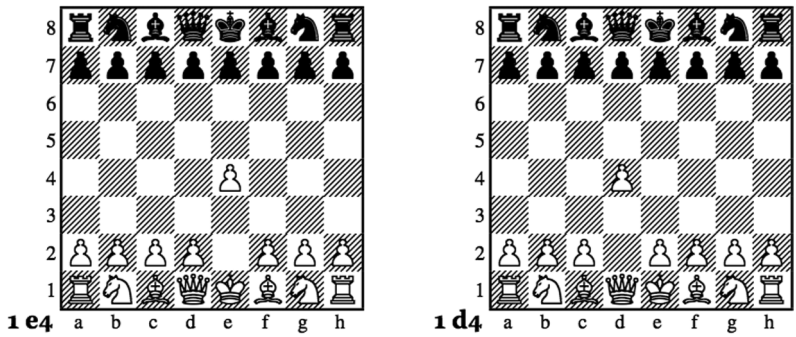

Figure 1. The King's Pawn (d4) and Queen's Pawn (e4) opening sequences are common and well documented in chess books.

Consider, for example, the two popular opening moves e4 and d4, which represent two possible options for the first ply of a hypothetical game (see Figure 1).

In order to evaluate the relative strength of these two moves, the computer first generates a set of possible responses by black. These responses would represent the second ply. In practice, the moves to be considered in the second ply would generally include only a subset of all the possible legal moves as determined by a 'plausible move generator'. The development of sophisticated plausible move generators was the key to Shannon's Type-B solutions, but even in Type-A solutions they are used (in much simplified form) to reduce the total number of moves to be considered. The plausible move generator is one of several components of minimax algorithm that were subject to constant experimentation by enthusiasts.

The subset of plausible moves generated for the first and second plies is then arranged into a decision tree, with each branch representing one possible combination of moves and counter-moves (see Figure 2). The tree can be grown as needed simply by using the plausible move generator to compute additional plies. The practical limit of the tree depends on the available hardware; the Deep Blue computer, for example, evaluated a tree that contained on average 17 to 18 plies.

Once a decision of the desired depth has been constructed, a numeric score is assigned to the terminal node of each branch using an evaluation function. The evaluation function attempts to rank the relative strength of the various final positions represented by the terminal nodes. Most evaluation functions make use of some combination of the number of pieces remaining and the strategic value of certain positions on the board to provide a relative rank for each outcome. In general, positive numbers are used to represent positions favorable to white and negative numbers for positions favorable to black.

In a complete decision tree that traced all of the possible branches to their ultimate conclusion, the construction of the evaluation function is trivial: each match would conclude with either a win, loss, or a draw. Such comprehensive trees are typically only computationally feasible in the end-game; for practical reasons, most decision trees are necessarily incomplete. For such partial trees, developing an evaluation function is much more complicated, because it must incorporate some method for evaluating the relative strength of each position, without reference to some known final conclusion. There is no single, optimal

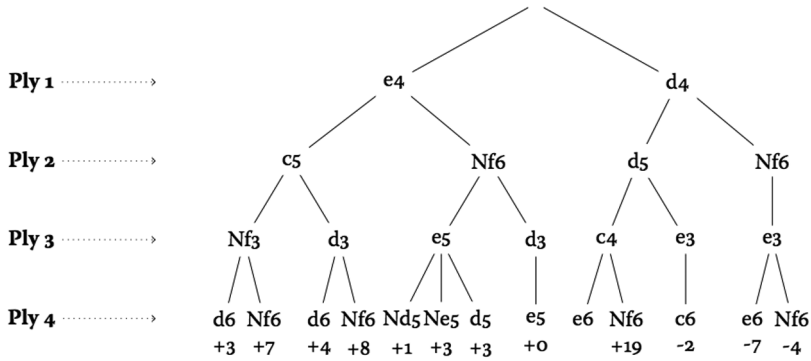

Figure 2. A truncated decision tree for the opening moves e4 and d4 with the evaluation function applied to the terminal leaves.

technique for ranking relative positions: the construction of a robust evaluation function is more of an art than a science, and as such, provides endless opportunities for tinkering.

One common technique used to develop an evaluation function is to assign each piece remaining on the board a numeric value: 1 point for a pawn, 3 points for a knight or bishop, 5 points for a rook, and 9 points for a queen, an effectively infinite number of points for the king. Other points might be rewarded for pieces that controlled tactically significant positions. Although all such evaluations were inherently subjective, workable techniques for ranking positions and outcomes were already well developed by the 1950s to facilitate chess learning and analysis by human players. This is yet another example of how the cultural history of chess made it an ideal experimental technology: computer chess researchers were able to leverage of the vast body of preexisting technique and literature that was both quantitative and analytical without being deterministic (van den Herik et al., 2002).

The fact that most branches of a decision tree do not culminate in an objectively measurable outcome (win, loss, or draw), meant it was always possible that the application of a given evaluation function would produce an outcome that was locally but not globally optimal – meaning that the outcome represented what appeared to be a strong position, but in the long term produced a losing outcome. This is known as the 'horizon effect'. Given the limits on the depth of the search tree, it is always possible that just over the horizon – the next, un-evaluated move, for example – might lie a looming disaster. To a certain degree, the horizon effect is similar to the problem of optimization in mathematics: although it is usually possible to determine local extrema (maxima and minima) for a given function, determining whether these are also global extrema is often analytically impossible. There are strategies and heuristics than can be used to assure the likelihood that a local solution is also a global optimum, but they are never infallible. Techniques for identifying how far over the horizon to pursue stable position solutions, also known as quiescence search, represented one of the few aspects of the minimax approach to computer chess not susceptible to mechanical solution (see Figure 3). As such, quiescence search served as a site of innovation and experimentation.

It is important to note that the development of the evaluation function is the only aspect of implementing the minimax approach to computer chess that requires any substantial

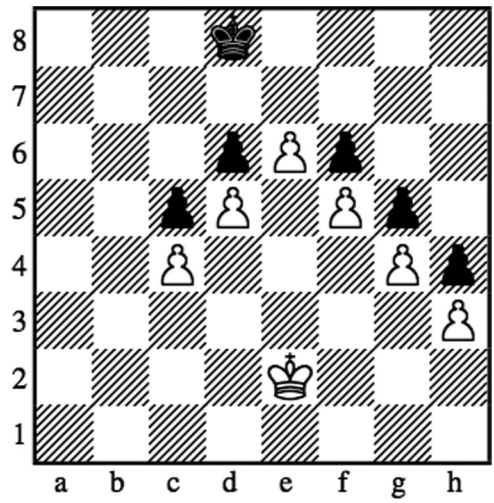

Figure 3. The Horizon Effect: White has a win in 10 moves, but a search depth of less than 20 plies will fail to reveal it. Example, with permission, from Brudno (2000).

chess ability or knowledge. The remaining steps of the algorithm were entirely mechanical. There is some evidence, in fact, that the less the computer 'knew' about chess, the better it performed (Schaeffer and Donskoy, 1989). This made the minimax approach exemplified by computer chess very different from other approaches to AI, such as rule-based expert systems or Bayesian statistics-based inference engines, which emphasized domain-specific knowledge. Computerized chess-machines might act intelligently, but they were intrinsically (and perhaps even necessarily) ignorant.

Once the decision tree is constructed and the evaluation function used to apply a numeric ranking to each of its terminal nodes (or 'leaves'), the application of the minimax algorithm is essentially mechanical. Only the terminal nodes needed to be evaluated, as the minimax algorithm itself would fill in all the intermediate nodal values, saving both time and processing power. Beginning at the terminal nodes, the algorithm works backward through the decision tree, assigning optimal values to each decision branch by alternatively maximizing or minimizing the outcome. If a decision point at a given level of the tree (for example, ply x) represents a move for black, the algorithm assumes that the player will always follow the branch that minimizes the score at the subsequent node (ply x+1). The white player, considering the set of branches earlier in the decision tree (ply x–1), would therefore know that the maximal outcome possible at that level is limited to the minimum of the level above. At each branch black will minimize, and white will maximize. By alternating between maximizing and minimizing strategies at each level of the decision tree, and 'backing up' these optimal outcomes to the previous branch level, the entire tree can be populated quickly and efficiently with numeric rankings. This is an entirely mechanical process that does not require the application of the computationally expensive evaluation function: the ranking of each node is determined by the optimum response (for the opposing player) at the following level.

Once all of the rankings have been assigned, the first player to move (in this case white) would simply walk through the tree, maximizing and minimizing alternatives. Given a

well-defined decision tree with unambiguous numeric rankings at the terminal nodes, and two players each with access to full information (a characteristic feature of chess – as opposed to, say, poker) the entire decision-making process can be reduced to a simple search process whose effectiveness relies almost solely on the brute force application of computational power. More significantly, the performance of the minimax search algorithm scales linearly with processing power. Build a faster computer, and you automatically have a better performing chess machine.

The multiple virtues of minimax

Although Claude Shannon published his important early paper on computer chess in 1950, it was not until 1957 that the first working chess program was implemented. There was a brief period of experimentation with alternatives to the Type-A minimax algorithms. For example, IBM programmer and chess player Alex Bernstein pursued a selective pruning Type-B strategy. However, the minimax algorithm soon emerged as the dominant approach to computer chess (Bernstein and Roberts, 1958). In 1958, Allan Newell, Herbert Simon, and Clifford Shaw modified the minimax algorithm to include a technique called alpha–beta pruning (which appears to have been invented simultaneously by at least several others, including John McCarthy). This program allowed entire branches to be quickly eliminated from the search process. The combination of the minimax algorithm with the alpha–beta pruning technique significantly reduced the total number of branches of the decision tree that need to be considered, which made it possible to play chess on almost any computer, including the earliest microcomputers. It also made the minimax algorithm faster, more robust, and generally more powerful than its competitors. After the Chess 4.0 program, which was based on minimax and alpha–beta pruning technology, won the first Association for Computing Machinery (ACM) computer chess championships in the early 1970s, alternative Type-B approaches were almost completely abandoned.

Many features of the minimax algorithm made it particularly well-suited to the application of computer chess. To begin with, it was easy to understand and to program, even on (when viewed with hindsight) primitive and low-powered equipment. The Microchess program, for example, written by Peter Jennings in 1976 for the MOS Technology 6502 microprocessor, could fit into a mere 1024 bytes of memory. The relative ease with which the minimax algorithm could be implemented allowed chess programs based on it to disseminate rapidly throughout the electronic computing ecosystem. Like a hardy weed or the adventurous fruit fly, the minimax algorithm was a fit competitor in many environments. And once established, it proved difficult to replace or eradicate.

In addition to being easy to program, the performance of the minimax algorithm scaled linearly with improvements in hardware. This meant that every improvement in computer hardware translated directly into faster, more powerful chess engines. With each new generation of computer hardware, chess systems based on the minimax algorithm became immediately, impressively – and above all measurably – better. This is not true of every computer algorithm, much less of every approach to AI. Compared with their competitors, researchers in computer chess were able to demonstrate constant progress, no small consideration for a field that generally failed to live up to its audacious claims and predictions. Every year minimax-based chess programs would win more tournaments and defeat more highly ranked human players.

The minimax algorithm also allowed for easy tinkering. Not only was it simple to implement, but it also was resilient to errors. In other words, it was easy to modify but difficult to break. Chess programs built around minimax were inherently modular, meaning that their component parts were relatively independent of one another. Once the basic minimax algorithm was up and running, the other elements of the system – the evaluation function, the plausible move generator, and the quiescence search – could be isolated with ease and experimented upon. The evaluation function was a particular site of much innovation, and incremental improvements drawn from chess theory, historical matches, or personal experience were relatively simple both to implement and test. The quiescence search, which helped alleviate the risks of the horizon effect, was similarly amenable to experiment and improvement. The plausible move generator proved more resistant to fundamental innovation, but nevertheless allowed for small, incremental improvements without threatening the stability of the larger system.

Finally, the minimax algorithm worked. The technique of deep searching in a decision tree proved effective when applied to the problem of computer chess. Although it would take decades before computers could approach the level of play achieved by the very best human players, even the early systems played a respectable game. Like Bridges' and Sturtevant's early chromosomal maps of *Drosophila melanogaster*, chess programs based on the minimax algorithm were over-simplifications of reality. However, they were relatively easy to develop, provided clear value, and over time could be consistently refined and improved. It is no coincidence that the first published claim that chess was the drosophila of AI appeared just as the practice of computer chess stabilized around the minimax algorithm.

Cultures of chess and computing

Although the multiple virtues of the minimax algorithm explain much about why minimax quickly became the dominant approach to computer chess, they do not, in and of themselves, explain how and why computer chess came to be seen as the drosophila of AI. Most of the earliest computer chess researchers justified their choice of experimental technology on the basis of its internal characteristics: according to them, chess was ideal because it was a simple game with straightforward rules that could be readily formalized. And it is true that, unlike other AI research agendas at the time (such as machine translation), chess provided a well-defined problem domain, with unambiguous rules and clear objectives and measures of success. But chess was not the only game in town with these particular virtues, as early flirtations by AI researchers with alternatives such as checkers, Nim, and Go made apparent. The choice of any of these (particularly Go) might have shaped the research agenda of the discipline in a very different direction (Brown, 1979; Dowsey, 1973; Schaeffer, 2001). Chess did have some unique advantages over its competitors, but these were not necessarily inherent in the structure of the game. Its ultimate triumph has to do with very specific material and social practices that had developed around chess over the course of the previous century. Again, consider Robert Kohler's explanation for the rise to dominance of the actual drosophila: although the specific characteristics of the drosophila genome were significant, so too were the features of the species and its lifecycle that fit well within the academic environment and culture. It was the combination of organism, culture, and practice that made drosophila so successful. So too with computer chess.

One obvious advantage of chess was that it was well-known and popular. This was not true of alternatives such as Go, which despite being one of the most ancient of board games, was not widely played outside of Asia. Perhaps more significantly, however, chess was also generally recognized as a complex, creative game that required strategy and planning; thus, the ability to play good chess was widely considered to be a strong indicator of more general intelligence. Chess was a prestigious activity, long associated with intellectuals, artists, and individual genius, as well as a grand metaphor for other complex human cognitive and social activities (Kasparov, 2007; Shenk, 2006; Steiner, 1971; Rasskin-Gutman, 2009). The presumed broader significance of chess made it particularly symbolic for AI researchers: recall, for example, Herbert Simon's claim that 'if one could devise a successful chess machine, one would seem to have penetrated to the core of human intellectual endeavor' (Simon and Chase, 1973). Because chess was historically regarded as such an essentially human endeavor, the ability of machines to play chess seemed to have fundamental metaphysical implications.

Chess also happened to be a game with a unique and idiosyncratic historical association with mathematics and computing. Many of the mathematicians who worked on computer chess, including Turing and Shannon, were avid amateur players. And as I have written about elsewhere, chess-playing ability has long been associated with programming ability (Ensmenger, 2010). Many of the early advertisements for programming positions emphasized chess and musical aptitude, and several of the IBM Corporation's earliest programmers were chess players, including one US Open Chess champion. 'Look for the chess player, the solver of mathematical puzzles', advised one representative contemporary article on the selection of computing personnel (O'Shields, 1965). Throughout the 1950s and 1960s the ability to play chess was embodied early on in the hiring practices of the computer industry through the use of aptitude tests and personality profiles, and so many computer practitioners brought with them an interest in the game that it quickly became self-perpetuating. It was simply assumed within this community that chess mastery was synonymous with generalized intelligence.

The existence of an extensive body of historical and theoretical literature on chess also greatly facilitated its adoption by computer enthusiasts. This was made possible by, and encouraged the standardization of, a comprehensive system of symbolic notation that allowed chess researchers to 'seed' their system with reliable data, and to validate its performance against a wide variety of opponents and situations. Books of standard chess openings could be stored as pre-computed solutions to the most common early sequences of moves, for example, and popular end-game puzzles could be used to prove the effectiveness of quiescence search algorithms. Any modifications made to a chess engine could be quickly tested against the standardized sequences of historical games and puzzles. In addition, well-established systems of chess notation facilitated the development of protocols for playing correspondence chess and other forms of chess-at-a-distance. The popularity of correspondence chess accustomed players to playing remotely against unseen, unknown opponents separated in space and time. This familiarity with playing chess by mail (and wire) smoothed the eventual transition to playing chess against a computer opponent.

In addition to systems of notation, the chess community had established by the middle of the 20th century a well-developed system for ranking the relative strength of its players. During the 1950s this was the Harkness system. By the 1960s, the Elo rating system, named

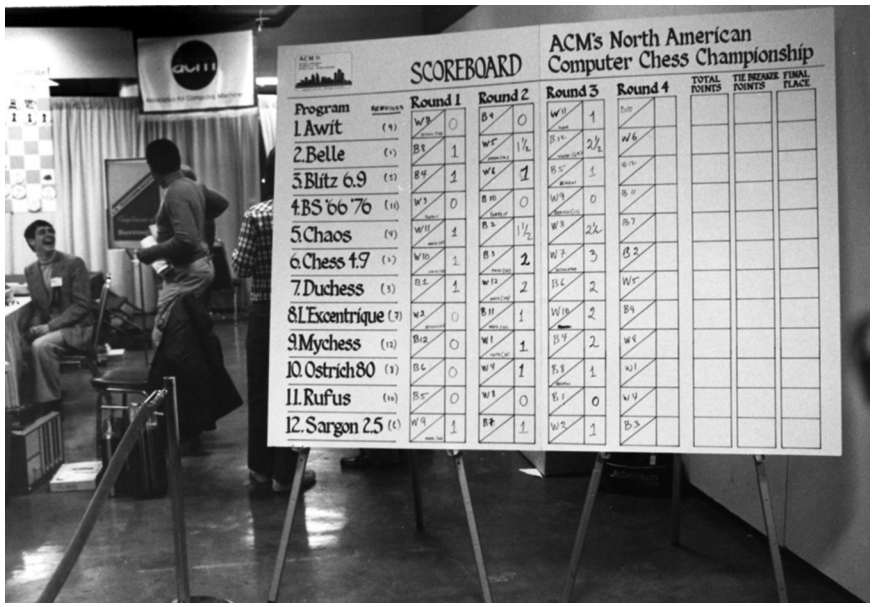

Figure 4. Scoreboard, ACM National Computer Chess Competition, 1979. Courtesy of the Computer History Museum.

after the Hungarian-born physicist Arpad Elo, had been formally adopted by the US Chess Federation. The details of these systems are irrelevant. What matters is that they provided clear numerical benchmarks for measuring performance and improvement. Against such benchmarks, it was possible for computer chess programs to make continual and measurable progress – a clear advantage that chess had over other problem domains in AI research, where such unambiguous measures of performance were generally not available. In science, as in other disciplines, success speaks volumes and, given the larger context in which AI found itself in the late 1960s and 1970s, the ability of computer chess programs to demonstrate continual and convincing progress, often in sensational fashion, helped elevate computer chess to its paradigmatic status as the experimental technology of AI. There were no equivalent accomplishments for researchers working on natural language processing or deductive reasoning programs. In terms of public perceptions of success, at least, chess was clearly king.

These numeric systems of ratings were enabled and reinforced by an extended network of chess tournaments that regularly took place throughout the world. Chess tournaments provided forums for players to establish their reputations and improve their ratings. They had well-developed protocols for determining winners and resolving disputes, and provided a community of practitioners that encouraged both competition and cooperation (Peterson, 1983). Chess tournaments enabled enthusiasts to cultivate their skills, validate their experiences and obsessions, and share information. Human chess tournaments became the model for the first computer chess tournaments, the very first of which were held in the mid 1960s (see Figure 4). By the late 1960s computers were able to cross over into the conventional

chess tournament circuit, and in 1967 the MacHack Six, a system developed by Richard Greenblatt and others at MIT, became the first computer to defeat a human player in official tournament play (Greenblatt et al., 1967; Levy, 1976).

Computer chess tournaments also served as a form of public spectacle. From the earliest origins of computer chess, chess matches between human and computer have been sensationalized as the ultimate embodiment of The Clash Between Men and Machine. The public staging of these matches capitalized on the dramatic conventions of the traditional chess tournament, including, in at least one case, formal evening wear (at least on the part of the human opponent). AI enthusiasts made the most of such performances. When the MIT philosopher Hubert Dreyfus (1965) mocked the limitations of contemporary chess engines (one of which had recently been defeated by a 10-year-old human opponent) in his sweeping critique of AI, AI researchers staged a match between Dreyfus and the state-of-the-art MacHack program. Dreyfus, who never claimed to be skilled player, was soundly defeated. 'A Ten Year Old Can Beat the Machine–Dreyfus: But the Machine Can Beat Dreyfus' triumphed the Bulletin of the Association of Computing Machinery's Special Interest Group in AI, and the so-called 'Dreyfus affair' was widely mobilized to deflect criticism of the discipline (McCorduck, 1979). The fact that Dreyfus himself was defeated by a computer was irrelevant to his larger critique, but made for compelling rhetorical theater (Koschmann, 1996; Papert, 1968; McDermott, 1976). The establishment in 1980 of the Fredkin Prize, which offered a US$100,000 bounty to the first computer program capable of beating a reigning world chess champion, would only accelerate this narrow focus on building machines that could defeat people.

The drama of the computer chess tournament played particularly well in the Cold War context. The very first game played between competing computer systems pitted the McCarthy–Kotok program against a Soviet system developed by Alexander Kronrod at Soviet Institute of Theoretical and Experimental Physics. Against the backdrop of the early 1970s confrontation between Bobby Fisher and Boris Spassky, the showdown between American and Soviet computing technology assumed a particular salience (Johnson, 2007; Schonberg, 1981). Throughout the 1960s and 1970s, international tournaments were held pitting American computers against their Soviet counterparts, both human and machine (*Wall Street Journal*, 1978) (see Figure 5). The culmination of this series would, of course, be the ultimate defeat of World Chess Champion Garry Kasparov by the IBM Deep Blue computer. Not only would such showdowns raise the profile and enhance the reputation of AI researchers; but they would also reinforce the association between human and machine cognition, at least as it applied to chess-playing ability.

Even without the Cold War theatrics, the computer chess tournament provided an opportunity for AI researchers to publicly demonstrate consistent and impressive progress. This was particularly valuable in the late 1970s, when the publication of the highly critical Lighthill Report by the British Research Council led to a reduction of funding for AI research worldwide (Crevier, 1992). Tournaments provided regular contact with a passionate community willing to tinker with and cultivate new technologies. This relationship proved particularly productive after the development of microcomputer technology in the late 1970s made it possible for thousands of amateur computer enthusiasts to make an additional hobby out of computer chess. The scalability of the minimax algorithm meant that even the most anemic microcomputers could play computer chess (in fact, chess was

Figure 5. Chess diplomacy, *Wall Street Journal* (1978).

often one of the few 'real' applications that these microcomputers could manage). The version of Microchess that Peter Jennings wrote for the Commodore, Apple II, and Atari line of microcomputers was the first microcomputer package to sell more than 50,000 copies. The culture of mutual encouragement and code-sharing that characterized many of the early 'homebrew' computer clubs mirrored the similarly open ethos on the part of chess aficionados. The norms and practices of both communities encouraged tinkering and the voluntary dissemination of information. Variations of the minimax-based computer chess programs traveled quickly and easily within this hospitable environment, and by 1977 the first International Computer Chess Association was established. Over the course of the next several decades, hundreds of microcomputer-based chess programs were developed by both amateurs and professionals.

Game over?

From a purely pragmatic perspective, it is easy to see why AI researchers so enthusiastically embraced chess as their discipline's drosophila. The link between the ancient game of chess and the novel technology of electronic computing had proven productive from the very beginning. Within just a few years of the coining of the term 'AI' (at the 1956 Dartmouth Conference), one of its leading proponents, Herbert Simon, had elevated chess to the top of the disciplinary research agenda. Although Simon's bold prediction that 'within ten years a digital computer will be the world's chess champion' was seen at the time as being wildly optimistic, within 50 years AI had achieved this defining accomplishment (Newell et al., 1958). Especially when compared with other areas of AI research, many of which had failed to develop, computer chess appeared to be an almost unalloyed success. By the early 1980s the Cray Blitz had achieved master status; in 1988 the computer Deep Thought became the first computer to defeat a human grandmaster in a tournament; and on 11 May 1997, the IBM Deep Blue computer triumphed over the world chess champion Garry Kasparov, then (and now) the highest rated human player of all time.

The dramatic victory of the Deep Blue over Kasparov would seem to represent an incontrovertible validation of the legitimacy of the computer chess approach to AI (Bloomfield and Vurdubakis, 2008; Hamilton, 2000; Newborn, 2003). The match bore all the hallmarks of what by then had become the standard modus operandi of computer chess research: a carefully staged (and well-publicized) showdown between man and machine; huge cash prizes (US$700,000 for the winner; US$400,000 for second place); and grand claims made about the supposed relationship between chess ability and general intelligence. Kasparov, who had a long history of playing against computers, was perhaps the ideal opponent for Deep Blue. In the years between 1986 and 2005, Kasparov was almost continuously ranked as the world's number one player, a record nearly three times as long as that of his closest rival, Anatoly Karpov. A more worthy opponent for Deep Blue – and a more definitive test of its abilities – could scarcely be imagined. And when, in game six of a six-game match, Deep Blue forced Kasparov to resign after just 19 moves, the aspirations of Herbert Simon and his fellow computer chess enthusiasts appeared to have finally been realized.

But after the initial flurry of sensationalist coverage in the popular press (Kasparov's defeat represented 'The Brain's Last Stand', according to the cover of *Newsweek* on 5 May 1997, and the book jacket of a popular book by computer scientist Monty Newborn (2003) declared it 'not just a triumph, but a rare, pivotal watershed beyond all other triumphs: Orville Wright's first flight, NASA's landing on the moon'), questions began to emerge about what, if anything, this apparently ultimate accomplishment actually signified.

Certainly Deep Blue was a symbol of the remarkable progress made in computer technology over the previous four decades: the 30 processors that formed the heart of the machine were capable of 11,380,000,000 floating point operations per second, making it one of the top 300 supercomputers in the entire world at the time. Each processor contained 16 customized chips designed to perform chess-specific functions such as move generation and position evaluation. Running on these 30 parallel processors (a total of 480 customized chips), the minimax algorithm used by Deep Blue was capable of evaluating 200 million positions per second (which translated into an average search depth of six to eight moves).

But was all this power proof of the existence of true AI? IBM had spent millions of dollars on Deep Blue, a machine that only played a grand total of six games (too few to even gain it an Elo rating) against a single opponent before it was dismantled. In fact, the machine was disassembled immediately after its narrow victory over Garry Kasparov, and its internal workings have never been revealed to the satisfaction of the research community – an important but unintended consequence, perhaps, of the competitive tournament system and the increasing reliance on cash prizes to fund system development. In any case, to many observers, Deep Blue's brute force approach to computer chess – along with its narrowly specialized 'Kasporov Killer' techniques – was too single-minded to suggest any meaningful general intelligence (Aleksander, 2001; Ekbia, 2008). 'My God, I used to think chess required thought', reflected the noted cognitive scientist Douglas Hofstadter in response to the Deep Blue victory: 'Now, I realize it doesn't. It doesn't mean Kasparov isn't a deep thinker, just that you can bypass deep thinking in playing chess, the way you can fly without flapping your wings' (quoted in Weber, 1996). In a 1997 response to the Deep Blue victory published in the journal *Science*, John McCarthy, the founding father of both AI and competitive computer chess, publicly lamented the degree to which computer chess had been led astray by the will-o-wisp of tournament victories: 'Computer

chess has developed much as genetics might have if the geneticists had concentrated their efforts starting in 1910 on breeding racing Drosophila. We would have some science, but mainly we would have very fast fruit flies' (McCarthy, 1997: 1518).

At the heart of McCarthy's critique is the perception that, although computer chess was productive in that it encouraged constant experimentation, it produced no new theories – either about human cognitive processes or theoretical computer science. In their influential 1958 paper arguing for the centrality of chess in the study of machine-based intelligence, Herbert Simon and Allen Newell had stressed that it was essential not only that the computer made good moves, but that it made them for the right reasons. Computer chess was, for Simon and Newell, valuable only to the degree that it represented a 'deliberate attempt to simulate human thought processes' (Newell et al., 1958). This lofty goal was soon abandoned in the quest to build stronger tournament performers. Other than making incremental improvements to the minimax algorithm, computer chess failed to deliver on its larger promise as a tool for exploring the underlying mechanisms of human intelligence. The sociologist Harry Collins (2010: 108) has called the triumph of chess-playing computers a 'hollow victory', in that it caused AI to stray from its original focus on directly modeling human intelligence. According to these and other critics, the lofty original goals of AI, which were to mimic general intelligence, had been increasingly narrowed by the discipline's investment in computer chess, and in particular an approach to chess that emphasized deep searching through a decision tree using the minimax algorithm (Kasparov, 2010; Nareyek, 2004). And so, even as chess-playing computers were reaching the highest levels of human accomplishment, the choice of chess began to be seen as limiting rather than productive (Hedberg, 1997; Horgan, 1999; Munakata, 1996)

The problem with chess, however, was not so much the game itself, but rather the particular way in which the game had come to be defined by the minimax algorithm. The way in which a minimax-based machine plays chess is not at all like the way a human plays chess; in many respects, the two are playing an entirely different game. If anything, the brute-force approach to computer chess highlighted the growing divide between AI and the human cognitive sciences. A growing body of research on human chess players indicated that human players rarely thought ahead more than one or two moves, relying instead on perception, pattern recognition, and the use of heuristics. Chess, as it was played by humans, turned out to be an even more complex cognitive activity than was imagined by the early artificial researchers (Wagner and Scurrah, 1971). As a result, computer chess came to be seen as increasingly distinct from human chess (Peterson, 1997; Westphal et al., 2002; Searle, 1999). Computers played computer chess, and did it quite well, but chess as played by computers was not a game of much general interest. In terms of advancing a larger research agenda, chess – or at least chess as understood by the vast majority of chess-playing computers – was increasingly considered a dead-end for AI (Levinson et al., 1991; Schank, 1991) Mikhail Donsky and Jonathan Schaeffer went so far as to refer to the rise to dominance of the minimax approach as AI's 'fall from grace', suggesting that it was 'unfortunate that computer chess was given such a powerful idea so early in its formative stages' (Schaeffer and Donskoy, 1989: 3). While blind searching through a decision tree might have been easy to implement, and while it scaled well with improvements in hardware and performed well in chess tournaments, it had limited applicability to other problem domains. The power of minimax had channeled research energy in computer chess into increasingly

narrow problem-specific solutions, to the point that Deep Blue computer has sometimes been described not so much as a chess-playing computer, but a playing-chess-against-Garry Kasparov computer.

For the growing number of critics of computer chess as a research agenda, the failure of chess to produce new theories of either computation or cognition suggested that chess was not, in fact, the drosophila that AI had hoped for. In recent years this dissatisfaction with the theoretical productivity of chess has led to renewed interest in a search for new experimental organisms. Of particular interest are Asian games such as Go, which are generally resistant to brute-force, Type-A solutions. Even the top rated Go-playing computers currently play at only the level of an advanced amateur, and the high level of branching in the decision tree for Go make minimax-based approaches largely irrelevant, even when combined with the most sophisticated techniques for alpha–beta pruning. Some researchers hope that alternative games such as Go might prove a more suitable experimental technology for producing theories in AI, which are more applicable to more general problem domains (Hendler, 2006; Müller, 2002). Others, such as Rodney Brooks, have argued that AI's obsession with games – and not any particular game – is the more fundamental problem, and that the discipline needs to shift its focus from the manipulation of symbolic systems towards interaction with the physical environment (Brooks, 1990).

Conclusions

If, as many AI researchers appeared to believe, the primary measure of an experimental organism was its ability to produce fundamental theory, then chess was probably not the drosophila of AI. Despite the impressive productivity of the computer chess researchers, the research agenda that computer chess encouraged was simply too narrow to be sustainable. It was as if drosophila-based genetics research had never advanced beyond the mapping of the drosophila chromosome. Chromosome mapping was, of course, an important contribution made by the drosophilists to genetics research, but as mapping techniques became increasingly routine, interest in drosophila stagnated. It was only with the introduction of new wild varieties of drosophila into the laboratory, and the migration of the drosophilists out of it, that the drosophila was reinvented as an experimental technology for investigating population genetics. Computer chess had no such second act. Although attempts were made to introduce variation into the game (by changing the rules slightly to discourage brute-force approaches), computer chess continued to pursue the very narrow goals defined almost solely in terms of tournament victories (and, consequentially, dominated by brute-force search algorithms such as the minimax).

On the other hand, as historians of science well know, the significance of even the real drosophila is not so narrowly reductionist. Certainly one reason why *Drosophila melanogaster* was so central to the history of genetics was its role in the production of new theoretical insights – but equally important were the ways in which its lifecycle and natural history fit so conveniently into the social and cultural landscape of the contemporary genetics laboratory. As Robert Kohler's work clearly reveals, the science and the organism were mutually constitutive: just as genetics as a science was reoriented around the theories and

methodologies most well-suited to the biology of the drosophila, drosophila as a species was reconfigured as a technology for producing new knowledge about complex physiological or biological systems. And in the same way that genetic scientists transformed the wild and highly variable fruit fly into the standardized organism *Drosophila melanogaster*, physically reconstructing it to conform to the fundamental principles of genetic mapping (Kohler, 1994: 78), so too did AI researchers redefine how chess was played, and what it meant to be an 'intelligent' chess player. In this respect, the parallels between chess and drosophila are impossible to ignore. The history of AI, and of the cognitive sciences more broadly, is incomprehensible without reference to its primary experimental technology.

The shaping influence of computer chess on AI, cognitive science, and a series of related decision sciences is, of course, of immediate concern to historians of those disciplines. The more general lesson to be drawn from this story is methodological, not historical, and is more widely applicable. The most lasting insight of the work on drosophila in the history of science has been to reveal the close relationship between theory and practice, between researcher and subject, between organism and technology. The goal of this paper has been to explore similar relationships within the history of computing. It is a constant temptation when contemplating the pervasive presence of computers and computer-based technologies in our modern society, to regard them in the terms defined by the academic discipline of computer science. To a computer scientist, what is essential about a computer is that it is programmable (Mahoney, 2002). What is important is the software, not the machine, and software itself is generally seen as being uniquely, and almost infinitely, protean. Unlike traditional technologies, which need to be demolished or disassembled before than can be rebuilt or replaced, software can be rewritten using only a keyboard. As Lawrence Lessig (1999) has suggested, within the virtual worlds contained within a computer, code is the ultimate law. The fundamental literary nature of computer programming – software is, after all, in its most basic incarnation simply a series of written instructions – suggests that software systems are effectively ideas made incarnate, mere abstractions that can readily be modified to suit a changing social, technical, and intellectual environment. 'The programmer, like the poet, works only slightly removed from pure-thought stuff', famously declared the computer scientist Frederick Brooks: 'He builds his castles in the air, from air, creating by exertion of the imagination' (Brooks, 1975: 7).

But in working software systems, which include the heterogeneous environment in which computer code is necessarily situated, it is often impossible to isolate the artifact from its social, economic, and political context. Despite the fact that the material costs associated with building software are low (in comparison with traditional, physical systems), the degree to which software is embedded in larger socio-technical systems makes starting from scratch almost impossible. To the degree that writing software does resemble literary production, the product is less an original poem than a palimpsest. This was clearly true of the practitioners of computer chess: the durability of the minimax algorithm wedded the discipline to a very specific, and ultimately very narrow, technological trajectory. Despite the ease with which software systems can, in theory, be readily rewritten, breaking free from this established trajectory proved extraordinarily difficult. The emphasis on tournament play and spectacular man-versus-machine confrontations, which did much to generate enthusiasm for the discipline, also encouraged computer chess researchers to privilege

constant and incremental improvements (Bramer, 1978; Robinson, 1979). This meant an increasing investment in Type-A strategies built around the minimax algorithm and alpha–beta pruning. No other technique could reliably deliver tournament victories. Minimax foreclosed other avenues of research, such as Type-B strategies that more closely mimicked the way in which human beings played chess. And although these Type-B strategies were more representative and more generalizable approaches to planning and cognition, they rarely produced machines that won tournaments. In theory computer chess researchers could pursue any number of software solutions to the problem of modeling human intelligence; in practice, they almost invariably settled on minimax.

It is the extraordinary durability of the minimax algorithm that makes the history of chess as the drosophila of AI relevant and interesting to the larger history of science and technology. Despite their seemingly intangibility, algorithms are anything but ephemeral or immaterial. Like all technological inventions, they are fundamentally human (and social) constructions, and as such embody and enable specific values, agendas, and possibilities. As the practice of science comes to rely more and more on the use of computers and computer-based technologies, the history of software will become as much a part of the history of modern science as instruments, laboratories, published papers, and social practices. It is essential, therefore, that we develop the tools and methodologies for studying software that incorporate an appropriate level of historical, sociological, and technological sophistication.

Notes

1. Kronrod would ultimately lose the directorship of the Institute because of complaints by physicists that he was squandering computer resources.
2. Although using science citation tracking is out-of-fashion in Science and Technology Studies as a measure of the influence of ideas, in this case some basic figures seem relevant. According to Neil Charness, the paper has been cited in the literature more than 981 times since 1973. A scientific paper is generally considered a classic if it is cited more than 200 times. By that standard, the Simon paper is an absolute blockbuster. See Charness (1992).
3. The Knight's Tour required a player to 'tour' a single knight around the entire board, touching each position exactly one time.
4. AM Turing, letter to Jack Good, 18 September 1948, King's College Cambridge Archive.
5. For example, a specific series of games played between Bobby Fisher and Boris Spassky in 1972 could be codified via notation in the generalized opening theory known as the Poisoned Pawn Variation of the Najdorf Sicilian.

References

Aleksander I (2001) *How to Build a Mind: Toward Machines with Imagination*. New York: Columbia University Press.

Ashby WR (1952) Can a mechanical chess-player outplay its designer? *British Journal for the Philosophy of Science* 3(9): 44–57.

Babbage C (1864) *Passages from the Life of a Philosopher*. London: Longman, Roberts, & Green.

Berkeley EC (1949) *Giant Brains; or, Machines that Think*. New York: Wiley.

Bernstein A and de V Roberts M (1958) Computer v chess-player. *Scientific American* 198: 96–105.

Bloomfield BP and Vurdubakis T (2008) IBM's chess players: On AI and its supplements. *Information Society* 24(2): 69–82.

Bramer M (1978) Advances in computer chess. *SIGART Bulletin* 67: 14.
Brooks FP (1975) *The Mythical Man-Month: Essays on Software Engineering*. New York: Addison-Wesley.
Brooks R (1990) Elephants don't play chess. *Robotics and Autonomous Systems* 6: 3–15.
Brown DJH (1979) Computer games – Is Go harder than chess? *Personal Computing* 3(12): 81–83.
Brudno M (2000) Competitions, controversies, and computer chess. Unpublished paper, Department of Computer Science, University of Toronto. Available at: www.cs.toronto.edu/~brudno/essays/cchess.pdf (accessed 15 July 2011).
Burian RM (1993) How the choice of experimental organism matters: Epistemological reflections on an aspect of biological practice. *Journal of the History of Biology* 26(2): 351–367.
Campbell-Kelly M (2007) The history of the history of software. *Annals of the History of Computing, IEEE* 29(4): 40–51.
Charness N (1992) The impact of chess on cognitive science. *Psychological Research* 54: 4–9.
Coles LS (1994) Computer chess: The drosophila of AI. *AI Expert* 9: 25–32.
Collins HM (1990) *Artificial Experts: Social Knowledge and Intelligent Machines*. Cambridge, MA: MIT Press.
Collins HM (2010) *Tacit and Explicit Knowledge*. Chicago: University of Chicago Press.
Crevier D (1992) *AI: The Tumultuous History of the Search for Artificial Intelligence*. New York: Basic Books.
Dowsey S (1973) Go and the computer. *Go Review* 13(3): 72–74.
Dreyfus H (1965) *Alchemy and Artificial Intelligence*. RAND Paper P-3244. Santa Monica, CA: RAND Corporation.
Dreyfus H (1992) Response to Collins, artificial experts. *Social Studies of Science* 22(4): 717–726.
Ekbia H (2008) *Artificial Dreams: The Quest for Non-Biological Intelligence*. Cambridge: Cambridge University Press.
Ensmenger N (2009) Software as history embodied. *Annals of the History of Computing, IEEE* 31(1): 88–91.
Ensmenger N (2010) *The Computer Boys Take Over: Computers, Programmers, and the Politics of Technical Expertise*. Cambridge, MA: MIT Press.
Franchi S (2005) Chess, games, and flies. *Essays in Philosophy* 6(1). Available at: http://commons.pacificu.edu/eip/vol6/iss1/6 (accessed 21 July 2011).
Franchi S and Güzeldere G (eds) (2005) *Mechanical Bodies, Computational Minds: Artificial Intelligence from Automata to Cyborgs*. Cambridge, MA: MIT Press.
Greenblatt R, Eastlake D III and Crocker S (1967) The Greenblatt chess program. Proceedings of the AfiPs Fall Joint Computer Conference, Vol. 31: 801–810.
Guterl F (1996) Silicon gambit. *Discover*, June, pp. 49–56.
Hamilton SN (2000) The last chess game: Computers, media events, and the production of spectacular intelligence. *Canadian Review of American Studies* 30(3): 339.
Hashagen U, Keil-Slawik R and Norberg AL (2002) *History of Computing – Software Issues*. New York: Springer-Verlag.
Hedberg S (1997) Smart games: Beyond the deep blue horizon. *IEEE Expert* 12(4): 15–18.
Hendler J (2006) Computers play chess; humans play go. *IEEE Intelligent Systems* 21: 2–3.
Hofstadter D (2005) Moore's Law, artificial evolution, and the fate of humanity. In: Booker L, Forrest S, Mitchell M and Riolo R (eds) *Perspectives on Adaptation in Natural and Artificial Systems*. New York: Oxford University Press, 163–198.
Horgan J (1999) The undiscovered mind: How the human brain defies replication, medication, and explanation. *Psychological Science* 10(6):470–474.
Husbands P, Holland O and Wheeler M (eds) (2008) *The Mechanical Mind in History*. Cambridge, MA: MIT Press.

Jay R (2000) The automaton chess player, the invisible girl, and the telephone. *Jay's Journal of Anomalies* 4(4).

Jesiek B (2006) The sociotechnical boundaries of hardware and software: A humpty-dumpty history. *Bulletin of Science, Technology & Society* 26(6): 497–509.

Johnson D (2007) *White King and Red Queen: How the Cold War Was Fought on the Chessboard*. London: Atlantic Books.

Kasparov G (2007) *How Life Imitates Chess: Making the Right Moves, from the Board to the Boardroom*. New York: Bloomsbury.

Kasparov G (2010) The chess master and the computer. *The New York Review of Books* 57(2) (11 February). Available at: www.nybooks.com/articles/archives/2010/feb/11/the-chess-master-and-the-computer/ (accessed 21 July 2011).

Kohler R (1994) *Lords of the Fly: Drosophila Genetics and the Experimental Life*. Chicago: University of Chicago Press.

Koschmann T (1996) Of Hubert Dreyfus and dead horses: Some thoughts on Dreyfus' *What Computers Still Can't Do*. *Artificial Intelligence* 80(1): 129–141.

Law J (1987) Technology and heterogeneous engineering: The case of the Portuguese expansion. In: Bijker WE, Hughes, TP and Pinch T (eds) *The Social Construction of Technical Systems: New Directions in the Sociology and History of Technology*. Cambridge, MA: MIT Press, 111–134.

Lessig L (1999) *Code, and Other Laws of Cyberspace*. New York: Basic Books.

Levinson R, Hsiung-Hsu F, Schaeffer J, Marsland TA and Wilkins DE (1991) The current and future role of chess in artificial intelligence and machine learning research. In: *Proceedings of the 1991 International Joint Conference on Artificial Intelligence*, pp. 547–552.

Levy DNL (1976) *Chess and Computers*. Woodland Hills, CA: Computer Science Press.

Lohr R (2007) *The Secrets of the Chess Machine*. New York: Fig Tree.

McCarthy J (1997) AI as sport. *Science* 276(5318): 1518–1519.

McCorduck P (1979) *Machines Who Think: A Personal Inquiry into the History and Prospects of Artificial Intelligence*. San Francisco, CA: W.H. Freeman.

McDermott D (1976) Artificial intelligence meets natural stupidity. *ACM SIGART Newsletter*, pp. 4–9. Available at: www.neurosecurity.com/articles/AI/AIMeetsNaturalStupidity.pdf (accessed 23 July 2011).

Mahoney M (2002) Software: The self-programming machine. In: Akera A and Nebeker F (eds) *From 0 to 1: An Authoritative History of Modern Computing*. New York: Oxford University Press, 91–100.

Mahoney MS (2008) What makes the history of software hard. *Annals of the History of Computing, IEEE* 30(3): 8–18.

Marsland TA (1991) Computer chess and search. Technical Report TR 91-10, Department of Computing Science, University of Alberta.

Mitman G and Fausto-Sterling A (1992) Whatever happened to planaria? C.M. Child and the physiology of inheritance. In: Clark AE and Fujimura JH (eds) *The Right Tools for the Job: At Work in Twentieth-Century Life Science*. Princeton, NJ: Princeton University Press, 172–197.

Müller M (2002) Computer Go. *Artificial Intelligence* 124: 145–179.

Munakata T (1996) Thoughts on Deep Blue vs. Kasparov. *Communications of the ACM* 39(7): 91–92.

Nareyek A (2004) Computer games – boon or bane for AI research? Computer Science Department, Carnegie Mellon University. Available at: www.ai-center.com/publications/nareyek-ki04.pdf (accessed 23 July 2011).

New York Times (1875) Babbage and a chess automaton. *New York Times*, 21 November, p. 3.

Newborn M (2003) *Deep Blue: An Artificial Intelligence Milestone*. New York: Springer.

Newell A, Shaw J and Simon H (1958) Chess-playing programs and the problem of complexity. *IBM Journal of Research and Development* 2(4): 320–335.

Nilsson (1998) *Artificial Intelligence: A New Synthesis*. San Francisco, CA: Morgan Kaufmann Publishers.
O'Shields J (1965). Selection of EDP personnel. *Personnel Journal* 44(9): 472–474.
Panek LL (1976) 'Maelzel's chess-player', Poe's first detective mistake. *American Literature* 48(3): 370–372.
Papert S (1968) The artificial intelligence of Hubert L. Dreyfus: A budget of fallacies. AI Memo 154, Computer Science and Artificial Intelligence Lab, MIT. Available at: http://hdl.handle.net/1721.1/6084 (accessed 21 July 2011).
Peterson I (1983) Playing chess bit by bit. *Science News* 124(15): 236–237.
Peterson I (1997) Silicon champions of the game. *Science News* 152(5): 76–78.
Poe EA (1836) Maelzel's chess player. *Southern Literary Messenger* 2:318–326.
Randell B (1982) From analytical engine to electronic digital computer: The contributions of Ludgate, Torres, and Bush. *Annals of the History of Computing* 4(4): 327–341.
Rasskin-Gutman D (2009) *Chess Metaphors: Artificial Intelligence and the Human Mind*. Cambridge, MA: MIT Press.
Riskin J (ed.) (2007) *Genesis Redux: Essays in the History and Philosophy of Artificial Life*. Chicago: University of Chicago Press.
Robinson AL (1979) Tournament competition fuels computer chess. *Science* 204(4400): 1396–1398.
Ross P (2006) The expert mind. *Scientific American* 295(2): 64–71.
Russell S and Norvig P (2009) *Artificial Intelligence: A Modern Approach*, third edition. New York: Prentice-Hall.
Schaeffer J (2001) A gamut of games. *AI Magazine* 22(3): 29–46.
Schaeffer J and Donskoy M (1989) Perspectives on falling from grace. *Journal of the International Computer Chess Association* 12(3): 259–268.
Schaffer S (1996) Babbage's dancer and the impressarios of mechanism. In: Spufford F and Uglow J (eds) *Cultural Babbage: Time, Technology and Invention*. London: Faber, 53–80.
Schank R (1991) Where's the AI? *AI Magazine* 12(4): 38–49.
Schonberg H (1981, September 27) Cold war in the world of chess. *New York Times*. Available at www.nytimes.com/1981/09/27/magazine/cold-war-in-the-world-of-chess.html (accessed 31 August 2011).
Searle J (1999) I married a computer. *New York Review of Books*, 8 April, 34–38.
Shannon C (1950) Programming a computer for playing chess. *Philosophical Magazine* 41(314) 256–75.
Shenk D (2006) *The Immortal Game: A History of Chess or How 32 Carved Pieces on a Board Illuminated Our Understanding of War, Art, Science, and the Human Brain*. New York: Doubleday.
Simon H and Chase W (1973) Skill in chess. *American Scientist* 61: 393–403.
Simon H, Simon HA, Schaeffer J and Schaeffer J (1992) The game of chess. In: Aumann RJ and Hart S (eds) *Handbook of Game Theory with Economic Applications, Volume I of Handbooks in Economics*. London: Elsevier Science, 1–17.
Standage T (2002) *The Turk: The Life and Times of the Famous Eighteenth-Century Chess-Playing Machine*. New York: Walker.
Steiner G (1971) A death of kings. In: Steiner G, *Extraterritorial*. Cambridge, MA: Athaneum Press, 47–57.
Sussman M (1999) Performing the intelligent machine: Deception and enchantment in the life of the automaton chess player. *TDR (1988–)* 43(3): 81–96.
Turing AM (1946) Proposed electronic calculator (National Physical Laboratory report, 1946). In: Carpenter BE and Doran RW (eds) *A.M. Turing's ACE Report of 1946 and Other Papers*. Cambridge, MA: MIT Press.
Turing AM (1950) Computing machinery and intelligence. *Mind* 49: 433–460.

Turing AM (1953) Digital computers applied to games. In: Bowden BV (ed.) *Faster than Thought: A Symposium on Digital Computing Machines*. London: Pitman, 286–297.
Van den Herik HJ, Uiterwijk JWHM and van Rijswijck J (2002) Games solved: Now and in the future. *Artificial Intelligence* 134: 277–311.
Von Neumann J (1958) *The Computer and the Brain*. New Haven, CT: Yale University Press.
Voskuhl A (2007) Producing objects, producing texts: Accounts of android automata in late 18th-century Europe. *Studies in History and Philosophy of Science* 38: 422–444.
Wagner D and Scurrah M (1971) Some characteristics of human problem-solving in chess. *Cognitive Psychology* 2(4): 454–478.
Wall Street Journal (1978) Chess diplomacy. *Wall Street Journal*, 14 November, p. 20.
Waters CK (2004) What was classical genetics? *Studies in History and Philosophy of Science* 35: 783–809.
Weber B (1996) Mean chess-playing machine tears at meaning of thought. *New York Times*, 19 February. Available at: www.rci.rutgers.edu/~cfs/472_html/Intro/NYT_Intro/ChessMatch/MeanChessPlaying.html (accessed 24 July 2011).
Wells S and Reed C (2005) A drosophila for computational dialectics. *Proceedings of the International Conference on Autonomous Agents and Multi-Agent Systems* (doi 10.1145/1082473.1082724): 1263–1264.
Westphal J, Hitterdale L, Cahn SM, Verhaegh M, Stevens CW, Machan TR and Yates S (2002) Letters to the editor. *Proceedings and Addresses of the American Philosophical Association* 75(5): 173–182.
Yood C (2003) Attack of the giant brains. *Online Research: Penn State* 24(3). Available at: www.rps.psu.edu/0309/brains.html (accessed 23 July 2001).

Biographical note

Nathan Ensmenger is an assistant professor in the School of Information at the University of Texas at Austin and the author of *The Computer Boys Take Over: Computers, Programmers, and the Politics of Technical Expertise* (MIT Press, 2010). His other publications include articles on gender in computing, the history of software, and healthcare informatics. His current project focuses on the development and use of computerized decision tools in medicine, public policy, and finance.