CHESS AS THE DROSOPHILA OF AI

John McCarthy
Computer Science Department
Stanford University
Stanford, CA 94305
jmc@cs.stanford.edu
http://www-formal.stanford.edu/jmc/

1997 Jun 16, 1:40 p.m.

Abstract

This is a condensed version of an invited presentation, delivered on May 31, 1989 to the Conference of the Canadian Information Processing Society on its meeting on May 29 to June 2, 1989 in Edmonton, Alberta, Canada. The article was mainly written by the editors of the ICCA journal, based on their notes on my lecture. The lively style is theirs. However, I added considerable material that appeared only in the printed version.

1 INTRODUCTION: THE FRUITFLY ON THE FLY

The phrase *Chess is the Drosophila of Artificial Intelligence* is not my own invention. To the best of my recollection, I owe it to the late Alexander Kronrod, who may have used it as a defense against physicists when they complained that he used so much of their precious computer time in a *mere* chess match. This was probably in 1966 during the year long telegraph match between Stanford University and the Institute of Theoretical and Experimental Physics in Moscow. Let me say that my thoughts were not entirely clear when choosing this topic and I am therefore very glad to be addressing a

homogeneous audience of "chess-types". I will expand the subject slightly so as to allow discussion of the *Drosophilas* of AI in a more general sense.

One of the pressures I was under came from people in computer science. They sometimes urged me to tackle topics of practical importance and to concentrate on experimental and theoretical work in precisely these applicable areas, as opposed to a backwater such as computer chess. This echoes a remark that might have been made to Thomas Hunt Morgan in 1910: "Elephants are far more useful than fruitflies and who wants better fruitflies? So why don't you do your work in genetics on elephants rather than on fruitflies?" To which Morgan could have countered: "It takes no more than two weeks to breed a generation of fruitflies, you can keep thousands of them in a bottle and they are cheap to feed".

With the choice of *Drosophila* out of the way—it is a superb experimental animal—I can now address my remarks to several groups.

First, consider the *philosophers*. Some of them hold that intelligence is impossible for a machine to achieve, even in principle. They should consider the chess programs' gradual and sometimes not so gradual climb in performance. The best of them has just beaten its first Grandmaster. Who knows, the best of them may beat the World Champion in 1992.

My next few remarks concern the *computer-chess community*. By and large, they come in two flavors: sportsmen and businessmen—makers of commercial chess machines. Neither group is primarily motivated to produce scientific papers explaining how the results were achieved and how others might build upon them for further improvement. Due to these factors, computer chess has only partly succeeded in playing a *Drosophila* role and I must adjure the community that computer chess should be more scientific: publication is what we need.

To *computer scientists in general* I have only one wish to express: let there be more of these *Drosophila*-like experiments, let us create some more specific examples!

Fourthly, I hold that *government and industry* often take an extremely short range view towards AI and towards computer science in general. For example, in 1971, the Defence Advanced Research Project Agency (DARPA) cancelled a robotics project they had been sponsoring, because quick results couldn't be promised. DARPA continued sponsoring the same workers in a different field. 18 years later, it is clear that this switch of objective was accompanied by a decline in quality. The 1970 and 1971 papers are still being cited in the literature (including the applications literature) as significant

advances; not so for their 1973-1975 publications (since 1971, nine two-years periods have passed). I claim that for applications to be developed, basic research is essential and must be supported in whatever form, be it done by companies or government or by universities, which after all specialize in basics. See (McCarthy, 1983)

2 THE FIRST DROSOPHILA

Switching my focus slightly, let me consider Expert Systems. Citing my own paper *Some Expert Systems Need Common Sense* (McCarthy, 1983), I recall a fairly detailed critique of the technology of expert systems as it was at the time. I addressed the fact that it was possible for the MYCIN expert system to be proficient at diagnosing bacterial infections without even knowing what a bacterium, a doctor or a hospital is. In fact, MYCIN did not even consider time-dependent processes (not allowing even the question: the patient yesterday died, what do you recommend for today's patient with the same symptoms?). Despite many such crippling limitations, we now possess a useful expert systems technology. Indeed many expert systems do reason about the results of events.

Consider the position of diagram 2 from Hans Berliner's (1974) Ph.D. dissertation. He used it to show the need for a *global strategy*. He argued that a chess program conducting a 5-ply search would play 1. Ke3, because this preserves "control of the center", and the program can't see how to win. He then gave a summary of the findings during the tree search for which I refer to his thesis. He concluded by saying "Most of the problems of chess are tactical (immediate) problems and for this reason, the lack of global ideas is frequently obscured in today's (1974) programs". I think (1989) that's still true. (1997 note: Deep Blue has problems with global ideas).

Returning to 1989, I asked some participants of the Sixth World Computer Chess Championship, on the morning before my lecture, to submit this problem to their programs. All programs succeeded, needing from three to twenty seconds, and all did it by brute force considerably mitigated by transposition tables. Transposition tables are particularly effective for this problem, because only the kings and the white pawn at f6 have any moves, and therefore only a small set of positions arise in the search.

Maybe some would justify programs using brute force by claiming: "Humans use brute force too, except that they are unaware of the massive par-

allelism built into their brains."

However, it seems to me that humans use a much simpler thought process that doesn't involve brute force. First, the player forms the goal of going around the pawns to the left and capturing the pawn on e6. He notes that the BK must stay where it can prevent the WP on f6 from queening and this allows the WK to reach c5 without interference. Then unless the BK is on d7, the WK advances to d6, but if the BK is on d7, the WK can advance to d6 in 3 moves anyway. With the WK on d6, it can capture at e6 unless the BK is on f7, but then he can capture it in 2 moves anyway. Thus W captures the pawn in at most 10 moves. If the WK were initially on h3, capturing the pawn would take 14 moves, but solving the problem would be no more difficult for a human chess player.

The interesting part for me is the 5 (or 9) move advance to c5. The key is determining that specific BK moves are irrelevant provided it stays within the set {d7,d8,e8,f7,f8,g8,h7,h8} of squares in order to prevent the WP at f6 from queening.

Maybe some parallelism is involved in the human pattern recognition that suggests the goal of going around the pawn formation. It probably isn't required, but human chess playing might be the beneficiary of parallel pattern recognition capability evolved in support of vision. Parallelism certainly isn't required to justify the plan.

3 Condensed Move Trees

The above solution to the problem can be formalized using *condensed move trees*. The formalization should make clearer what human intelligence does and what AI will have to do. Maybe it will also help the writers of chess programs. We consider two versions.

In the first version, moves are still represented by edges, but a set of moves can correspond to a single edge and a set of positions, equivalent from the point of view of the strategy being evaluated, corresponds to a node. The size of the tree is enormously reduced compared to the actual move trees. In the second version an edge corresponds to a sequence of moves, but then we have to allow branching from the middle of an edge. Still greater economy is achieved.

Figure 2 shows a move tree. The labels on its edges are propositions about the moves rather than the moves themselves. One should probably

attach propositions about the positions to the nodes. In exchange for a more complex concept, we get an enormously smaller tree.

However, we can abstract further as in Figure 3. Here the double arrow denotes W following the path to c5 **and** B staying within the eight squares {d7,d8,e8,f7,f8,g8,h7,h8}. The one arrow represents the behavior of both players. Since B can deviate from this behavior, the figure shows a mult-tailed arrow leaving the edge followed by the pawn move f6-f7 that refutes it. We don't carry Figure 3 beyond getting the WK to c5, because the rest is exactly the same as in Figure 2. We call trees like that of Figure 3 *edge branching trees*, since edges can have branches as well as nodes. This representation is even more compact and doesn't even bother to note how many moves there are in the sequence. I think it corresponds more closely to the human reasoning.

Vladimir Lifschitz (at lunch) suggested that instead of an edge-branching tree we put a node in the middle from which moving the BK out of A branches. This gives a conventional tree at the cost of making the edges before and after the node of indefinite length, since the same node has to represent whatever point black deviates. Figure 4 represents Lifschitz's idea.

There needs to be a mathematical theory of the correspondence between move trees and these more abstract condensed move trees.

4 Remarks

Although chess is a game in which the players move alternately, we see that in this position it is worthwhile to regard them as moving simultaneously. Fortunately for chess players, simultaneous action is the more common in life, and we can adapt our non-chess habits in order to understand this position. To be able to model consecutive moves as processes involving concurrent actions is a challenging task which may make the programs better.

In the proper sense of using chess as a *Drosophila*, the first requirement is a satisfactory theory of concurrent actions. This is a theory of how to achieve goals with the opponent moving concurrently rather than in alternation with you. Such a theory, as it happens, is relevant to the naval battle-management project: in the world of naval warfare, ships *do* move concurrently. However, this is one more project which DARPA is currently (1989) dropping. Very likely in the course of this naval-management project there are some useful scientific results. Maybe these results will be published in an abstract way

serving as a foundation for further work. But I am concerned about the more likely outcome that whatever has been done in this area will have been done half-heartedly, or merely done, to the extent it is applicable to this specific naval battle-management problem and, even so, abandoned half-way through.

Suppose we wish to investigate the true relation between the position of Diagram 1 and the human strategy given above. Is the strategy valid in analogous positions? Assume we add two Pawns on the board, a black Pawn on a7 and a white Pawn on a6 as depicted in Diagram 5, to create one such analogy.

Following the same strategy as outlined above, the conclusion is that it does not work. In particular, we are unable to play 9. Kb6. Unfortunately, White can win anyway. Berliner gave in Diagram 1 the addition of two Pawns on b4 (white) and b5 (black) as an example, which turned the position into a draw. Another fine experiment in relation to the original strategy in Diagram 1 is the addition of white Pawns on c2 or g2 or both. All I can state is that when adding one white Pawn the chess machine Bcbc found the solution quicker than when adding two white Pawns.

5 ANOTHER DROSOPHILA

Another *Drosophila* worth mentioning is not taken from chess. But again, it is one which roused the impatience of the practically-minded people, who were saying, "You theoreticians are wasting too much time on this Yale Shooting Problem". I have a different opinion, namely that the Yale Shooting Problem is a kind of key to understand the practical use of non-monotonic reasoning. It is a problem in *generality*. I would like to have a general database of

common-sense knowledge, such as the general fact about getting somewhere is by going, and the general fact that when people die they remain dead. I would like to see that all sorts of trivial facts of that kind would be available to *any* computer program to use in a general form. In monotonic reasoning, deduction follows from premise without the possibility of refutation, and, indeed, with absolute necessity. However, monotonicity is not a realistic model of the world. In the well-known problem of the missionaries and cannibals, the problem solver is bound to retract conclusions in the fact of certan facts contradicting his all-too-simple assumptions.

In terms of the Yale Shooting Problem the details of which are irrelevant here, what does the sequence of loading, waiting and shooting mean? What is added by the common-sense-knowledge database? We may state by definition that (1) loading loads and (2) shooting kills if loaded; but what about waiting? Moreover, we may define (3) one remains alive unless shooting happens and similarly (4) a gun remans loaded unless shooting happens. Hence, during the process of loading, waiting and shooting, the gun came unloaded. However, a gun having run out of ammunition cannot be assured to stay unloaded: a spent gun will not always remain spent.

Non-monotonicity as in the above is typical of human reasoning, It involves the need to modify plans or strategies seen later to embody unwarrantable assumptions.

In more formal terms, what we would like to have is a scheme starting from

$$A \vdash P \qquad (\text{AsyntacticallynecessitatesP})$$

$$A \subset B \qquad (\text{AisimpliedbyB})$$

therefore

$$B \vdash P \qquad (\text{i.e., BsyntacticallynecessitatesP})$$

and the latter, alternatively written

$$[A \subset B] \supset [Th(A) \subset Th(B)]$$

which is to say that if A is implied by B this implies that any theorem derived from A is implied by any theorem derived from B.

This neat formulation unfortunately is not always achievable by a human being or a smart-robot.

This is merely one of too many examples, and as a research area in computer reasoning, non-monotonic problems and common-sense-knowledge

problems have hardly been explored. To my opinion, if a bird can fly the fact need not to be mentioned, because that would be part of the common-sense-knowledge database. By contrast, if a bird cannot fly the fact **must** be mentioned and the fact should take precedence over the database.

6 Yet another Drosophila

Returning to chess, over the years we have seen that all chess programs rely on evaluaton functions. Since Nimzovich, we possess rules intelligible for human beings and along the lines of which a game will proceed adequately. Hence, next to the rules of adequacy we have the value of a position (or of its successors) produced by the evaluation function. A twofold problem now arises:

(1) this value may not correspond to what a human being does in such a position;

(2) this value may not correspond to what a program **must** do in such a position.

Chess players always compare related positions, they never evaluate them independently. Chess programs evaluate each position in almost complete disregard of any evaluations done previously. Therefore, chess programs, were they able to take a leaf out of the human book should be much simpler than they are now by relying on prior evaluations of related positions. I think that the models as described by my former student, now named Barbara Liskov-Hubermann (then named Barbara Jane Huberman), is very close to the model of Capablanca in his endgame book. Of course, the model has only been proved to be appropriate for KRK, KBBK and KBNK, and admittedly, the ideas are not applicable for the full game, but perhaps we can gain something out of the comparison of positions with the predictates *better* and *worse*.

Figure 1 shows an example of a forcing tree (Huberman, 1968, p.5)

Figure 1: Example of a forcing tree.

"The program has the move in p; it must make a move leading to a position q judged *better* than p for every sequence of moves by the opposition. Each iteration of the program will produce a tree like this; several iterations will be required to reach checkmate." The difficulty in this procedure comes in deciding what pattern features of the positions are important.

Huberman's first hypothesis was: the model used in the program is a good representation of the abstract model assumed by chess books. It has resulted in insights about translation from book problem-solving methods into computer-program heuristics. The relation between methods and heuristics are embodied in the functions *better* and *worse*.

Huberman's secnd hypothesis was that the information in the chess books is sufficient for the definitions of *better* and *worse*. Speaking for myself, I consider a combination of the right attributes, the idea of comparing positions instead of evaluating them, and the introduction of *better* and *worse* predicates to be yet another true *Drosophila* in computer-science research.

7 The Fourth Drosophila

As a fourth *Drosophila* I would like to mention the research on Computer Go. Let me first state that there is a fundamental difference with chess. Comparing the playing strength of the best Computer-Go program with the playing strength of the Go World Champion I estimate such a program to be twenty levels behind the Champion, whereas I opine that the best among computer-chess programs is "only" three levels behind Kasparov.[1][This issue provides you with a direct match between Kasparov and Deep Thought, suggesting that a few levels should be added.-Eds.] The deepest issue about

[1]*

intelligence is its very foundation; it seems to be based on some collection of intellectual mechanisms, however embodied. Of some we know they exist and they are in the collection, from others we do not know they exist, let alone whether they are in the collection or not. (1997 note: The best *Go* programs are still only 6th kyu. Compared to chess, this is below a level that would get an official rating at all.

In human chess, a position is considered as a whole; it is not divided into parts (sometimes, an example of a part of the board helps illustrating a concept, but it is not vital). In *Go*, separation is absolutely vital. Searching all legal moves in a *Go* game tree must be discarded by their sheer number of ramificaions and the consequences of their combinatorial explosion after a few plies only.

Therefore, a *Go* program ought first to identify small groups of stones and conduct searches and evaluations regarding each of these groups separately. The research of *Go* programs is still in its infancy, but we shall see that in order to bring the *Go* programs at a level comparable with nowaday chess programs investigations of a totally different kind than used in computer chess are needed. [1997 note: This still seems to be true.]

8 Conclusions

A plethora of conclusions can be drawn about the impact of the four *Drosophilas* listed. As research develops, the richness of conclusions possible has the habit of proliferating. I will refrain from anticipating but, speaking personally, I expect the more fruitful results to emphasize the ideas of "local" estimations on groups (*Go*), of configurations (*better* or *worse*), of databases with common-sense knowledge, and of distinguishing phases in plans (as in dealing with endgame positions). Some combinations of these, I believe, are a potentially fertile basis for the much-needed global strategies for solving a general problem of any sort.

8. REFERENCES

Berliner, H.J. (1974). *Chess as Problem Solving: The Development of a Tactics Analyser*, Ph.D. thesis, Carnegie-Mellon University, Computer Science Department.

Huberman, B.J. (1968). *A Program to play chess end games*, Technical Report, No. CS 106, Ph.D. thesis, Stanford University, Computer Science Department.

McCarthy, J. (1983). Some Expert Systems Need Common Sense.
Nelson, H.L. (1985). Hash Tables in Cray Blitz, *ICCA Journal*, Vol. 8, No. 1, pp. 3-13.

Let me recall a few more examples, starting with a (slightly amended) chess position from Berliner's Ph.D. thesis (1974, his Fig. 1.7), here depicted as Diagram 1. However, only those applications are feasible that require only those intellectual mechanisms that are well understood and readily implemented. Moreover, sentences in the expert system databases are usually correct in the limited context of the application implemented.

Berliner used the position of Diagram 2 to show that there is a need for a *global* strategy. At that time (1974) he argued that any chess program conducting a 5-ply search would play something like 1. Ke3 because it controls the center. He then gave a summary of the findings during the tree search for which I refer to his thesis. He concluded by stating: "Most of the problems in chess are tactical (immediate) problems and for this reason, the lack of global ideas is frequently obscured in today's [1974] programs".

Let us now revert to 1989. For the position of Diagram 2, I have asked some participants of the Sixth World Computer Chess Championship, on the morning before my lecture, to submit this problem to their programs. All programs succeeded, needing from three to twenty seconds, and all did it by brute force. They did so according to a well-established argument for the use of brute force in chess, which runs: "Human beings use brute force too, except that they are unaware of the massive parallelism built into their brains". To human beings, the argument by which White is due to win is simple and irrefutable: move the white King via a3 to b5 (first phase), then maneuver it via tempo and opposition in the appropriate configuration to c6 (WK on c6, BK on e6, BTM) (second phase), and finally capture the d5 Pawn (third phase). For instance: 1. Kd2 Kd7 2. Kc2 Kc6 3. Kb2 Kb6 4. Ka3 Kc6 (after 4, ... Kb5 White wins by 5. e6, for human beings the ideas are similar) 5. Kb4 Kb6 6. Ka4 Kc6 7. Ka5 Ke7 8. Kb5 Kd7 9. Kb6 Ke7 10. Kc7 Ke6 11. Kc6 Ke7 12. Kxd5 and White wins easily. Computer programs, incapabale of understanding this, do a tree search in which possible white moves, black ripostes and possible white counterripostes are considered in all the detail necessary. If you do this in the abstract, using the full game tree, no current program would be able to solve the problem. When Belle, in the beginning of the 1980s the most powerful chess engine, was tried, with its transposition tables not functioning, even that program did not find the solution, because it was overwhelmed by the work to be done [transpositions

tables are a well-known trick saving many searches (cf. Nelson, 1985)]. But even with the transposition tables and all other labor-saving devices installed, the computer does enormously more work than is necessary.

What is the crux? It is to formulate a good theory on the conjoint actions of both Kings. The first question to solve is, of course, to see how you can achieve your goal (by arriving at b5; by reaching the appropriate configuration with the WK on c6; by capturing the d5 Pawn). The second question is to find a path to b5 to achieve this goal even though your opponent is also moving.

Here I would like to stress that Chess is a game in which players move alternately; in this it is rather exceptional, because life contains many competitive processes in which competing players move simultaneously rather than alternately. It turns out that it is many times advantageous to think about a competitive process (such as the chess position of Diagram 1), in which the players are usually moving alternately, as a process in which they are moving simultaneously. From the problem-solving point of view this is a gain. Any human being contemplating, even at a glance, the configuration of Diagram 1 sees at once that the important fact is not precisely how Black may be wandering around in the eight squares e6, e7, e8, d8, c6, c7, c8, the only human question is whether he is somewhere within them or is not. So, we are exonerated from generating the move tree, while a computer must do so time and time again. Nor do we need any transposition tables, physical or mental in this particular problem, because it is easily proved that no position can ever repeat. Thus, the human brain does not even have to invoke whatever the massive paraalelism might be built into it.

/@steam.stanford.edu:/u/ftp/jmc/drosophila.tex: begun Sat Jun 14 12:24:26 1997, latexed June 16, 1997 at 1:40 p.m.