C O M P U T E R

C H E S S

edited
by

A G Bell

Atlas Computer Laboratory
Chilton
Didcot
Berkshire
OX11 0QY

October 1973

Proceedings of a One-Day Meeting
on

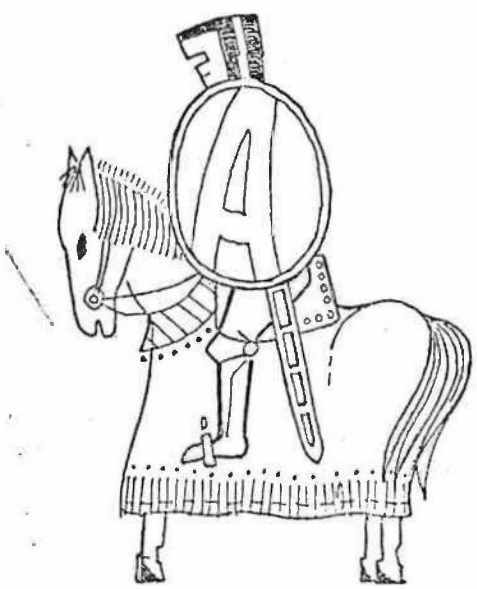
CHESS PLAYING BY COMPUTER

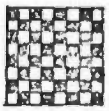
organised by
the Atlas Computer Laboratory
of the Science Research Council
on 21 May 1973

ACKNOWLEDGEMENT

The knight was drawn
by Philip G Crane

F O R E W O R D





C O N T E N T S

		Page
	Foreword	i
1	Computer Chess Experiments A G BELL	1
2	A Simple Working Model P KENT	15
3	Psychology and Computer Chess A H BOND	29
4	Mathematical Relations in Chess R H ATKIN and I H WITTEN	37
5	A Knowledge Based Program to play Chess End-Games S TAN	81
6	Observations R MALIK	89
7	Descriptor Index A H BOND	95
8	References	103
9	List of Participants	113

FOREWORD

"Why have a conference on computer chess?" This question was put to me a number of times on the day and I gave a number of different answers. This is because any answer must depend on an assessment of how much I think the person asking the question knows about the subject. To write this foreword is therefore difficult. I am now trying to explain to everyone without any of the feedback that is so necessary in conversation and so useful when lecturing.

Let me first kill two myths, both perpetrated by the Lighthill report:-

(i) "It is interesting to consider the result of all this work some twenty-five years after the researches aimed at chess-playing programs began: unfortunately these results are discouraging. The best programs play chess of only 'experienced amateur' standard characteristic of county club players in England. Chess masters beat them easily."

The implication here is that the only reason people write chess programs is to actually play chess and the discouraging result is that they cannot beat Bobby Fischer. There are other reasons (some of which are given in these collected papers) and it seems unfair to define the only encouraging result as a program that would beat everybody and doubly unfair that nobody is going to get support if that is their declared aim. And the main reason why? Quite simple, because for the last twenty years 'results are discouraging' etc.

(ii) Apparently quite considerable resources have been devoted to producing an effective program. This is rubbish! Until the end of 1972 there were only two people in this country who had ever earned a living by writing chess programs. One was John Scott, who had just left school at the time, and the other was myself who had just left university. Neither of us cost the country a great deal, indeed I was employed by a Norwegian-Italian with an American grant. I agree that our results were discouraging: John's program did not quite manage to hold its own against Greenblatt, and my program (written in three months in 1962)

has now been translated into only six different computer languages and used as the basis of only eight chess programs (three in America, four in this country and one in Norway).

My point here is that it is unfair to criticise the results of a subject that has never been officially supported or funded. Do not make the error that the Americans or the Russians are any better off; most of the work done in those countries is by people who also beg, borrow and steal computer time. David Slate and Keith Gorlen, co-authors of CHESS 3.5 (the current champion program) wrote it in their spare time having failed to obtain NIH funds.

There are, to my knowledge, only five people at the moment who are paid to write chess programs. Three of them (Gillogly, Berliner and Simon) are at Carnegie-Mellon University; Richard Cichelli at Lehigh, Pa, and Soei Tan at Edinburgh.

The discouraging results are therefore probably due to low funding but the fault still lies with the people who would like to work on computer chess. They rarely give clear reasons (I include myself) why and how they wish to spend money and time on the problem. Why don't they?

Well let's hold a conference and get people together. Find out why and how people want to work on the problem now and in the future. And the result: most people don't want funds! Instead they would like more access to their firm's computer (in their own time) and less persecution from their superiors. Most practitioners like the idea of meeting other 'amateurs' at a conference; they can compare notes and size up the opposition. But they still prefer to work on their own ideas in a small group. In short, the British want, now and in the future, to treat it as a hobby, but a reputable hobby.

So here is an impressive document to enhance the reputation of this hobby; perhaps it should be subtitled 'Teach yourself advanced programming' because most hobbies are concerned with exercising talents and abilities which our normal work does not either permit or encourage.

This point of view is most common amongst computer scientists who have tried to program chess. They will also point out the spin-off in techniques first tried in a game playing experiment; for example, hash tables, directed search, alpha-beta cut-off, catalogues.

Despite these very real successes most people actually dabbling in computer chess (there were 14 people present who had written programs) are, on the whole, reluctant to commit themselves completely to the problem. Perhaps, like Einstein, they are happier in the obscurity of their 'patents' office' where they are not expected to continually 'lay golden eggs'. This is fair enough. But I would like to co-ordinate some of these labours of love. There are a number of problems and experiments on which I would appreciate other people's opinions and I have described some of them in 'Computer Chess Experiments'. Although I agree with Soei Tan that the Turing-Shannon model is probably inadequate I still maintain that it is the only well defined model

that we have and that there are many useful techniques it can be used to investigate, particularly the refutation (or killer) heuristic. This is basically the computer scientists' viewpoint but, in my case, is almost certainly due to the way the subject was first 'imprinted'. I fully appreciate that other people see computer chess very differently but I firmly believe that only a computer scientist can gather together and implement all these different ideas because, in the end, it has got to be tried in a machine and very few people really know how to program; I do not include Botvinnïck.

'Imprinting'? As I mentioned above I wrote my first chess program over ten years ago. I was employed to generate a 'list of legal moves' for any chess position; this generator had to be as fast as possible because the research was into models of evolution using symbio-organisms. It was hoped that they would learn to play chess.

At the time we did consider making the program play a game. I again stress that this was not the main purpose of the research. Without reference to any literature we wrote a Turing-Shannon lookahead (it is a very obvious model) and an evaluation function based purely on mobility. We spent a whole week on this work and the results were discouraging, Even we could beat it, let alone chess masters.

At this point in time the fund ran out and, seeing no future in the subject, I went off to earn a living doing something useful. I was however left with the naive impression that a chess program could be built in three separate pieces, namely:-

(a) list legal moves;
(b) lookahead;
(c) evaluation function.

To 'list legal moves' is no problem, to write a crude lookahead is also well defined and trivial but to construct a successful evaluation function is where it all fouls up. It is a fact that the fewer heuristics in the evaluation function, the more accurate it is, ie capture the Black King is exact; material balance much less accurate and if you worry about pawn structure during search you are looking for a very inaccurate evaluation.

Berliner says in his paper that special heuristics (eg 0-0 early in the game, not moving a piece twice early in the game, advancing pawns during the endgame) are an admission of defeat. I agree. I have never tried to construct a sophisticated evaluation; never tried to express my 'chess knowledge' because the performance becomes extremely difficult to measure or explain. Indeed, many evaluation functions have not so much been designed as been created ad hoc, the programmer has had a 'feeling in his water' and it is impossible to reproduce his results no matter how closely you read his publications or listen to him. I must emphasise the point that science is concerned with repeatable experiments.

I said the problem can be considered in three pieces. This is not true in practice. The crude lookahead is simply unacceptable and, in order to reduce the tree search time, it is necessary to use an evaluation function to prune, back up, order and direct this activity (particularly if alpha-beta cut-off is incorporated) and, even more important, to

know when to stop searching and when to go deeper. So the results are discouraging because nobody really knows how to write accurate evaluation functions. I was, therefore, very interested when I read Ron Atkin's paper 'Multi-dimensional structure in the game of chess'. Here was a mathematician who, with lots of squiggly things and some hard sums, appeared to propound a mathematically reproduceable evaluation function. The missing link? Unfortunately, I could not understand it, so why not get him to talk about it? There were other people who had published work I did not understand, so why not have a conference? If nothing else I might get some idea of what was going on.

The SRC and the Atlas Computer Laboratory were almost embarrassingly helpful (again my impression is that research would be supported if only people would make a clear and committed case). Not all the speakers I wanted were available but, despite appearances, there was a thread in the order of the lectures.

Basically the morning was intended to be hors d'oeuvres. Peter Kent and I agreed that we would merely set the stage (a) to get a relaxed, informal atmosphere and (b) to introduce the subject with a simple working model. We hoped to get people talking and in the right mood for the main course in the afternoon.

The three principal speakers were therefore Alan Bond on psychology, Ron Atkin on the multi-dimensioned approach and Soei Tan on knowledge. Rex Malik very kindly agreed at the last minute to lead a discussion. I again interpret his remarks as an unconcious appreciation (by him) that most people in the audience do not want the responsibility of funds but much prefer the subject as a hobby.

I still believe that successful computer chess will be the first step in the ascent of machine intelligence. I make no hypotheses of how it will be realised but one thing is certain. If you want to practice and improve your ability to program a computer then the subject is similar to Fermat's last theorem; you most probably will fail to produce anything significant but you will learn a hell of a lot about programming and, incidently, psychology, maths, urban development models, epistemology, and the theory of evolution.

I have given a short introduction to each paper. I would emphasise that these are personal observations.

COMPUTER CHESS EXPERIMENTS

by

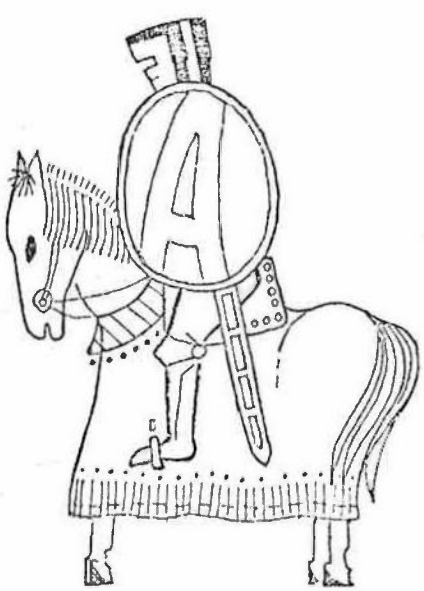
A G Bell

Rutherford Laboratory
Science Research Council
Chilton
Didcot
Berkshire
OX11 0QY

"The first professor ... said perhaps I might wonder to see him employed in a project for improving speculative knowledge by practical and mechanical operations."

- JONATHAN SWIFT

Gulliver's Travels



Editor's Note -

Five finger exercises. The ability to program a computer is a necessary but not sufficient condition for producing a successful chess program. A sound knowledge of modern I/O equipment (particularly interactive graphics) is also necessary but the computer scientist is still dependent on ideas from other fields. Meanwhile he should practise his art.

In the lecture I related the sad fates of a number of pioneers in the field of machine intelligence. These included Raimon Lull, Blaise Pascal, Jonathan Swift, the Spanish captain, Charles Babbage and Alan Turing.

It was to show that, although the fascination of intelligent machinery has a long history, we still have not achieved the first significant step. The analysis and construction of a successful chess machine could be that step.

One reason for these discouraging results is a lack of co-ordination between the different groups and disciplines which dabble in the subject.

The computer is the only machine we have to perform experiments in machine intelligence in general and chess in particular. It is essential to have experience of the strengths and weaknesses of these machines. This is the province of the computer scientist and the necessary co-ordination must come from computer science.

Science is concerned with the measurement of repeatable experiments and application of the results. Computer chess has usually been treated less rigorously, almost an art form, with the emphasis on the computer playing the game and humans gauging its performance.

This paper describes some repeatable experiments for a chess program. The intent is that programs can and should be assessed without them actually playing each other. Of course they should play occasionally but it is an expensive operation and not always conclusive as to which is the better chess program.

Handicaps

When two chess programs play each other with time limits invoked then not only the programs but also the computer/compiler systems are in competition. Alan Bond raised the question as to whether it is possible to isolate the programs performance and ultimately give handicaps to the computer/compilers.

I published an Algol chess algorithm to solve any two move mate problem (Bell, 1970) and have since received correspondence from people who have tried it on a number of machines in at least six languages.

The times obtained for the different computer/compiler systems to solve a two move mate and prove it unique (no cooks) have been interesting. At first I believed that because the algorithm was so specialised its performance on different systems could only give comparitive results to within a factor of two or three. In fact the times for Algol systems agree to within 20% with results obtained by B Wichman who has used a sophisticated technique to compare and analyse the execution performance of over twenty Algol computer/compiler systems (see Computer Journal, February 1972).

Because of the good agreement with Wichman it is my belief that the results for the translations into PL/1 and FORTRAN can give handicap data on the performance of these and other computer/compiler systems. Moreover, because the algorithm is table driven and highly language independent, it can be translated into most computer languages in a matter of days.

The table below gives the comparable results for six powerful modern computers and demonstrates the empirical agreement of the algorithm with Wichman's analyses. The Gibson mix is a measure of the hardware power of a machine. The ICL Atlas Algol is taken as the standard.

Comparable Results for Six Powerful Modern Computers

COMPUTER/ALGOL COMPILER	GIBSON MIX	WICHMAN	CHESS.MATE IN SECS
ICL Atlas /MK1	1.0	1.0	100
B5500 / MK1	0.3	0.5	220
UNIVAC 1108 / (obsolete)	2.0	1.2	90
ICL 1906A / XALT MK5	2.5	3.3	30
CDC 6600 / MK1	4.7	1.1	100

The results for other computer/compilers are:-

Atlas Basic	10 seconds
CDC 6600 Basic	4 seconds
IBM 360/195 in FORTRAN H	4 seconds
IBM 360/195 in PL/1	7 seconds

All times are for the problem in (Bell, 1970). The Univac 1108 and CDC Algol compilers have been rewritten; they now have Wichman figures of 2.3 and 3.0 respectively.

The effort to implement the algorithm in the various languages was:

Algol	about 1 man-day
PL/1	about 4 man-days
FORTRAN	about 2 man-days
Atlas Basic	about 10 man-days
CDC Basic	about 10 man-days

From the table we see that, for example, a chess program in Atlas Algol should be given 100/4=25 times longer to consider a move if we wish to compare it with a chess program in CDC Basic.

Programs not using the algorithm can be adjusted to solve two move mates. This will mainly measure their power in listing legal moves but useful handicaps could result because the conventional program spends the majority of its time in this activity; philosophical programs would not be so easy to handicap.

Two and three move mate

The algorithm mentioned in the previous section is crude. To obtain consistent handicaps it should not be altered however, it is open to great improvement and it is instructive to discuss the inefficiences of the algorithm and so introduce a significant programming principle - the principle of 'refutation'.

The algorithm is table driven (the most powerful of computer languages). One important feature is that the 64 squares of the board are not scanned but an integer array is consulted. This array, 'piece', contains the number of white (black) men on the board and their actual locations. For example, in the position

								64
					BP			56
								48
					WP			40
				BP		BP		32
			BP	WP		WP		24
BP	BP	BP	WP			WP	WB	16
BQ	BK	BB			WN	WR	WK	8

then white 'piece' is:

9	6	7	8	12	15	16	21	23	38

← Direction of scan

and black 'piece' is:-

10	1	2	3	9	10	11	20	29	31	54

In this problem the mating sequence is:-

W1 P-B6
B1 BB*P
W2 N*B checkmate

but to discover that it is checkmate the program must actually capture the king. It does this as follows:-

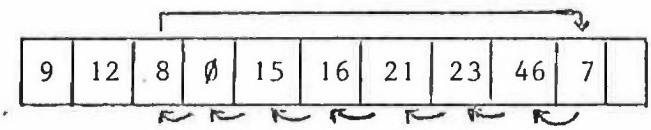
W2 N*B 'am I checking the king?' (to avoid stalemate).

The list of 'piece' is scanned backwards so it finds the rook check first. It now continues:-

B2 K-B8
W3 R*K

Black has no further alternatives? at the B2 or the B1 ply so the problem is solved. The problem introduces the concept of 'refutation move and/or man', in this case the rook. Gillogly calls this the 'killer heuristic' and shows it to be relevant to actual computer chess play (Gillogly, 1972).

The solution of the mate problem can be speeded up. When white discovered the move R*K at the stalemate check it could have re-ordered the white 'piece' array thus:-



The rook would now have its moves listed first and in isolation, the actual capture of the king at W3 can then be detected without listing the moves of any other white men. But this misses the really important gain. Black will backtrack to ply B1. Now in this case it does not have another alternative but normally black would. However the alternatives are rarely significant and the same refutation move and man will usually checkmate at move W2.

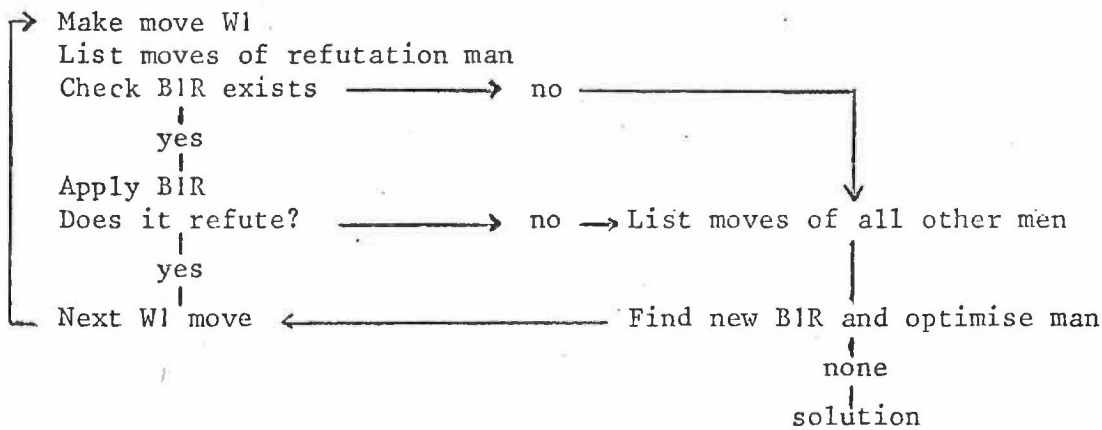
The fact that the order of white's 'piece' for W3 can be heuristically optimised from the stalemate check is applicable to the previous plies B1, W2 and B2. Here is a simple experiment to verify this statement.

Put in a two move mate problem. Print white's first move W1 and now print all black's responses B1. Eventually black will make a move B1R which refutes W1, the algorithm will cutoff and white will try another W1. So we have:-

W1a
B1a B1b B1c B1R
W1b
B1a' B1b' B1c' B1R
etc

The result of this experiment is that the black move B1R which refutes the present W1 is usually (60%) the same B1R which refuted the immediately previous W1. Even more significant is the refutation man (75%), very often the king who just moves away.

Let us assume that we modify the algorithm to preserve the refutation move B1R. Also assume we can check it exists for the next W1 in zero time. This means that the timings given in the previous section can be reduced by 60% ie 100 seconds becomes 40 seconds. By similar argument W2 and B2 can have their refutation moves optimised and we obtain a limit of improvement → 93% ie 100 seconds in Atlas Algol could drop to about 7 seconds and basic programs could be less than a second. A further bonus is that the program is more capable to giving an 'appreciation' of the problem; reporting white's threats and black's replies. Unfortunately, the full reduction cannot be realised, one reason is that we must check if the move B1R still exists for the next W1 etc. The best we can do is to only list the moves of the man which generated the previous B1R. So we have the following flow:-



We are now spending most of the time in 'list moves of all other men'. However the reordering algorithm (previous page) does optimise the finding of the next B1R. Note that full implementation requires a different 'piece' array for B1, W2, B2, stalemate and W3. When a solution has been found the order of the men in the various 'piece' arrays will give a further appreciation of the problem by the computer.

Two move mate problems are too short to accurately measure these improvements. Consequently three move mates have been used to test them. Preliminary results indicate that, in Algol on the ICL 1906A, the time for a two move mate can be reduced from 30 seconds to about 6 seconds, ie 5 times faster, and a three move mate takes about 50 times longer, ie about 5 minutes.

Apart from Gillogly, other people have 'discovered' refutation; in particular Richard Cichelli of Pennsylvania. In that all 'killer'

or 'refutation' heuristics are similar the above implementation is the same as Cichelli's and Gillogly's. However the cost effectiveness of refutation can vary widely. The McCarthy-Gillogly killer associates a particular move with a particular position. Gillogly says that this does not pay for the overheads.

My implementation, associating and ordering particular men with the current area of the search tree, is much less specific; more hits but less accurate. Cichelli's work is somewhere between these two extremes.

The big failure of my implementation is that when the hoped for refutation does not exist or fails to work then I list all the moves of all the remaining men. It would (or should) be more efficient to only list the moves of the next man in 'piece'. However this will require a major change to the program.

The fact that the program will not then immediately check the legality of the opponent's previous move should not matter. It is prepared to do so; either the refutation is effective against an illegal move or the normal cutoff will occur eventually.

Another improvementwould be along the lines of COKO III (Cooper-Kozdrowicki, 1973) which concentrates on white moves W1, W2 and W3 that can capture the king ('attack paths') and consequently narrows the search. This again accelerates the solution of mate problems unless Zugzwang is involved.

Evaluation functions

It is in the evaluation function rather than any other part of the conventional chess program that scientific measurement is most lacking. Here the programmer must express what he considers to be relevant to chess; his chess 'knowledge' is programmed into the computer, an admission of defeat according to (Berliner, 1970). The usual test of the evaluation function is to play the program.

Here is an experiment. Obtain about 500 positions in chess and have them examined and assessed by a panel of experts. For each position the panel gives an ordering, from best to worst, of all the moves worth consideration, ie the non-Fischer set. This not only allows us to compare programs without them actually playing but if Fischer would do the test we can compare champion v human and champion v program.

This is not quite fair. Every time a chess program has to make a move it behaves like it has never seen the previous moves (unless it does something like the reordering of the pieces discussed in the previous section). Fischer, presented with 500 unconnected positions, would probably not be as dominant over a computer as when he actually plays a game.

Note that one does not have to write a complete program to test an evaluation function. If it is expressed as an algorithm in an acceptable language, Algol or FORTRAN, there is no reason why this should not be tested by someone else's well written, modular program.

Now let us consider perhaps the simplest evaluation function which has any relevance to chess ie material and mobility evaluation. This could be a criterion for other functions.

The evaluation function is:-

(a) for a given position list all the captures first (material);
(b) all remaining moves are graded by the resulting mobility ratio, ie make the move and then calculate (how many moves you have/ how many moves the opponent has) in the new position.

To investigate how relevant this function is to actual chess play I took the selected games ot ten chess masters described in Golombek's book 'The Game Chess'. The masters are Anderssen, Morphy, Blackburne, Steinitz, Tarrasch, Lasker, Capablanca, Nimzovitch, Alekhine and Botwinnik. In the ten games the masters were faced with 336 positions. Now we are not going to get full agreement on the opening moves they chose, nevertheless for 95% of the cases the move chosen by the master was one of the top 16 moves selected by the simple evaluation function. Can your chess program do better? It not throw it away.

Another feature of this evaluation function is that it appears capable of distinguishing between conventional players and revolutionary players. Conventional players, like Anderssen and Capablanca, are more in accord with the function than players like Reti and Reshevsky, but this is the province of game theory not game playing.

Is it possible to prove that a given evaluation function is incapable of winning against best play? This is a neglected approach but it does have possibilities. For example:-

(a) If the program can capture then do so, ie like the no-huffing rule in checkers. It is possible to disprove this algorithm, however the opponent must offer some important captures to control the game.

(b) If you always have more moves than your opponent then you must win. Obviously true? He has no moves when he loses his king but is stalemate avoidable? Also how long can white maintain more moves than black? P-K4 gives white an initial ratio of 30 moves to black's 20 moves. One unverified result is that white can maintain a mobility advantage over black for the first 20 moves from the P-K4 opening. Note that if white does have a forced win and there is a limit to retaining the greater mobility then white's best play must include a 'mobility gambit'. How long does your evaluation function keep ahead against all black's responses?

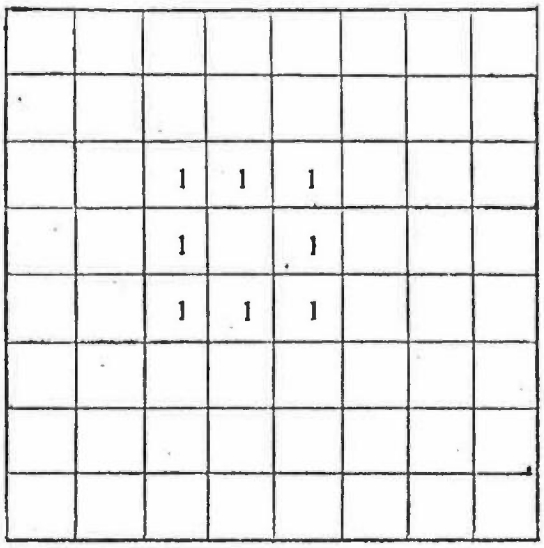
(c) It is not possible to play losing chess by reversing the signs of parameters in an evaluation function, eg give black the greater mobility? Try playing 'giveaway' checkers; two kings against one win in either version of the game. Samuel suffered from this misconception.

Repetition

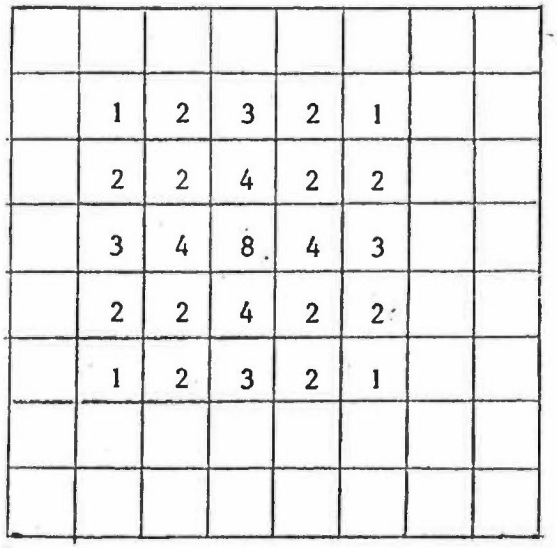
One of the reasons chess playing programs have not progressed further than the strong amateur level is that they waste time recreating and

reanalysing exactly the same position in the lookahead. This is not so apparent in games which computers can play at master level: Kalah, Gomoku and checkers. In these games the pieces (and therefore the positions) do not usually cycle; the only troublemakers are checker kings, relatively rare pieces. This is not the case in chess, all the pieces (as distinct from pawns) can cycle. Humans do not generate these loops but computers spend most of their time in pointless repetition, even in the improved two-move mate algorithm already discussed.

Consider the chess king. If we look ahead 1, 2 and 3 moves we find the following histograms of the king's terminal position:-



I



II

1	3	6	7	6	3	1	
3	6	12	12	12	6	3	
6	12	27	27	27	12	6	
7	12	27	24	27	12	7	
6	12	27	27	27	12	6	
3	6	12	12	12	6	3	
1	3	6	7	6	3	1	

III

I : Total 8 Distinct 8 New 8

II : Total 64 Distinct 25 New 16

III : Total 512 Distinct 49 New 33

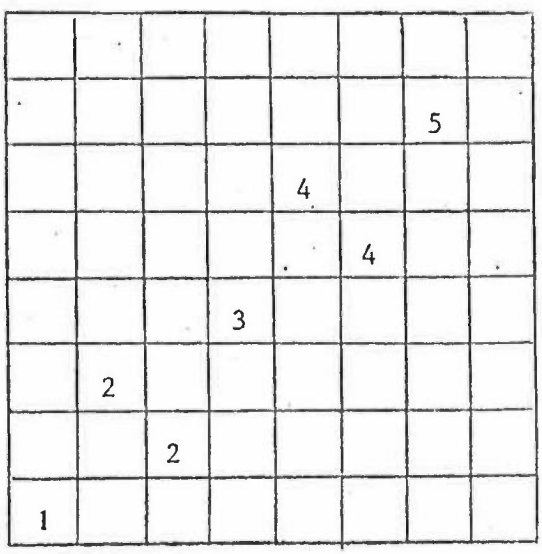
Make no mistake, a crude program playing the simple K, R v K ending will generate similar rubbish. Of course the actual path can be important sometimes; whether castling is still possible and en passant capture exists.

To quickly check for repetition of a position (and hence save re-evaluation) would apparently be easy on a CDC STAR. The word length is 64 bits; equivalence of two words containing the two positions of the chessmen would indicate possible repetition. A closer check would then be necessary and the immense complications of full recognition, cataloguing, garbage collection etc become apparent.

Now humans do not appear to work in this way, we know that (W1-B1-W2-B2) is usually equivalent to (W2-B1-W1-B2) and do not generate the final position; we recognise similar paths not similar final positions.

There is no simple answer to this problem, the intent is to spotlight the time wasted by chess programs in their evaluation and re-evaluation of positions. It seems that almost any attempt to recognise or suppress repetition at or before the evaluation level must be highly rewarding in terms of saving time - but how rewarding?

Here is an experiment. Starting in a corner, how many different ways can a knight tour the board visiting each square only once and returning to the starting square at the 64th move? The answer is not known but any person attempting to solve it will quickly realise how repetitous the knight's path can be. For example:-



There are four ways the knight can get to square 5 but all four must get the same answer from symmetry?

A similar problem which has been solved might give some indication of the possible savings. The problem is how many different ways a fly can crawl round a five-dimensional cube, visiting each corner once only and returning to the starting corner at the 32nd move. An abortive attempt was discussed by Martin Gardner in Scientific American, August 1972. Professor Ronald Read had estimated the solution

would require ten years by computer. Donald Russell, a computer scientist, obtained the answer 906,545,760 in five minutes! The trick was to treat the problem like a game, ie make legal moves with a 32 ply look-ahead but similar paths were recognised and ignored. This crude recognition of paths resulted in a program running one million times faster than a qualified estimate. The benefits to chess programs of recognising equivalence of moves will not be so great but even ten times faster can be significant when machines like the CDC STAR, about 100 times faster than Atlas, become available to chess programmers.
See Tan's paper for other ways of approaching this problem.

Unscientific myths

A dangerous myth has arisen from the fact that chess was designed by humans to be used by humans. Examples of this myth are statements like:-

'Chess is a paradigm of human mind'.
'Master play will require a program "modelled on human thought processes" '.
'The program must "make use of essentially the same methods as those used by men" '. (Women's Lib: Please contact I J Good).
'The program must be given "chess knowledge" '.

Such statements have a polarising effect on research. It allows philosophers, phsychologists, geneticists, chess masters etc to waste hours of machine time and then pronounce on the problem as too difficult. Computer scientists rarely have the opportunity, yet surely the less information a good program requires from us to attempt a problem the quicker and better it can attempt a variety of problems. It involves us with less work and eliminates misconceptions on our part, allowing the computer more freedom and efficiency to do its own thing, ie mini-max, alpha-beta and refutation. It seems obvious that if we concentrate more on programming technique and produce a chess program which only 'knows' legal moves and only plays to master level then this is more useful and adaptable than a highly specialised chess model which could play at a higher international master level with the high probability that we still could not understand how it worked.

But to return to the computer doing 'it's own thing' with a human activity. Consider the Morse code. Like chess it was invented by a human for humans to use, surely this must have some effect on how a computer should handle Morse code?

It is my experience that non-professional people who (claim to) know Morse code do so in a variety of inefficient visual and phonetic mnemonics. Professional Morse coders and Bobby Fischer are not included; people whose expertise has developed to such sub-concious levels that they are no longer aware of how they do it. Laymen, confronted by the laborious virtuosity of the non-professionals, are impressed; obviously the problem is difficult. You may suffer from this impression. Here is an experiment. How long would it take you to learn Morse code? Define learn as a permanent memory of how to decode a Morse message written on paper; speed is not important. If a person knows binary, ie 'thinks' like a computer then the answer is about five minutes. Hopefully, you are surprised. A human activity can be learnt and applied more

effectively by humans if they behave like a computer. Maybe aircraft do not have to flap their wings either.

Here is a pseudo Algol program to decipher Morse code, the input is assumed to be a bar (-), a dot (.) or a space ( ) to delimit the letters:-

```
    N:=0;
A:  if its a dot then N:=2*N+1 else
    if its a bar then N:=2*N+2 else

    print and clear(letter [N] );
    goto A;
```

The 'array letter [1:28]' contains the following sequence

ETIANMSURWDKGOHVF L PJBXCYZQ

which a human must commit to memory. This is possible in five minutes but it is left to the reader to see how the algorithm works. A final word on paradigms:

... -.-. .... -- .- .-. .- -.. .. --. -- ... !

Conclusion

The previous sections have discussed some repeatable experiments. They are illustrations of how a limited but more scientific approach to chess programs could be made and are intended more as stimulating examples in advanced programming than experiments to be slavishly emulated.

Computer chess is a rich field of research for programming technique, games have been the original test bed of many important developments eg hash tables, alpha-beta cutoff, pattern recognition, M and N procedure, information retrieval studies. It is important to measure and report the efficiency of new techniques.

In this way we could approach, step by measured step, a master chess model. In the meantime the techniques that are developed must be a valuable fallout, far more important than knowing if white does have a forced win.

Finally a word of encouragement. Compared to the man-decades that have been spent on developing computer languages we have only spent a few man-years on chess programs. Lord Rutherford once wrote to Niels Bohr that 'you cannot expect to solve the whole problem of modern physics in a few years. So be cheerful over the fact that there is still a great deal to do.'

2

A SIMPLE WORKING MODEL

by

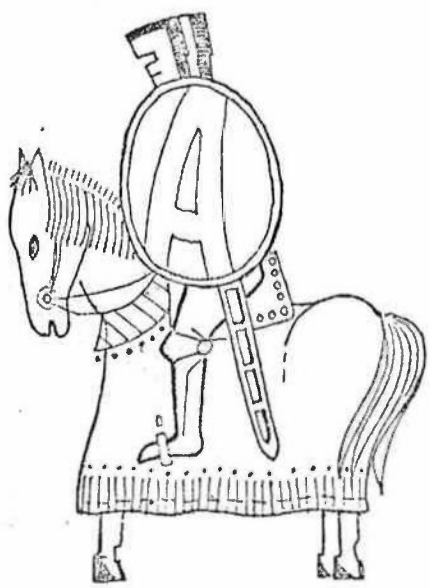
P Kent

Atlas Computer Laboratory
Science Research Council
Chilton
Didcot
Berkshire
OX11 0QY

"And take man's vaunted power of calculation. Have we not engines which can do all manner of sums more quickly and correctly than we can? In fact, wherever precision is required man flies to the machine at once, as far preferable to himself."

- SAMUEL BUTLER

Erewhon



Editor's Note -

A very simple working model. The program is deliberately constrained to answer within a few seconds and the chosen move is computed almost entirely from a shallow search and evaluation function based on threat and counter threat to pieces and squares.

By limiting the depth of search to two plies it is easier to understand why an evaluation function contains insufficient "chess knowledge". There appears to be little proof that deeper searching must necessarily improve performance.

However the paper is mainly intended to introduce the classic Turing-Shannon model.

The program I am going to describe is based on the '2 move mate' problem solving program written by A G Bell (Bell, 1970).

As this has been published in the Computer Journal I will not describe the move generating routine but will instead describe the development of the position evaluation function and some of the problems encountered during that development. The program is written entirely in Algol, originally for the Atlas Computer. It is probably the only program to have been moved from one machine to another machine in a different language.

Initially the program based its evaluation solely on the number of moves available to each side. The greater the difference in the number of moves available to one side over the other, the better the position. This evaluation function has been suggested before, and although it works surprisingly well, it does have a number of faults:-

(i) No value is given to an undeveloped piece, such as a rook, in the early part of a game.

(ii) The queen tends to be developed far too soon. (Unless one uses a library of openings this problem is very difficult to overcome.) The value of keeping the queen in reserve for a few moves is something that is learned by experience and cannot easily be programmed in.

To overcome the problem of evaluating undeveloped pieces, it was necessary to take account of two separate values for each piece on the board.

First its immediate value (which depended on its position) and second its potential value (which usually remained constant throughout the game). This potential value is related to the expected mean value of the squares controlled throughout a game. These potential values are approximately in the ratio P=1: N=3: B=3: R=5: Q=9.

The number of moves available to each side had initially been adopted as the evaluation because of its ease of computation.

Although it had worked surprisingly well there did not seem to be any logical reason why it should.

One did not need a lot of moves, one good one was all that was necessary, and a choice of 50 moves was little more likely to provide this than a choice of 25.

I then realised that there was a close correlation between the number of moves available and the number of squares threatened.

I therefore modified the program to compute the number of squares controlled. A square is considered to be controlled if one has more threats to that square than the opponent. One should also take account of the value of the pieces threatening a square. A pawn would for example have more effective control than a queen. Strictly speaking a square is only controlled if, during a complete sequence of swaps on that square, the difference in the total value of the pieces swapped off is never negative.

To speed the program up, I evaluated all positions one ply deep, selected the 'best' six or so, re-ordered these so that the 'best' were tried first and then looked one ply deeper, using alpha-beta cutoff to avoid unnecessary work (Samuel, 1967). To get the effect of a deeper look ahead while minimizing the extra computing time, I gave a value for threats to pieces. If I had just moved, the values of the threats were as follows. All threats to my pieces were worth half the value of the pieces to my opponent, and all threats to my opponent's pieces were worth one third of the value of the pieces to me.

I tried several values for these threats between one and one quarter of the piece values but half and one third seemed most reasonable.

Essentially, the value is based on the likelihood of a capture. If we have one piece en prise, one move may save it, but if we have two pieces en prise, we are unlikely to be able to save them both or capture their equivalent value.

All these threats could be computed quite cheaply from two arrays containing the number of threats I had on each square of the board and the number of threats my opponent had. At this point the program captured if you gave it the chance, moved a piece if threatened, but generally displayed no imagination.

The computer operators used to play the program at night and write sarcastic comments on the output after winning in 15 or so moves.

I then decided to try building some sort of strategy into the program by giving the squares different values. Initially the ratios were 3 for the central four squares, 2 for the next ring of twelve and 1 for all the remainder.

The next night the best player among the operators tried playing the program, expecting to win with his usual ease. The program opened with the rather aggressive if unsound Blackmar gambit:-

1. P-Q4, P-Q4
2. P-K4, P*P
3. N-QB3.

First Winning Game (Blackmar gambit)

	W (Atlas)	B (C.H.)
1.	P-Q4	P-Q4
2.	P-K4!?	P*P
3.	N-QB3	N-KB3
4.	B-KN5	N-KN5
5.	B-QN5ch	P-QB3
5a	B-QB4	Q-N3
6.	B-KB4	N-Q2
7.	Q*N	Q*QP?
8.	Q-KB5	Q*B
9.	O-O-O	P-K3
10.	Q-KN5	P-KB3
11.	Q-QR5	P-QN3
12.	Q-KR5ch	P-KN3
13.	Q-KR3	P-K4
14.	P-QN3	B-QR6ch
15.	K-QN1	Q*N?
16.	Q*Q	P*B
17.	Q*QBP	R-QN1
18.	Q-QB7	O-O
19.	R-Q5	P-K6
20.	N-KB3	P*P
21.	R-Q1	R-K1
22.	P-QN4	B-QN2?
23.	R*N	B*N
24.	P*B	QR-QB1
25.	Q-QN7	R-K8?
26.	Q*Rch	R-K1
27.	Q*R MATE	

It then proceeded to develop all its pieces fairly rapidly, castled queen side, doubled its rooks on the open queen file and stormed down the board using both rooks and the queen, ending the game with a check mate by its queen on the 8th rank and its rook on the 7th. The comment on the output was 'well it seems to work now'. It is true that the player had made several errors during the game, such as giving pieces away, but prior to this modification he had always been able to recover such losses with little difficulty. For the first time the program seemed to have developed a purpose.

From then on the operators played more carefully and demonstrated a number of weaknesses in the program. Some are not easy to overcome. There was a very definite inability to cope with advancing pawns, no danger was seen until the pawn reached the 7th rank and was about to queen, at which point it could well be too late. To overcome this problem I created new tables for black and white to give the value of a pawn, and the value of a threat to a pawn, on any square of the board. These values increased as the pawn advanced. This encouraged the program to move up its own pawns and to attack its opponent's advanced pawns. One could also fiddle the table to force the program to open in a particular way. For example, by giving the pawn in QB2 a large negative value one could force it to use the English opening P-QB4, one that it would not normally value very highly, in spite of what Petrosian or Spassky might think.

Another problem more difficult to overcome is the classic failure of searching to a fixed depth (Turing, 1953). If the program finds a potentially bad position at the full depth of its search, it cannot search deeper for a refutation and can only search wider. If, as in my program, the width of search is also limited, it is often unable to find a sensible reply.

As a result it adopts a policy of 'sufficient unto the move is the evil thereof' and will do anything to avoid the 'fatal' move. The program will put off the apparently fatal move by an irrelevant check or an attack on a queen, even if the checking piece can be taken and the original threat remains. A good example of this occurred before the advancing pawn problem was corrected. The program (white) had a won game but its opponent had pushed a pawn through to the seventh rank to reach the following position after move 24.

BLACK

WHITE

		K	N				
	P	P			P		
							N
	P						P
P		*P*					
	P	*K*			*P*	P	*P*
			R				

The program continued:-

25.	R*Nch	K*R
26.	N*Pch	K-K2
27.	N-Q6	P*N
28.	P-N4	P-N8=Q

This ridiculous continuation was simply due to the fact that the move R-KN1 was not placed in the top few moves when evaluated at level 1. After all to do so the program would have to give up a threat on a knight for one on a pawn, and also give up control of an entire central file for control of the KN2 square. With the complete queen's file open there were far too many other moves worth considering first, eg:-

R*Nch
R-Q5
R-Q7

or even:-

R-Q4
R-Q6

and N*P fills up the buffer of six moves. With no sensible move in the buffer it could only put off the fatal pawn queening move and hence lost a winning position. This problem of searching to fixed depth can also arise with irrelevant threats to the king or queen by pieces that can be easily captured or avoided.

The program tends to give castling a rather low priority unless the king is vulnerable or the rook's new file is already open. Knowing that castling is potentially a good move, I have had to encourage it by adding a number of points for this move.

To summarize the present position. The program evaluates all the positions on the first ply, selects the best n, orders these n moves and then evaluates at the second ply using alpha-beta cutoff. The position evaluation function uses the following factors:-

	K	Q	R	B	N
Piece Value	-	324	180	108	90
Your Threats	90	162	90	54	51
My Threats	90	108	60	36	34

Square Values

4	Centre	22
12	Next	12
48	Others	6

cf (Berliner, 1970)

Pawn Tables

Rank	2	3	4	5	6	7
Q Pawn	36	36	36	48	54	66
K Pawn	24	36	36	48	54	66
QB Pawn	4	18	18	30	48	66
The Rest	18	18	18	30	48	66

Your Pawn Threats

Rank	2	3	4	5	6	7
Q Pawn	18	18	18	24	27	33
K Pawn	12	18	18	24	27	33
The Rest	9	9	9	15	24	33

My Pawn Threats

Rank	2	3	4	5	6	7
Q Pawn	12	12	12	16	18	22
K Pawn	8	12	12	16	18	22
The Rest	6	6	6	10	16	22

Also

Your EP pawn threat = 9
My EP pawn threat = 6
castling = 25
stalemate value = 0
checkmate value = 10000

Testing the move selector

In order to test the move selection and ordering routine, I collected statistics on over 100 positions.

I take the move selected at level 2 and find what its position is in the complete ordered list at level 1. If the selection is good all the best moves at level 2 should be near the top of the list at level 1. The results were as follows:-

Position	No of Occurrences	Cumulative %
1	40	32.8
2	24	52.5
3	12	62.3
4	11	71.3
5	16	84.4
6	4	87.7
7	0	87.7
8	4	91.0
9	1	91.8
10	1	92.6
11	0	92.6
12	0	92.6
13	3	95.1
14	1	95.9
15	-	95.9
16	-	95.9
17	-	95.9
18	-	95.9
19	-	95.9
20	-	95.9
21	2	97.5
22	-	97.5
23	-	97.5
24	2	99.1
25	1	100

Thus over 90% of the final moves appear in the first eight selected at level 1. This seems quite a good distribution of selections if it were not for the long tail.

One would like to be able to cut off the search at width eight but some key moves occasionally occur much further down the ordered list, the worst example was a mating move that was listed as 24th at the first level.

One can see that while the move selector is reasonable most of the time, there are certain positions where it goes completely haywire for no apparent reason. It will also at times find the right move for the wrong reason. Alex Bell asked me to try it out on an opening trap, namely the Blackburne shilling game:-

1. P-K4 P-K4
2. N-KB3 N-QB3
3. B-B4 N-Q5

does it play 4. N*P accepting the offer of a free pawn?

4. .... Q-N4
5. N*BP?

forking the queen and rook but

5. .... Q*NP

which is a win for black.
In fact it played:-

4. N*N!

because of several moves that it considered potentially dangerous to its maximum depth of search, particularly N*Pch?

If we now force it to the position after 4. N*P? , Q-N4, it does not play 5. N*BP? but 5. B*Pch, because the check puts off several potentially dangerous moves such as N*Pch? or Q*Pch? or even Q*NP! (but that was about 4th).

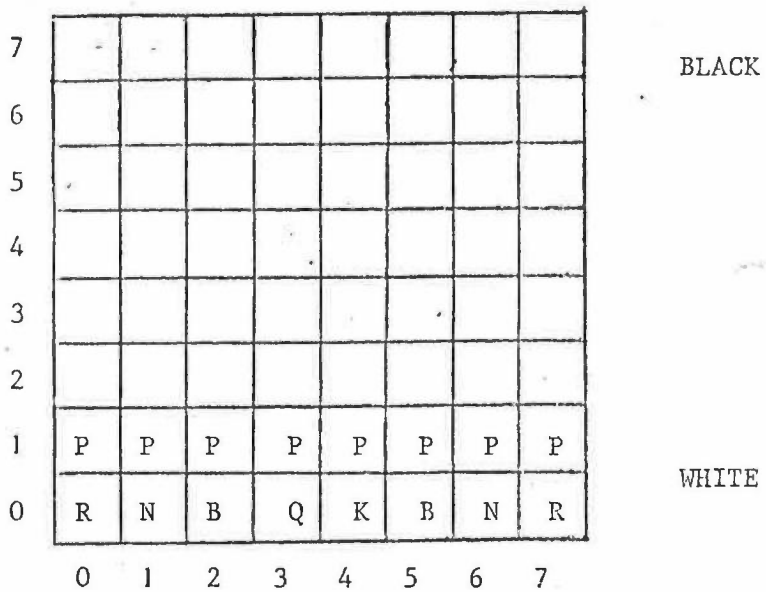
So the programs sometimes finds the best move or avoids the worst move for totally the wrong reasons. This odd behaviour is due in part to the shallow fixed depth of search. But searching deeper is not going to cure the problem, only hide it from view. The erroneous position evaluations will take place deeper in the search tree where their effect cannot be easily observed. It is a mistake to use deep searching too soon in a program's development. The position evaluation function and move selection really need to be very well developed and understood before attempting deep searching.

Running the program

Finally another sample game to show how moves are input:-

	W(1906A)	B	Octal Input (Black)
1.	P-K4	P-K4	6444
2.	N-KB3	N-QB3	7152
3.	B-N5	P-QR3	6050
4.	B-Q3	N-KB3	7655
5.	0-0	B-B4	7542
6.	N-B3	0-0	00
7.	K-R1	P-Q4	6343
8.	P*P	N*P	5543
9.	Q-K1	B-Q3?	4253
10.	N*N	R-K1	7574
11.	Q-K4	N-Q5	5233
12.	Q*Pch	K-B1	7675
13.	Q-R8 mate		

To play the computer we use an octal notation for input:-



Thus P-K4 for white is 1434, you may also put in 14 34, spaces are ignored. ← means it is waiting for your input. If you input the character @ the present position of the board is output. If you wish to castle, it will accept 00 or 000. If you get a pawn to the 8th rank it will ask what you want. Input N, B, R or Q; anything else and it will assume Q. It always turns its own pawns into a queen. If the character + is input after the move, eg 1434+, the program will make the move without checking it. It will then type out:-

```
MOVE ACCEPTED
YOUR MOVE
←
```

and wait for further input. This is useful for setting up board positions for testing purposes. Input an A to finish the game and then type in QU to quit the chess macro, The program is started by typing CHRUN or CHRUN B, in the latter case the program plays black.

Recent developments

A number of improvements have been made to the program since the conference. Several people have noted that the program often achieves its primary aim of controlling the centre squares of the board but then fails to capitalise on its position. This has been corrected by setting up a new array which lists all the squares surrounding the two kings.

Initially, an extra 6 point 'king bias' is awarded for control of each of these squares. An extra point is added to the king bias on each

move from the 10th to the 22nd. Thus after 22 moves 18 extra points are awarded for control of these squares. In addition, from the 10th move, all squares having a value greater than 6 are reduced in value by one point per move until all squares have the value 6.

It has been stated (Zobrist and Carlson, 1973) that it is difficult to include new chess concepts in a conventional chess program. Several of the concepts mentioned in Zobrist's paper have now been implemented with little effort by adding extra tables of piece value to the program.

The value of a knight is now read from a table. It has the value 85 at the edge of the board and 90 elsewhere. This not only discourages the program from moving its knights to the edges but also encourages it to develop its knights from their initial squares. A bishop table gives 95 points for a bishop in its initial position and 108 in all other positions. This encourages early development of the bishops A queen table gives 350 points for a queen in its initial square and 324 elsewhere. This discourages early development of the queen. After 10 moves all values in the bishop table are set to 108 and all those in the queen's table to 324.

The program was translated into PL/1 and all the above modifications included, in about six weeks of spare time programming by John Birmingham of AERE, Harwell. He has also modified it to search three plies deep, ie one more ply.

At present it uses the unsound centre counter defense 1. P-K4 P-Q4 and also tends to attack its opponent's undeveloped queen with an undeveloped bishop. These problems can be overcome by suitable modification to the tables.

The program now plays a far better game, both 2-ply and 3-ply versions. The change from central control to attacking the king is very noticeable. It defends well and if the position becomes complicated it takes level swaps (or better) until it can detect no further threats. Once a dead position is reached it moves all free men to attack squares round its opponent's king. It does not as yet test for a draw by repetition and as a consequence has drawn several won games.

On occasion it has played very good end games, queening its own pawns and preventing the queening of its opponent's pawns by long sequences of pins and checks. However if a pawn is still on its initial square in the end game there is no incentive to advance it because the value of all the squares is now reduced to 6 and the value of a pawn does not start to increase until it reaches the 5th rank. This can be corrected easily by modifying the pawn tables to give small increases in value on the 3rd and 4th rank.

FUTURE DEVELOPMENTS

The existing program uses the alpha-beta cutoff technique to speed up tree searching. This is most effective when the moves are ordered so that the most likely cutoff moves are examined first. The moves are already ordered at the higher levels of the look ahead tree, but so far no attempt has been made to order the moves at the deepest level

of search as this would require a prior knowledge of the value of each move.

A method of performing this ordering has now been proposed. It is based on the idea that a refutation for one of your opponent's best moves is likely to be a refutation for most of his following moves (see Computer Chess Experiments).

The algorithm (known as the 'killer heuristic') will operate as follows:-

(i) List and evaluate all replies to the first move at (full search depth - 1). Re-order the men in the WHITEPIECE or BLACKPIECE array so that men having a good reply are examined first.

(ii) Order the moves for each man and use this information to re-order the tables used in computing each man's move, so that preferred directions are examined first.

(iii) Modify LISTMOVES so that the moves of each man are generated and evaluated separately. This will avoid unnecessary work listing moves that are never examined.

3

P S Y C H O L O G Y A N D C O M P U T E R C H E S S

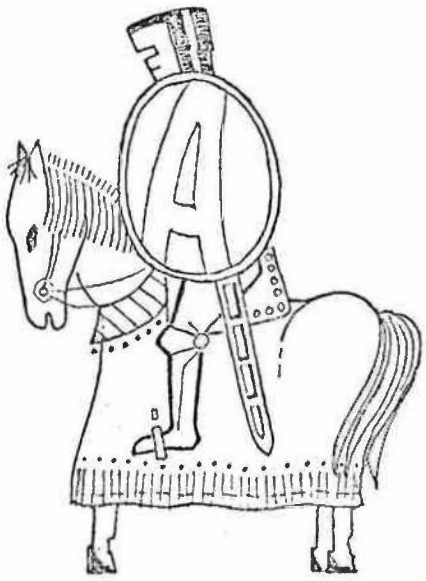
by

A H Bond

Queen Mary College
University of London
Mile End Road
London
E1 4NS

"If you know the enemy and know yourself you need not fear the result of a hundred battles. If you know yourself but not the enemy, for every victory you will suffer a defeat. If you know neither you will always be beaten."

- GENERAL SAN-TZU



Editor's Note -

It is mainly due to Alan Bond's interest in psychology and his enthusiasm in communicating that interest that I have become convinced that anyone who wants to write a successful chess program must "know the enemy". Unfortunately, due to other commitments, Dr Bond has been unable to describe investigations of the "enemy" and his "methods" to the extent that I think the subject deserves.

I have spent many hours discussing the problem with him and, by and large, we agree upon what we disagree upon. I have therefore included a very short resume of his talk but have taken the liberty of expanding on the subject, not as dogma but for contrast.

I would like to acknowledge the DESCRIPTOR INDEX and REFERENCES he has provided.

Introduction

The fraction of workers who believe that the study of human behaviour can illuminate the study of "pure" artificial intelligence is disturbingly low, probably less than one third. As a member of this fraction I tend to picture the relationship between AI and cognitive psychology as one of mutual benefit particularly if the subject is chess.

The main illumination that cognitive psychology can suppy to AI is in providing ideas. There is no doubt that ideas are now needed for a successful chess machine; psychology has used the game for decades as a standard task environment.

The results of such work should be studied more. Apart from helping to produce chess machine ideas such studies have given us concepts and mechanisms which help us to pose interesting problems about intelligence in general.

Motivation

As motivation for this talk let me caricature an idea from Simon's "The Sciences of the Artificial". It is that since an adaptive machine adapts to its environment, it will in general incorporate an efficient adaptation provided the requirements of this new state do not violate any natural contraints such as speed or storage of the machine.

Thus when a machine is well adapted to its environment and operating within its limitations, its behaviour will be the same as all other optimally adapted machines and will be principally a property of the environment and not the machine. Only when operating near their limitations do the machines differ.

If we assume that the best human information processing in the environment of chess problems is almost perfect, then we may postulate that the human mechanism is the most efficient in the

sense of being the best adapted. Hence the most efficient chess program must behave like a human.

I believe that support for this argument exists in that changes in recent chess programs brought about for efficiency's sake have been changes towards human behaviour.

Furthermore the key to efficiency seems to rest in the acquisition and use of miscellaneous information about the chess position which in turn rests upon the flexible description of information. Humans are demonstrably impressive at extracting and using information in a flexible way in the chess environment.

Experimental Methods

Turning now to what is known of human behaviour in chess situations we find the subject in its infancy. Humane methods to study human information processing must necessarily be rather indirect. We do not however need to go to the extremely behaviourist position and exclude introspective reports. Verbal reports from a subject are, after all, data and by definition true. Whether there is a simple relationship between this data and the information processes under study is another matter. I do not know of any model of the verbalisation process.

The methods used then are mainly two, both verbalisation. They are introspection and thinking aloud.

An introspective verbalisation is done after the process to be investigated has taken place and consists of the subject's description of what he thinks he thought. It may include accounts of moving images, intuitions, etc. Introspection was used a lot until about 1920 when it fell into disuse.

Thinking aloud was used as a technique first by Duncker in 1935 and is just what it says, namely the subject talks whilst he is solving the problem. This must interfere to some extent with his thinking, probably inhibiting the non-verbal processes and enhancing the rationalisation processes. The relationship of the verbal report to the total information processing activity is unclear. However most workers accept it as a rough indication of partial contents and order of the thoughts described.

One usually studies a subject's behaviour on a choice of move problem in chess, ie one does not study a complete game but instead gives the subject a chess position and asks him to play just the next move. Usually the position is taken from a game but not one played by the subject. However in one study (Wagner, 1971) a subject played a game and in one position verbalised his thought in choosing the next move. The behaviour observed was similar to that in the artificial positions.

Another experimental method that has been used in the study of human perception of the chess board is the eye movement camera which produces a film showing the point on the board on which the eyes are fixed at any moment.

Summary of Chess Studies

The earliest study was by the great psychologist Binet who, in 1893 (reprinted 1966), studied introspective reports of blindfold chess players. His paper remains a classic.

Cleveland (1907) made some remarks on the stages of learning chess, based on reports by players. The main work on chess was done in the war period 1939-45 by De Groot and is presented at length in his book. De Groot is still professor in Amsterdam and has pursued his 'thinking aloud' method and the study of thought. More recent remarks by him are in (De Groot, 1967). His book ends with some illuminating remarks on chess playing programs and there is also a separate paper (De Groot, 1964).

Following De Groot a detailed analysis of exploratory processes in chess was made by Newell and Simon (1965) and this work is described in their book (1972). An independent study of their findings was undertaken by Wagner and Scurrali (1971).

Recent work on eye movements has been done by De Groot and his student Jongman in (De Groot, 1966) and (Jongman, 1966) and by the Russians Tikhomirov and Poznyanskaya (1966).

Simon and Barenfield (1969) tried to explain some perceptual phenomena as coding processes into "chunks" (see next section - Editorial Extension) and Chase and Simon (1972, 1973) tried to establish the existence of and identify some of the perceptual "chunks" by further experiments, particularly the technique of board reconstruction.

Editorial Extension - A G Bell

The following repeatable results have been obtained by psychologists studying chess players.

(1) From the experimental methods of introspection and thinking aloud used by De Groot it was not possible to distinguish the grandmaster from an ordinary player - the number of moves examined is the same (usually 2 or 3) per position; the depth and apparent speed of search differs only slightly.

Obviously the grandmaster must be able to select stronger moves for his consideration. How does he do this is the key question.

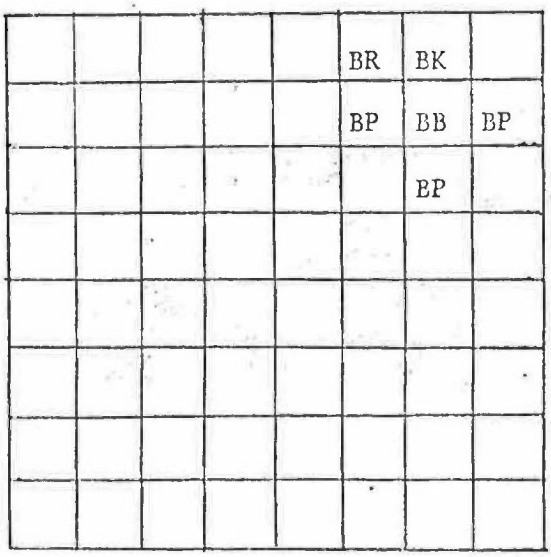
(2) De Groot repeated and extended a classic experiment first performed by the Russians. He verified that it is possible to distinguish the master from the amateur by briefly displaying, for about 5 seconds, positions from master play. Grandmasters can reproduce such positions almost perfectly, amateurs can replace only one third of the pieces on average.

(3) If the chess positions displayed are random - the pieces are placed haphazardly - then again performance becomes indistinguishable. Most people can only replace about one sixth of the pieces irrespective of their chess skill.

The conclusion drawn from these experiments is that chess skill cannot be detected from observing the search process but can be detected by pattern recognition ability.

(4) The recognition and reconstruction of a position is done from short term memory. G A Miller, in a famous article "The magical number seven, plus or minus two", proposed a short term memory model with a capacity of about seven "chunks". The master player must be able to recognise a meaningful position by describing it in about seven chunks, ie for about twenty pieces recalled he must have about three pieces per chunk. We can partially explain the remarkable ability of chess masters to reconstruct positions by them possessing an enormous repertoire (vocabulary) of familiar patterns (chunks) any seven of which can be put together to reproduce what he has seen.

(5) Experiments have been performed to find how many "chunks" a master player possesses and try to isolate some of them. It appears that a chess master can recognise about 100,000 different clusters of pieces. Here is one of them



(Also the most likely position these pieces will occupy at the 21st move in a master chess game)

This is a very familiar pattern to the master player. The fact that it is familiar can be verified by eye movement experiments, where it can be shown that the master hardly (if at all) fixates on any of these pieces. His peripheral vision informs him about a pattern he has seen thousands of times before, he does not need to look at it closely.

(6) The chess chunks (words) in a master's vocabulary can be isolated more convincingly by timing and observing the order in which the master reproduces a position he can see upon another board. The subject indicates (unconsciously) the end of one chunk and the start of another by turning his head. If the first board is not displayed continuously then any pauses in the reconstruction process can also be inferred to be inter chunk boundaries.

And so a partial understanding of the processes that expert chess players use when choosing a move has been obtained. At first sight however it does not appear to be of much use to the computer scientist for the following two reasons.

Firstly, the acquisition of a vocabulary of 100,000 patterns takes a human at least six solid years staring at chess positions in games he is playing against experts.

Secondly, even if we could identify and input many of these patterns, how does any combination of seven of them suggest plausible, strong moves to the master player?

Quite obviously chess knowledge is not going to be acquired by a computer in the same, inefficient way a human acquires it. The belief that the program must "make use of essentially the same methods as those used by men" seems fatuous because human methods derive from practice - thousands of hours of practice - with an inbuilt limitation of a seven "chunk" short-term memory apparently playing an important role.

However what may be usefully derived from this work in terms of a chess program is that we may have discovered some of the weaknesses of the "enemy". We may now be able to "jam" his system.

Let's assume we can identify and input many of the chess "chunks". The program now tries to produce positions or situations which require more than seven chunks to be recognised and described providing its normal evaluation function (however derived) is not too seriously violated. This might seriously impair the human's ability to have strong moves suggested to him.

Whether such an approach is possible or not, the point I am trying to make is that there appears little proof that humans are particularly efficient at chess. If it were so, then I would agree that a program would have to simulate very exactly the human behaviour at the time of playing, but the limitation of seven chunks in the short-term memory could imply that the best humans are operating near the human limitation and therefore a successful chess machine need not be a "paradigm of the human mind".

4

M A T H E M A T I C A L R E L A T I O N S I N C H E S S

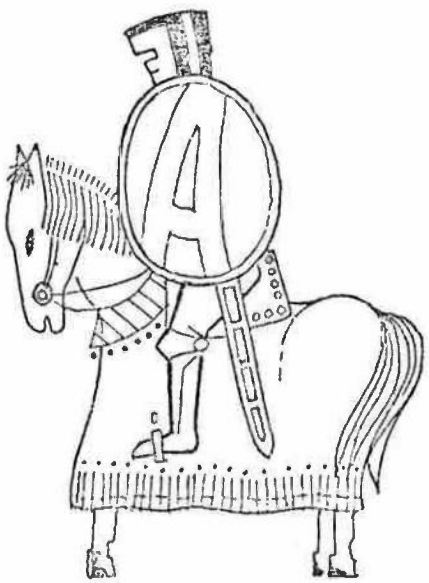
by

R H Atkin
and
I H Witten

Mathematics Department
University of Essex
Wivenhoe Park
Colchester
Essex
CO4 3SQ -

"I gather this work is so learned that few people are able to read it."

Comment on the "Treatise on the Application of Mathematical Analysis to the Game of Chess" by JAENISCH (a Russian), published about 1890.



Editor's Note -

Should be read in conjunction with (Atkin, 1972). The emphasis is no longer on tree searching, position evaluation is done mathematically and should be repeatable.

At present the program considers each legal move by white and then examines the consequent changes (increases) in (only) seven features. It then sums the scores under these seven headings to give an overall positional score for the move.

Atkin states that there is great scope for improvement particularly if chess masters can be persuaded to help in the research.

1.0 Introduction

We examine the game of chess by looking at an important relation which exists between the pieces and the squares, and which embodies the moves allowed to the former. This relation is mathematically equivalent to a simplicial complex which, in its turn, possesses a geometrical representation in the euclidean space E^{53}. It is therefore possible to interpret the course of a game of chess as the expansion and contraction of two geometrical structures (one for White and the other for Black) in this multi-dimensional space[1]. This seems to provide us with a natural language with which to discuss the accepted positional theories in chess. It is also particularly well suited to expression in a computer language, and we illustrate this aspect by demonstrating some typical analysis in specific situations.

Finally we try to indicate the potential richness of this structural language and to suggest various lines of research which might be profitable in the broader context of board games played by computers.

1.1 The relations Γ_W, Γ_B

Let $W = \{W_i;\ i = 1, 2, \ldots 16\}$ be the set of White men, and $S = \{S_j;\ j = 1, 2, \ldots 64\}$ be the set of squares on the board. Then we define the relation $\Gamma_W \subset W \times S$ in the following way.

Definition: $(W_i, S_j) \in \Gamma_W$ if and only if W_i "attacks" S_j. By "attacks" we mean that one of the following holds true:

(a) if it is White's move, and W_i is a piece (not the king or a pawn), then "W_i moves to square S_j" is a legal move;

(b) if W_i is a pawn then S_j is a "capturing square" for W_i;

(c) if there is a White man, W_k $(k \neq i)$, on S_j then W_i is protecting W_k, in the ordinary sense of chess-players' parlance;

(d) if W_i is the White king (WK) then S_j is an immediate neighbour to the square occupied by W_i, horizontally, vertically, or diagonally;

(e) if S_j contains a Black man, B_k $(\neq BK)$, and if it is White's move, then "W_i captures B_k" is a legal move;

(f) the BK is on S_j and is in check to W_i.

We notice that, under (a), the empty square in front of a pawn is not related to that pawn via Γ_W. Also we notice that if W_i is on square S_j then $(W_i, S_j) \notin \Gamma_W$; a piece cannot defend itself. These

points are not crucial to our discussion - which primarily illustrates a <u>*method*</u> *of attacking the problem.*

It is clear that there is another relation between W and S which cannot be ignored, viz., that relation which tells us on which squares the men are to be found. But this relation is actually a mathematical <u>*mapping*</u>*,*

$$pos: W \to S$$

and therefore possesses a trivial structure (c.f. section 1.2).

We now have the two relations $\Gamma_W \subset W \times S$ *and* $\Gamma_B \subset B \times S$*, one for each player. When the difference is irrelevant we shall denote either by* Γ *. As a further point of detail, relevant in discussion of specific cases, we shall otherwise denote the members of S by their accepted algebraic notation a1, ..., h8 (in the order of 1, ... 64) and we shall denote the members of W by the obvious*

WQR, WQN, WQB, WQ, WK, WKB, WKN, WKR, WQRP, ... WKRP

in the order 1, ... 16; with a similar notation for B.

1.2 Γ_W *defines two simplicial complexes*

If there exists at least one W_i *such that (p+1) squares* S_{α_r}*,* $r = 1, \ldots (p+1)$*, are* Γ_W*-related to* W_i*, we say that these S's constitute*

a p-simplex (one of whose names is W_i), and denote it by σ_p, so that

$$W_i = \sigma_p = \langle S_{\alpha_1}, S_{\alpha_2}, \ldots, S_{\alpha_{p+1}} \rangle$$

Any subset of these (p+1) S's is called a face of this p-simplex, and is a t-simplex ($t \leq p$) in its own right. It follows that the relation Γ_W can be described as a collection of p-simplices, for various values of p. Such a collection (closed under the relation " — is a face of —") is called a simplicial complex (a "complex" of simplices) and is denoted by $K_W(S; \Gamma_W)$.

This notation is used to suggest that the set S plays a special role - in terms of which the simplices W_i are defined. This set S is usually referred to as the vertex set. When Γ_W is understood we can abbreviate the notation to $K_W(S)$; then

$$K_W(S) = \{p\text{-simplices};\ 0 \leq p \leq N\}$$

where $p = 0$ corresponds to 0-simplices of the form $\langle S_\alpha \rangle$, and where N is the maximum value of any p in this collection. The value of p is called the dimension of the p-simplex (v. section 1.3) whilst N is called the dimension of the complex; $N = \dim K$.

We notice too that Γ_W may be such that some squares S_j are not vertices of any simplex W_i, not being "attacked" by any of White's men.

But we also notice that Γ_W possesses an inverse relation Γ_W^{-1} (which relates S_i to a set of W_j) - the incidence matrix of Γ_W^{-1} being

the transpose of that of Γ_W. This relation therefore defines a simplicial complex

$$K_S(W, \Gamma_W^{-1}), \text{ or } K_S(W)$$

referred to as conjugate to $K_W(S)$. In $K_S(W)$ the vertex set is W whilst each S_i (in Γ_W^{-1}) is a simplex; for example, if S_i is a p-simplex, then

$$S_i = \langle W_{\beta_1}, W_{\beta_2}, \ldots W_{\beta_{p+1}} \rangle$$

which means that S_i is simultaneously attacked by the $(p + 1)$ White men W_{β_i}, $i = 1, \ldots, (p + 1)$.

We shall describe the complex

$K_W(S)$ as White's view of Board

and $K_S(W)$ as Board's view of White

Similarly, $K_B(S)$ is Black's view of Board

whilst $K_S(B)$ is Board's view of Black .

These complexes are well-defined at each stage of the game. When White has made I moves and Black has made J moves we shall say that the game is in mode $[I, J]$. Clearly $J = I - 1$ or I. Also, in terms of a well-known convention, mode $[I, J]$ corresponds to the completion of $(I + J)$ plys.

The complexes defined above are functions of the mode; they need to be recomputed after every move. It is clear that, in general, a move by White, say affects all four complexes.

1.3 *A geometrical representation of* $K_W(S)$ *in* E^{53}.

If we identify the p-simplex $\langle S_{\alpha_1}, \ldots S_{\alpha_{p+1}} \rangle$ *with a convex polyhedron, vertices the* S_{α_i}, *in p-dimensional euclidean space* E^p *then we can obtain a geometrical representation of the whole complex* $K_W(S)$, *in a suitable space* E^H. *A well-known theorem*[2] *tells us that, if* $N = \dim K$, *an economical value of H is*

$$H = 2N + 1$$

From a consideration of Γ_W *we notice that the maximum value of dim* W_i, $W_i \in W$, *is 26. This occurs when the WQ is in the centre of the board (say, on square d4), for if its range is unobstructed it then attacks a total of 27 squares. This means that under these circumstances WQ is a 26-simplex. It follows that* $K_W(S)$ *can always be represented in the space* E^{53} *- and this is independent of how many Queens are on the board.*

Since $\dim(BQ) \leq 26$ *in* $K_B(S)$ *we can contemplate the complex* $K_W(S) \cup K_B(S)$ *and find a representation of it in the euclidean space* E^{53}. *Thus both of the geometrical structures* $K_W(S)$ *and* $K_B(S)$ *can be regarded as existing in* E^{53}, *for all possible modes [I, J], in all possible chess games.*

In this sense we can say that a game of chess can be modelled, via the interplay of connected polyhedra, in E^{53}.

1.4 q-connectivity in the complex $K_W(S)$

Simplices W_i and W_j are said to be joined by a chain of connection if there exists a finite sequence of simplices

$$\sigma_{\alpha_1}, \sigma_{\alpha_2}, \ldots \sigma_{\alpha_h}$$

such that

(i) σ_{α_1} *is a face of* W_i

(ii) σ_{α_2} *is a face of* W_j

(iii) σ_{α_i} *and* $\sigma_{\alpha_{i+1}}$ *have a common face (say)* σ_{β_i} ; $i = 1,\ldots,(h-1)$.

We say that this chain of connection is a q-connectivity if q is the least of the integers

$$\{\alpha_1, \beta_1, \beta_2, \ldots, \beta_{h-1}, \alpha_h\}$$

As a special case, a simplex σ_p is p-connected to itself, but is not (p+1)-connected to any σ_r.

If we define a relation γ_q as meaning " is q-connected with" then γ_q is an equivalence relation on the simplices of K. The classes of γ_q, or the members of the quotient set K/γ_q, are now the pieces of K which are separately q-connected. We use the notation

$$Q_q = \text{cardinality of the set } K/\gamma_q$$

and the process of computing all the values of Q_q, for $q = 0,\ldots,$ dim K, is referred to as a Q-analysis[5]. *If N = dim K, we arrange these*

q-values to give a vector, what elsewhere[5] has been called the structure vector,

$$\mathcal{Q} = \{Q_N, Q_{N-1}, \ldots, Q_1, Q_0\}$$

The value Q_0 is, in fact, the same as the zero-order Betti-number of the complex, but the higher order values Q_q must not be confused with the higher order Betti numbers. Thus, 0-connectivity is the same concept as arcwise connectivity; our higher order Q_q-values are a generalisation of this notion.

2.0 Positional motifs arising in $K_W(S)$

In this section we use the following definitions:

$\hat{q}$ = top q-value of a simplex
= dimension of the simplex in $K_W(S)$,

$\check{q}$ = bottom q-value of a simplex
= largest q-value at which the simplex is connected to a distinct simplex,

$Ecc(\sigma)$ = eccentricity of a simplex σ
= $(\hat{q} - \check{q}) \div (\check{q} + 1)$, when that ratio exists.

(A) The value of $\hat{q}(W_i)$ is the dimension of the White man W_i in the complex $K_W(S)$; it therefore equals the value

{number of squares attacked} - 1

This top-q value therefore tells us the dimension of that subspace

of E^{53} in which is located the polyhedra whose name is W_i. It is therefore an indication of the geometrical horizon (in E^{53}) enjoyed by W_i. This suggests that $\hat{q}(W_i)$ is a measure of the mobility of W_i, in this particular mode.

We notice that the maximum values of the $\hat{q}$-numbers for the various men are as follows:

$$\max \hat{q}\ (Q) = 26, \quad \max \hat{q}\ (R) = 13, \quad \max \hat{q}\ (N) = 7$$

$$\max \hat{q}\ (B) = 12 \text{ (on half the board)} = 6 \text{ (on whole board)}$$

$$\max \hat{q}\ (P) = 1$$

The ratio of $\max (\hat{q} + 1)$ for all the men are therefore

$$27 : 14 : 8 : 7 : 2$$

for $$Q : R : N : B : P$$

These should be compared with the classical static "values" of the pieces, namely,

$$Q : R : N : B : P = 9 : 5 : 3 : 3 : 1$$

(B) Since $\check{q}\ (W_i) = \max_k \dim (W_i \cap W_k)$ it follows that W_i and W_k are simultaneously attacking $(\check{q} + 1)$ squares; they share a $\check{q}$-face in the structure $K_W(S)$. This value indicates the extent of the co-operation of the pieces W_i, W_k, as well as their mutual mobility.

(C) The Ecc $(W_i) = (\hat{q} - \check{q}) \div (\check{q} + 1)$ indicates the extent to which W_i is a lone attacker. If Ecc $(W_i) = 0$, then

$$\hat{q} = \check{q}$$

and so the action of W_i is entirely in harmony with (at least one of) the other pieces. We notice that when $\check{q} = -1$ (W_i is then totally disconnected from all other pieces) Ecc $(W_i) = \infty$. Otherwise the largest value of Ecc (W_i) is $\hat{q}$ (when $\check{q} = 0$). Excluding the extreme case, when $\check{q} = -1$, we therefore have the inequality.

$$0 \leqslant \text{Ecc } (W_i) \leqslant \hat{q}$$

for Ecc (W_i).

A move which lowers Ecc (W_i) can clearly do so in one of two ways; either by decreasing $\hat{q}$ (W_i) - reducing its effectiveness (development) on the Board, or by increasing $\check{q}(W_i)$ - increasing the co-operation and mutual mobility with other pieces.

We notice, for example, that if we were to use the value of Ecc (W_i) to obtain a static "value" for W_i then the classical numbers 9, 5, 3, 3, 1 can arise in various ways - which depend upon the bottom-q values. Thus if we take

(i) $\check{q} = 0$ we obtain Ecc $(W_i) = 9, 5, 3, 1$
when $\hat{q}(W_i) = 9, 5, 3, 1$

(ii) $\check{q} = 1$ we obtain Ecc $(W_i) = 9, 5, 3, 1$
when $\hat{q}(W_i) = 19, 11, 7, 3$

(iii) $\check{q} = 2$ we obtain Ecc $(W_i) = 9, 5, 3$ (pawn excluded)
when $\hat{q}(W_i) = 29, 18, 12$

This latter case is impossible for any of the pieces, and so we deduce that the classical values can only plausibly correspond to Ecc (W_i) at the level of $\check{q} = 0$. At this level W_i and W_j are 0-connected if they simultaneously attack a common square (only one).

(D) A move which lowers the value of Q_t, for some fixed t, in the structure vector $\underset{\sim}{Q}$ can do so in more than one way. In the first place, Q_t can only change by multiples of unity ($\Delta Q_t = \pm n$), and if

$$\Delta Q_t < 0$$

then $\check{q}$ must have increased for some W_i, and $\check{q}$ cannot have decreased for any W_j. Thus $\Delta Q_t < 0$ can result from an increase in the co-operation of the pieces.

On the other hand it is possible for $\Delta Q_t < 0$ by some one (at least) component disappearing at the t-level. This can happen by a reduction in $\hat{q}$ (W_i) for some i - in such a way that, after the move,

$$\hat{q}\ (W_i) < t$$

whereas, before the move, $\hat{q}$ (W_i) > t. We notice too that it need not be the piece W_i which is involved directly in the move; the movement of W_j can effectively block the action of W_i so as to induce the reduction of $\hat{q}$ (W_i).

The co-operation of pieces and pawns, manifest at various t-levels, can be displayed as follows:

$t = 0$	any pair of $\{K, Q, R, N, B, P\}$	(share 1 square)
$t = 1$	any pair of $\{K, Q, R, N, B, P\}$	(share 2 squares)
$t = 2$	any pair of $\{Q, R, N, B\}$	
	any pair of $\{K, Q, R\}$	(share 3 squares)
$t = 3$	any pair of $\{Q, R, N\}$	
	any pair of $\{K, Q, R\}$	
	B and N	
	Q and B on same diagonal	(share 4 squares)
$t = 4$	Q and B on same diagonal	
	Q and R on same file or rank	
	R and R on same file or rank	(share 5 squares)
$t = 5$	Q and B on same diagonal	
	Q and R on same file or rank	
	R and R on same file or rank	(share 6 squares)
$t = 6$	Q and R on same edge file	
	or edge rank	(share 7 squares)
$t \geq 7$	no two pieces 7-connected	(share ≥ 8 squares)

It follows that if $\Delta Q_t < 0$, *when* $t \geq 7$, *the reason must be the fact that a piece* W_i *exists for which the move has resulted in old* $\hat{q}(W_i) \geq 7$ *and new* $\hat{q}(W_i) < 7$

An exception to this occurs if there are two White Queens on the Board - say, one on a1 and the other on a8. These Queens are then 7-connected (if the 1st and 8th ranks, as well as the leading diagonals, are otherwise clear. In this special (and

unusual) case $\Delta Q_7 < 0$ can be the result of an increase in piece co-operation at the 7-level.

2.1 Positional motifs arising in $K_S(W)$

It is in a consideration of the geometry of $K_S(W)$ that we can see an expression of the positional theories first advanced by Steinitz[6].

(A) Each square $S_i \in K_S(W)$ is a p-simplex, for some value of p, so that

$$S_i = \left\langle W_{\beta_1}, W_{\beta_2}, \ldots, W_{\beta_{p+1}} \right\rangle$$

where the W_{β_i} denote White men attacking S_i. Other things being equal, it is clear that dim (S_i) is a measure of the control exercised over S_i by the White men. But the question of absolute control cannot be settled without

(i) comparing dim (S_i), $S_i \in K_S(W)$, with dim (S_i), $S_i \in K_S(B)$, and (ii) allowing for the relative "values" of the vertices (the W_{β_i}) in the simplex S_i.

In the sense of Steinitz, S_i is a strong square for White when the control is maximal or absolute. Ideally, for White, $S_i \in K_S(W)$ but $S_i \notin K_S(B)$. But failing this, and taking (ii) into consideration, the presence of pawns in the p-simplex S_i of $K_S(W)$ - and their absence in the simplex S_i of $K_S(B)$ - makes S_i a strong square for White, and a weak square for Black. A notable example of such a square is one which lies in front of an isolated Black pawn; for here we have a situation in which Black cannot (usually) introduce a pawn into the

simplex S_i of $K_S(B)$. Thus Black has a permanent weakness - the geometry cannot be repaired (except perhaps with White's co-operation).

This would suggest that the whole simplex S_i is in some sense, which must eventually be given a numerical value, a measure of the strength of that square S_i. In this context we must clearly distinguish between the control value of a piece W_j and its piece value; the former being, in some sense, inverse to the latter.

It seems natural in the light of these remarks to interpret control value of W_j as a mapping

$$\underline{c\ val} : W \to J$$

from the vertex set W of Board's view of White, $K_S(W)$, into (say) the integers J; whilst the piece value of W_j will be a mapping

$$\underline{p\ val} : W \to J$$

from the simplices of $K_W(S)$, White's view of Board.

Thus c val and p val are "conjugate" in the sense that they have conjugate complexes as their domains. Naturally the choice of J as the range for these mappings is not crucial - but it can be a convenient computing feature.

(B) The bottom q-value, $\check{q}\ (S_i)$, means that S_i shares a $\check{q}$-face with at least one other square S_j. This means that S_i and S_j are

simultaneously attacked by $(\overset{\vee}{q} + 1)$ pieces. Let this $\overset{\vee}{q}$-face be the simplex

$$\sigma = \langle W_1, W_2, \ldots, W_{q+1} \rangle$$

which must therefore be an indication of the "square-co-operation" between S_i and S_j, via the White men. The value of $\overset{\vee}{q}$ therefore indicates the flexibility inherent in White's game, the existence of multiple threats. The squares S_i, S_j, S_k,... which share a common $\overset{\vee}{q}$-face define areas of the board where White's flexible threats are to be found. The "value" of a threat depends on whether it is

(i) a threat to control (a square)

or (ii) a threat to occupy (a square).

If it is a threat to control then, for White, it would be valued as (plausibly)

$$\sum_i \underline{c\ val}\ (W_i) \qquad W_i \in \sigma$$

whilst if it is a threat to occupy its value will be

$$\sum_i \underline{p\ val}\ (W_i)$$

This is because, in the first case, we are dealing with $K_S(W)$, but in the second case we are dealing with $K_W(S)$.

(C) A move which lowers Q_t, for some fixed t, in the structure vector Q for $K_S(W)$ will (c.f. section 2.0) do so because of two possibilities. On the one hand there might be an increase in $\overset{\vee}{q}$

for some square S_i, so that $\Delta\check{q} > 0$ results in $\Delta Q_t < 0$ $(\check{q} \leq t)$. This means that the flexibility of White's threats has increased.

On the other hand $\Delta Q_t < 0$ can result from a decrease in $\hat{q}\ (S_i)$, for some square S_i $(\hat{q} \geq t)$. This means that White's control over S_i has been reduced.

The "knight fork" is an obvious example of square co-operation, at the level of $q = 0$, in $K_S(W)$ - when S_i and S_j are not adjacent. For other pieces the co-operation involves neighbouring squares - either on the ranks, files, or diagonals. The action of the N cannot be blocked by other pieces or pawns so that, placed in the larger central area of the Board, a knight always induces a 0-connectivity between 8 squares.

(D) The positions of the squares $\{S_i\}$, relative to the locations of the Black men, are clearly important. This is embodied in the importance normally attached to the centre squares, to open files, to open (long) diagonals, to the seventh/eighth ranks. We naturally add to these the squares occupied by the Black men, that is to say, the set

$$\{pos\ (B_i)\}$$

as well as the King flight squares.

The centre squares, as $pos\ (W_i)$, allow the possibility of maximum $\hat{q}$-values for the White Q and B. The open files are necessary for the achievement of maximum $\hat{q}$-values for the White R's. Each

of these features is further enhanced by open diagonals and open ranks. The R on the seventh rank is usually associated with the King flight squares, but also it can pose strong tactical threats behind the Black pawns (which are then on weak squares).

All these dimensional considerations are expressive of the complex $K_W(S)$. Thus we see that the consideration of "square-value" in $K_S(W)$ is inevitably involved with considerations of "control-value" in $K_S(W)$. And the conjugate nature of $K_S(W)$ and $K_W(S)$ would then suggest that there should be a close relation between "piece-value" in $K_W(S)$ and what we might introduce and call "strength-value" in $K_W(S)$.

Precisely, we can proceed as follows.

Define a mapping to represent the square-value of a simplex $S_i \in K_S(W)$,

$$\underline{s\ val} : S \to J$$

and require the condition that

$$\text{whenever } S_i = \langle \ldots W_j \ldots \rangle \text{ in } K_S(W)$$

$$\text{then } \underline{s\ val}\ (S_i) = \sum_j \underline{c\ val}\ (W_j) \qquad \ldots (I)$$

Define a mapping to represent the strength-value of a vertex $S_j \in K_W(S)$,

$$\underline{st\ val} : S \to J$$

and require the condition that

$$\text{whenever } W_i = \langle \ldots S_j \ldots \rangle \text{ in } K_W(S)$$

$$\text{then } \underline{\text{p val}}\ (W_i) = \sum_j \underline{\text{st val}}\ (S_j) \qquad \ldots (II)$$

The process of estimating the relative "values" of pieces and squares can now be seen as a cyclic one which alternates between the two conjugate complexes. This is because, in some sense to be defined, we must have $\underline{\text{p val}}\ (W_i)$ *to be "inverse" to* $\underline{\text{c val}}\ (W_i)$ *and, similarly,* $\underline{\text{s val}}\ (S_j)$ *to be "inverse" to* $\underline{\text{st val}}\ (S_j)$. *By "inverse" we mean only*

that if $\underline{\text{p val}}\ (W_1) > \underline{\text{p val}}\ (W_2)$ *or* $\underline{\text{s val}}\ (S_1) > \underline{\text{s val}}\ (S_2)$

then $\underline{\text{c val}}\ (W_2) > \underline{\text{c val}}\ (W_1)$ *or* $\underline{\text{st val}}\ (S_2) > \underline{\text{st val}}\ (S_1)$

One way of ensuring this reversal of ordering is to take, for example,

$$\underline{\text{c val}}\ (W_i) \cdot \underline{\text{p val}}\ (W_i) = \text{a constant integer}$$

and then truncate on division.

Another way would be to fix an integer n_o *and take*

$$\underline{\text{c val}}\ (W_i) \equiv \underline{\text{p val}}\ (W_i) \pmod{n_o}$$

The cycle can be entered in a crude way by taking, for example,

$$\underline{\text{p val}}\ \{K, Q, R, N, B, P\} = \{10, 9, 5, 3, 3, 1\}$$

and inventing some similar rigid square-values, depending on the Board, for example

$$\underline{\text{st val}}\ \{S_i, S_j, S_k\} \leq \{3, 1, 5\}$$

where S_i = a centre square

S_j = an off-centre square

S_k = BK flight-square, with obvious extensions.

(E) The condition of checkmate can be described in terms of the apparent conflict between the geometrical structures of White and Black. In $K_B(S)$ the BK is a p-simplex, with $2 \leq p \leq 7$. Each square S_i in this simplex BK is a possible flight-square, allowing for obstruction by Black men. In addition pos (BK) is a single square, say S_{BK}.

Now suppose that, in mode [I, I-1], we have

(i) $\langle S_{BK} \rangle \in K_S(W)$

and (ii) $BK \in K_S(W)$

then we know that the BK is in check, by (i), and the possible flight-squares are under attack, by (ii). Hence the BK cannot get out of check by moving (himself). The only escape is for Black to change (i), presumably by blocking or capturing the checking piece.

We can therefore deduce that Black is in checkmate if the above conditions (i) and (ii) are invariant under all legal transitions from mode [I, I-1] to mode [I, I].

The White geometrical structure has "annexed" that portion of Black's structure which contains the BK and his flight-squares.

3.0 A computer program for the analysis

The computer program which embodies the positional chess heuristics is written in Fortran and runs in 9K of core on a PDP-10. Although more modern and sophisticated languages like LISP, BCPL, and POP-2 were considered, Fortran was chosen in spite of its many and obvious disadvantages for the following reasons:

(i) speed of execution - the PDP-10 Fortran compiler produces unusually efficient object code,

(ii) transparent compilation - when writing sections of Fortran, one knows (roughly) what machine code the compiler is going to produce,

(iii) modular subroutine structure,

(iv) critical sections written in machine code can be interfaced easily to the rest of the program,

(v) good compiler error diagnostics,

(vi) fast array handling.

The program is designed to allow interactive analysis of existing chess games from a teletype keyboard. In addition, requests for extensive analysis of complete games can be submitted to the PDP-10 batch system. Using existing games by master players permits repeated analysis of a large number of high-quality games, eliminating time-consuming keyboard sessions with chess players. A further advantage of playing existing games is that it allows

study of all phases of the game - in computer chess, checkmate usually intervenes before the end-game is reached! The program has never attempted to play a complete game itself.

To facilitate human interaction with the computer, the program accepts and obeys commands typed in at the keyboard. The chess game to be analysed is stored in a disk file in a slightly extended version of the International Algebraic Notation for chess games (a BNF description of the notation is given in Appendix A). Commands are provided to print the board, make a specified number of moves from the game, move to a specified point in the game, and so on. It is possible to investigate variations on the game by typing in a sequence of moves different from those actually played. Further commands print the connectivity matrix and structure vector for either side's view of the board, and initiate a complete positional analysis of the current state of play. Repeated analysis of the game at various stages is accomplished by a MACRO command which continually performs any sequence of other commands. A typical command sequence for a batch run is

```
BOARD     /print the board
POSN      /perform a positional analysis
MOVE 2    /make two moves (one for White, one for Black)
MACRO     /repeat the above command sequence until
          /the end of the game.
```

3.1 Board representation and move generation

The chess-board is represented as an array of length 144, with the central 64 elements giving the position on the 8 x 8 board, and the remaining elements containing -1 to indicate that they are off the board. This representation allows move calculation by repeated addition of offsets, with a simple test at each stage to check that the proposed destination square is still on the board. For example, the offsets for a rook's move are +1, -1, +12, and -12, and each of these is repeatedly added to the square number of the rook's initial position to give the moves. More details about how moves are generated with this board representation are given by Kozdrowicki et al (1971).[4] Although it may appear that a one-square border containing -1 - giving an array of length 100 - is all that is needed to detect when a man has reached the edge of the board, a knight would be able to cross such a border, causing unexpected results. In fact only 132 elements are necessary in the array, representing a 12 x 11 "extended board" (Gillogly, 1972)[3], but we have found that the 12 x 12 extended board is easier to deal with and facilitates program writing and debugging. The men on the board are indicated by numbers 1 - 16 (for White) and 17 - 32 (for Black), so that, for example, the WQR can be distinguished from the WKR. The board in the initial position is shown below:

-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	17	18	19	20	21	22	23	24	-1	-1
-1	-1	25	26	27	28	29	30	31	32	-1	-1
-1	-1	0	0	0	0	0	0	0	0	-1	-1
-1	-1	0	0	0	0	0	0	0	0	-1	-1
-1	-1	0	0	0	0	0	0	0	0	-1	-1
-1	-1	0	0	0	0	0	0	0	0	-1	-1
-1	-1	9	10	11	12	13	14	15	16	-1	-1
-1	-1	1	2	3	4	5	6	7	8	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1

Moves are generated by adding offsets as outlined above, bearing in mind that

(i) a man cannot move off the board,

(ii) a man cannot move to a square occupied by another man on his side,

(iii) for pawns, knights, and kings, the offsets must be added once only,

(iv) pawns in their initial position have a special move available.

This algorithm generates all "normal" moves (but not castling, etc). Because of the importance of the relationship of "attacking" for the connectivity analysis, and the similarity of this

relationship to that of "moving", the routine which generates moves also lists all legal attacks. All moves are attacks, except pawn moves, which are never attacks. In addition, a man can attack a square occupied by another man on his side. Pawn attacks are generated by adding offsets different from those used for pawn moves.

In order to find all legal moves, each move generated by adding offsets must be tested to see if a check results. If castling is still legal, and the squares between the king and rook are empty and not attacked, the appropriate castling move is added to the list of legal moves. En passants are spotted by examining the previous move in conjunction with the current list of attacks. The possibility of pawn promotion is also considered. The result of all this is a list of legal moves, each stored as 4 computer words:

(SOURCE SQUARE, DESTINATION SQUARE, X, Y)

where X and Y are only used for castling moves, pawn promotion, etc..

When moves are read in from the teletype or the disk file containing a chess game, they are decoded from International Algebraic Notation into the 4-word internal move representation, using standard methods of syntax analysis. Checks are made for obvious errors, and then the move generation routine is called and the list of moves is searched for the proposed move. To make a move on the chess-board, one can either place it on a move stack in a 5-word reversible representation (the fifth word specifies the man taken in the move, if any) so that the previous board position can be recovered by unstacking, or empty

the stack, make the move, and place it at the base of the stack (so that the previous move is always available to check en passant legality).

3.2 Connectivity analysis

The relationship, Γ_W, of "attacking", computed in the form of a list of squares attacked by each man, provides the basis of the connectivity analysis. To find the connectivity matrix CONN, where

CONN (I, J) = number of squares attacked by both man I and man J, minus one,

the lists of squares attacked by the men are compared in the obvious way.

Computation of the structure vector for White's view of Board is a rather tricky matter. For each Q-level from zero up, a routine is called which returns the number of simplices at that level. The array of numbers obtained at each Q-level is the structure vector. If any component of the structure vector is zero, all higher components will be zero too. To determine the number of simplices at any Q-level, one starts with an array of length 16 - an element for each man on the side - which is destined to hold a simplex identification number for each man. A man I is allocated a new simplex identification number if

(i) CONN (I, I) ≥ Q-level,

and (ii) he has not already been allocated a simplex identification number.

*If he is allocated a new simplex identification number, CONN (I, *) and CONN (*, I) are scanned for elements at least as big as Q-level, and for any that are found, the man who is connected to I at that level is allocated the current simplex identification number. A similar search must be carried out for all the new men who are attached to the simplex. When no more men in the simplex can be found, the next man on the side is examined and allocated a new simplex identification number if he satisfies the above two criteria. Once all men on the side have been given a simplex identification number, the structure vector component at that Q-level is found by counting how many distinct identification numbers have been issued.*

An example of a board position, the corresponding connectivity matrix for White, and the simplices at each Q-level, is given below, for the complex $K_W(S)$.

Board position

8	BR	BN	BB	**	BK	BB	**	BR
7	BP	**	**	BP	**	BP	BP	BP
6	**	**	**	**	**	BN	BQ	**
5	**	BP	**	**	**	WN	**	**
4	**	**	**	**	WP	BP	WP	WP
3	**	**	**	WP	**	**	**	**
2	WP	WP	WP	**	**	**	**	**
1	WR	WN	WB	WQ	**	WK	WR	**
	a	b	c	d	e	f	g	h

Connectivity (shared-face) matrix for $K_W(S)$

QR	QN	QB	Q	K	KB	KN	KR	P	P	P	P	P	P	P	P	
1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	QR
	2	0	0	-	-	-	-	-	1	-	-	-	-	-	-	QN
		3	0	-	-	0	-	-	-	-	-	-	-	-	-	QB
			8	1	-	-	1	-	-	0	-	-	-	-	-	Q
				4	-	-	0	-	-	-	-	-	-	-	-	K
					-	-	-	-	-	-	-	-	-	-	-	KB
						7	0	-	-	-	-	-	-	-	-	KN
							4	-	-	-	-	-	-	-	-	KR
								0	-	0	-	-	-	-	-	P
									1	-	-	-	-	-	-	P
										1	-	-	-	-	-	P
											1	-	-	-	-	P
												1	-	0	-	P
													-	-	-	P
														1	-	P
															0	P

q-level	q-connected components	Q_q
0	(QR) (QN QB Q K KN KR P P P) (P) (P P) (P)	5
1	(QR) (QN P) (QB) (Q K KR) (KN) (P) (P) (P) (P)	9
2	(QN) (QB) (Q) (K) (KN) (KR)	6
3	(QB) (Q) (K) (KN) (KR)	5
4	(Q) (K) (KN) (KR)	4
5	(Q) (KN)	2
6	(Q) (KN)	2
7	(Q) (KN)	2
8	(Q)	1

Structure vector for $K_W(S)$

$$\underline{Q} = \{\overset{8}{1}, 2, 2, 2, 4, 5, 6, 9, \overset{0}{5}\}$$

3.3 A simple valuation procedure

Before the program can analyse the structures positionally it needs to compute the mapping

$$\underline{s\ val}\ (S_i)$$

for each S_i, (v. section 2.1). This mapping is independent of what piece occupies S_i. However we shall introduce, in 3.4, what

might be called a tactical value, tact (S_i), and this will involve some valuation of any occupying piece.

Using the relations I, II of section 2.3 we set about finding c val (W_j), p val (W_i), and st val (S_j).

We allow that st val (S_j) depends upon

(i) whether S_j is in the central block,

(ii) the value p val (B_k) of any Black man B_k occupying S_j,

(iii) if W_i is a pawn, whether S_j is on the 7th or 8th rank,

and is best indicated by giving the numbers {st val (S_i)}, for White, of the squares in the mode [0, 0] position.

If attacking man is a piece,

8	16	10	9	29	10	9	10	16
7	4	4	4	4	4	4	4	4
6	2	2	2	2	2	2	2	2
5	2	2	4	6	6	4	2	2
4	2	2	4	6	6	4	2	2
3	2	2	2	2	2	2	2	2
2	2	2	2	2	2	2	2	2
1	2	2	2	2	2	2	2	2
	a	b	c	d	e	f	g	h

If attacking man is a pawn,

8	30	24	23	43	24	23	24	30
7	12	12	12	12	12	12	12	12
6	2	2	2	2	2	2	2	2
5	2	2	4	6	6	4	2	2
4	2	2	4	6	6	4	2	2
3	2	2	2	2	2	2	2	2
2	2	2	2	2	2	2	2	2
1	2	2	2	2	2	2	2	2
	a	b	c	d	e	f	g	h

We then get $\underline{p\ val}\ (W_i)$ *by relation II, and we use the hypothetical relation*

$$\underline{c\ val}\ (W) \cdot \underline{p\ val}\ (W) = 200$$

to obtain $\underline{c\ val}\ (W_i)$.

The positional analysis with which we have experimented to-date considers each legal move by White and then examines the consequent changes (increases) in (only) the following features.

(i) dim $K_W(S)$, *or the maximum q-value;*

(ii) $-Q_o$, *the minus sign being justified in sections 2.0 and 2.1;*

(iii) $-Q_1$;

(iv) $\underline{c\ val}\ (\underline{pos}\ BK)$;

(v) $\sum_i \underline{c\ val}\ (S_i)$ *where* $BK = \langle \ldots S_i \ldots \rangle$ *in* $K_B(S)$;

(vi) $\sum_j \underline{c\ val}\ (\underline{pos}\ B_j)$, *for all Black men* B_j $(\neq BK)$;

(vii) $\sum_i \underline{c\ val}\ (S_i)$ *where* S_i *is a centre square.*

The side's control over any set of squares is just the sum of the positional values of the squares for the side.

At present the program simply sums the scores under these 7 headings to give an overall positional score for a move. There is obviously great scope for improvement over this, but even with such a naive method of scoring, significant correlation with chess-players' positional judgement is obtained.

3.4 Loss/gain tactics

The program as described so far is a weak tactician. It is designed to score moves on a positional basis, taking into account the control over important sets of squares and the co-operation of men on the board. It neglects forcing moves and is oblivious to material loss and gain. Because positional features of the game cannot be completely divorced from the tactical viewpoint - for example, experienced players simply do not consider moves which are tactically unsound when asked to make a positional judgement - an elementary material loss/gain calculation has been incorporated, and the program orders moves primarily according to material exchange, and only secondarily from a positional analysis. (Clearly a less extreme balance should be struck here. Material sacrifices for positional gain are not uncommon in master chess).

We take the tactical value of a man, W_i, as $1+\max \hat{q}\ (W_i)$, in $K_W(S)$. This gives a measure of the potential of the geometrical

horizon of W_i, whereas the piece value, $\underline{p\ val}\ (W_i)$, reflects the man's actual worth in the present board position. The value of a bishop is halved to account for the fact that it can potentially control only half of the board's squares. This gives the values (c.f. section 2.0)

K	Q	R	N	B	P
8	27	14	8	7	2

(For investigation of material exchanges, the king is assigned an arbitrary value of 1000).

The tactical value of a square is then given by the mapping

$$\underline{tact} : S \rightarrow J$$

where

$$\underline{tact}\ (S_i) = \underset{i,\ j}{minimax} \{\underline{tact}\ (W_j), \underline{tact}\ (B_K)\}$$

where $\underline{tact}\ (W_j)$ is the above (special) case of $\underline{p\ val}\ (W_j)$, and

$$S_i = \langle \ldots W_j \ldots \rangle \text{ in } K_S(W)$$

$$= \langle \ldots B_K \ldots \rangle \text{ in } K_S(B).$$

The tactical value of squares in Board positions taken from actual chess games is almost always zero, but the positional analysis often suggests moves which, if made, would result in some squares having negative values for the side under consideration. By first ordering moves from this simple material viewpoint, this situation is usually avoided.

3.5 Some comparisons with actual games

Using the valuation procedures discussed above we obtained the following positional assessment of actual games.

(A) Morphy v Duke of Brunswick et al. (1858)

	Game score	Positional ranking of Morphy's moves	
1.	e2 - e4 : e7 - e5	2	1 = d2 - d4
2.	N - f3 : d7 - d6	8	1 = d2 - d4
3.	d2 - d4 : B - g4	1	
4.	d4 * e5 : B * f3	1	
5.	Q * f3 : d6 * e5	2	1 = g2 * f3
6.	B - c4 : N - f6	4	1 = c2 - c4
7.	Q - b3 : Q - e7	24	1 = c2 - c3
8.	N - c3 : c7 - c6	4	1 = f2 - f3
9.	B - g5 : b7 - b5	5	1 = f2 - f4
10.	N * b5 : c6 * b5	1	
11.	B * b5+ : N(b8) - d7	2	1 = Q * b5
12.	0-0-0 : R - d8	3	1 = f2 - f4
13.	R * d7 : R * d7	1	
14.	R - d1 : Q - e6	2	1 = f2 - f4
15.	B * d7+ : N * d7	2	1 = R * d7
16.	Q - b8+ : N * b8	8	1 = R * d7
17.	R - d8 mate.		

A total of 82% of Morphy's moves fall in the first 5 positional rankings, and 70% fall in the first 3.

(B) Anderssen v. Kieseritsky (1851), the Immortal Game

	Game score	Positional ranking of White's moves	
1.	e2 - e4 : e7 - e5	2	1 = d2 - d4
2.	f2 - f4 : e5 * f4	2	1 = d2 - d4
3.	B - c4 : Q - h4+	9	1 = d2 - d4
4.	K - f1 : b7 - b5	1	
5.	B * b5 : N - f6	2	1 = B - b3
6.	N - f3 : Q - h6	3	1 = Q - f3
7.	d2 - d3 : N - h5	1	
8.	N - h4 : Q - g5	26	1 = K - f2
9.	N - f5 : c7 - c6	4	1 = K - f2
10.	g2 - g4 : N - f6	2	1 = h2 - h4
11.	R - g1 : c6 * b5	2	1 = B - c4
12.	h2 - h4 : Q - g6	10	1 = a2 - a4
13.	h4 - h5 : Q - g5	9	1 = a2 - a4
14.	Q - f3 : N - g8	6	1 = a2 - a4
15.	B * f4 : Q - f6	6	1 = a2 - a4
16.	N - c3 : B - c5	4	1 = a2 - a4
17.	N - d5 : Q * b2	2	1 = g4 - g5
18.	B - d6 : Q * a1+	5	1 = B * b8
19.	K - e2 : B * g1	1	
20.	e4 - e5 : N - a6	25	1 = B * b8
21.	N * g7 : K - d8	14	1 = N(b5) - e7
22.	Q - f6+ : Resigns	4	1 = c2 - c4

A total of 64% of Anderssen's moves fall in the first 5 positional rankings, and 45% fall in the first 3; White's play emerges as highly tactical by this program.

(C) Fischer v.Petrosian (1971)

	Game score	Positional ranking of White's moves	
1.	e2 - e4 : c7 - c5	2	1 = d2 - d4
2.	N - f3 : e7 - e6	12	1 = d2 - d4
3.	d2 - d4 : c5 * d4	1	
4.	N * d4 : a7 - a6	2	1 = Q * d4
5.	B - d3 : N - c6	27	1 = c2 - c3
6.	N * c6 : b7 * c6	1	
7.	0 - 0 : d7 - d5		
8.	c2 - c4 : N - f6	1	
9.	c4 * d5 : c6 * d5	1	
10.	e4 * d5 : e6 * d5	1	
11.	N - c3 : B - e7	4	1 = f2 - f3
12.	Q - a4+ : Q - d7	6	1 = f2 - f3
13.	R - e1 : Q * a4	3	1 = f2 - f3
14.	N * a4 : B - e6	1	
15.	B - e3 : 0 - 0	21	1 = f2 - f3
16.	B - c5 : R(f8) - e8	2	1 = N - c5
17.	B * e7 : R * e7	1	
18.	b2 - b4 : K - f8	17	1 = f2 - f3
19.	N(a4) - c5 : B - c8	3	1 = f2 - f3
20.	f2 - f3 : R(e7) - a7	2	1 = b4 - b5
21.	R(e1) - e5 : B - d7	10	1 = b4 - b5
22.	N * d7 : R * d7	1	
23.	R - c1 : R - d6	6	1 = b4 - b5
24.	R(c1) - c7 : N - d7	4	1 = b4 - b5
25.	R - e2 : g7 - g6	4	1 = R - g5

	Game score	Positional ranking of White's moves	
26.	K - f2 : h7 - h5	5	1 = b4 - b5
27.	f3 - f4 : h5 - h4	12	1 = g2 - g4
28.	K - f3 : f7 - f5	3	1 = g2 - g3
29.	K - e3 : d5 - d4+	17	1 = g2 - g4
30.	K - d2 : N - b6	2	1 = K - f3
31.	R - e7 : N - d5	2	1 = R(c7) - e7
32.	R - f7+ : K - e8	2	1 = R(c7) - e7
33.	R - b7 : N * b4	3	1 = R(c7) - a7
34.	B - c4 : Resigns	16	1 = R(b7) - a7

A total of 68% of White's moves fall in the first 5 positional rankings, and 56% fall in the first 3.

3.6 Research prospects

The positional criteria used so far, and illustrated in the previous section, are characterised by the following features.

(1) Restriction to consideration of ΔQ_o, ΔQ_1, Δ dim K, when the argument shows that the other Q_t values in $\underline{Q}$ have profound positional influences.

(2) Restriction to a consideration of $K_W(S)$ and $K_S(W)$ so as to improve certain geometrical properties of White's position. Clearly it would be desirable to assess the possible changes in the Black position, by considering $K_B(S)$ and $K_S(B)$. A good move for White will presumably improve White's geometry whilst at the same time cause a deterioration in Black's structure.

(3) Restriction to a particular mode [I, J], that is to say, without any effective "look ahead" analysis. Future research must clearly take into account the overall positional features over a sequence of moves. Drastic changes in the abstract geometrical structures might well be the result of "give-and-take" over 3 or 4 moves by White. Thus we need to allow for the positional advantages which can accrue by way of moves which are apparently tactically chosen. But even here, and referring back to the discussions in sections 2.0 and 2.1, we begin to see how the line between "tactical" and "positional" becomes blurred.

But this approach to the game means, above all things, that the emphasis is no longer on tree-searching. Positional features must be used to reduce the conventional tree-search to manageable proportions. Further study of the connectivity structures of the various complexes, such as the search for specific chains of q-connection or the dependence on such chains of the mappings c val, s val, p val, st val, should greatly assist in this aim. It is to be expected that during the course of a game these mappings must themselves vary a great deal, and so we must search for the dependence of c val etc. on the structures $K_W(S)$ etc. which are linked to the modes. This would allow the possibility of the positional criteria being influenced by the tactical possibilities, and therefore of the computer (as player) being able to change its mind about the positional goals as the game proceeds.

Furthermore it is obviously going to be of great help if chess masters can be persuaded to help in the research - if only by

ranking the positional motifs in a few hundred positions. So far there has been an encouraging response to this cry for help, although we have not yet reached a level of organised co-operation with those players who are anxious to help.

APPENDIX A

International Algebraic Notation

A chess move in international algebraic notation has the following form:

$$\langle move \rangle ::= \langle source \rangle \langle operation \rangle \langle square\ designation \rangle \langle check\ indication \rangle \quad (1)$$

where

$$\langle source \rangle ::= \langle piece \rangle \mid \langle square\ designation \rangle \mid \langle piece \rangle (\langle square\ designation \rangle)$$

$$\langle piece \rangle ::= R \mid N \mid B \mid Q \mid K$$

$$\langle square\ designation \rangle ::= A1 \mid A2 \mid \ldots \mid A8 \mid B1 \mid \ldots \mid H8$$

$$\langle operation \rangle ::= - \mid *$$

$$\langle check\ indication \rangle ::= + \mid \varepsilon$$

The following special move types are also allowed:

$$0 - 0 \quad (2)$$

$$0 - 0 - 0 \quad (3)$$

$$\langle move \rangle \nabla EP \quad (4)$$

$$\langle move \rangle \nabla \langle promotion \rangle, \text{ where } \langle promotion \rangle ::= *R \mid *N \mid *B \mid *Q \quad (5)$$

(∇ denotes a blank; ε denotes the null string)

*The interpretation of a string of type (1) is that the man indicated by ⟨source⟩ makes the ⟨operation⟩ on the destination ⟨square designation⟩. If the source man is a pawn, he is specified by giving the ⟨square designation⟩ of the square he occupied before the move; if he is a piece, then the piece's name alone is used unless ambiguity results, in which the ⟨square designation⟩ must also be specified. The ⟨operation⟩ can be either " - ", which indicates that the designation square was unoccupied prior to the move, or " * ", which indicates that it was occupied by one of the opponent's men. The ⟨check indication⟩ is " + " if and only if the move results in a check.*

Type (2) and (3) moves indicate castling on the King's side and on the Queen's side, respectively. A type (4) move signifies capturing en passant, and type (5) refers to pawn promotion, the new piece being specified explicitly as ⟨promotion⟩.

REFERENCES

[1] Atkin, R.H., *Multi-dimensional Structure in the Game of Chess, Int. J. Man-Machine Studies (1972), 4, 341-362.*

[2] Hilton, P.J. & Wylie S., *Homology Theory, (1960), Cambridge University Press.*

[3] Gillogly, J.J., *The Technology Chess Program, (1972). Artificial Intelligence 3:145 - 163.*

[4] Kozdrowicki, E.W., Licwinko, J.S., Cooper, D.W. *(1971) Algorithms for a minimal Chess Player. Int. J. Man-Machine Studies 3(2), 141 - 166, April.*

[5] Atkin, R.H., *From Cohomology in Physics to q-connectivity in Social Science, Int. J. Man-Machine Studies (1972), 4, 139.*

[6] Lasker, E., *A Manual of Chess, Constable (1932); Dover (1947).*

5

A KNOWLEDGE BASED PROGRAM TO PLAY CHESS END-GAMES

by

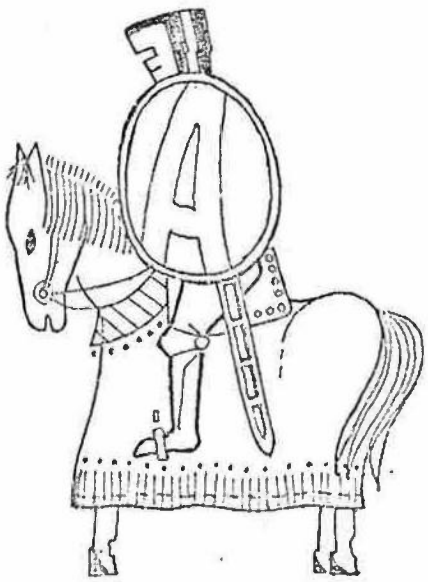
S Tan

Department of Machine Intelligence
University of Edinburgh
8 Hope Park Square
Edinburgh
EH8 9NW

"A little (knowledge) is a dangerous thing.
Drink deep or taste not the Pierian spring.
There, shallow draughts intoxicate the brain
But drinking largely sobers us again."

- ALEXANDER POPE

Essay on Criticism



Editor's Note -

Must be read in conjunction with (Tan, 1972). The psychological studies described by Bond and the connectivity described by Atkin are mainly concerned with (and most relevant to) the middle game. Clusters of pieces (chunks) and connectivity become less evident; the decisions are more critical, in the end game.

Note that Tan is not concerned with how a program may reach an end game but with the problems of representing and using chess knowledge for the very deep analyses which must be performed.

An outline is given of a program to solve endings with king and two pawns vs king and bishop. The approach is basically the same as in (Tan, 1972) except that a more flexible interpreter is used this time. Also added are extensions of the notions of predicates, actions and patterns, as well as the use of goals, simple cross-communication between branches of the analysis tree and the extraction of plans from analysis trees.

A. PROBLEM AND APPROACH

This work is a continuation and extension of the knowledge-based approach described in (Tan, 1972). Our concern is with the problem of representing and using chess knowledge, not how knowledge is acquired. The emphasis on knowledge is important in view of the inadequacy of the classical Shannon-Turing game-playing framework: game-tree, evaluation, minimax etc. We envisage programs that play almost always correctly (never throw away a win or a draw) in their problem domains, which means having to make very deep analyses (the domain we are tackling now is that of king and 2 pawns vs king and bishop, it contains studies where analyses of ply-depth 20 are necessary, for king, rook and pawns vs king and rook the corresponding number is about 40), and must therefore be radically selective in generating moves. Variations of the Shannon-Turing type of programs may be able, assuming that a good evaluation function can be found, to find good moves, but that would be far from sufficient for solving end-game studies correctly.

In the following, representation and use of knowledge are considered inseparable, representation is specified by giving a virtual 'chess machine' which acts as an interpreter. Given an input board situation, this interpreter will then 'parse' it to produce the move to be played, plans and a prediction of the value of the situation. The 'parsing' process is directed by a network representation of the program's knowledge of playing methods. Some of the problems encountered in designing such a interpreter are:-

(i) Since specifying an interpreter is in effect developing a mini-theory of end-games, one may ask what sorts of things are allowed in the ontology of the theory (does the theory accomodate plans, threats, intentions, episodes, scenes, demons, etc) and what are the relations between these sorts (eg how are goals used in a situation-action production system).

(ii) How do we choose the primitive actions of the interpreter, in other words, how much compilation should be done (in (Tan, 1972) the whole program is compiled, the virtual POP-2 machine is the chess machine). Shall we adopt a multipass interpreter that can

account for the phenomena of progressive deepening (De Groot, 1965), and if so, how do we handle communication between the different passes (besides the problem of communication between branches of the analysis-tree).

(iii) How much advice, deductive power should the program have, what search strategy and evaluation function should be used.

There are many more question that can be asked, but here we can only attempt to answer a few of them with respect to the problem domain we have in mind. No doubt there are no general answers to most of the above questions; compiler-interpreter, deduction-search, advice-search, evaluation-look-up, backward search-forward search etc, are pairs of items that are often traded-off inter-changeably.

The attitude taken here is to try to proceed from the simple to increasing complexity, and to be flexible and delay ultimate decisions when further clarification or experimentation are necessary.

The next section outlines the typology of the theory, it is relatively rich compared to existing chess programs, but not as rich as found in chess psychology (compare the De Groot op-cit). We have not made provisions to include progressive deepening (which may be implemented serially or in parallel by coroutines) at present, but De Groot may be right in pointing out the importance of it for computer programs (De Groot, 1965, p 401). Kotov [1], on the other hand, who is interested in teaching 'human beingsto analyse with the accuracy of a machine', argues that a branch of the analysis-tree should never be searched more than once, and only lack of confidence can make us do otherwise.

There is no explicit deductive power at present, other than those that can be implicitly embedded in the program's playing methods. A simple depth-first strategy augmented by a preliminary breadth-first search is adopted. For the last mentioned search, an evaluation function similar to the one used by (Newell et al, 1959) is adopted, the value of a position is a feature vector. Unlike Newell et al however, there is not a prior lexicographic ordering of the vectors, since it seems to be counter-intuitive; possibilities of trading-off material for space or development, pawn structure for a bishop etc, which is certainly the essence of what chess is about, being excluded.

B. BRIEF OVERVIEW

In this section we will only give an informal description of the different categories of object in the theory, their relations to each other and their properties. No attempt is made to present a formal theory. Most examples given apply to the case of 2 pawns vs a bishop, the pawns are always white. Some of the assumptions made below are somewhat arbitrary, they are made with this restricted problem domain in mind.

1. Situations

A situation is a data-structure containing all the board information: board position, who is to move, and sometimes a little history (has the king been moved, what was the last move etc), clock etc. In the following,

situations will be distinguished from states of the interpreter (see 7 below), and we assume that no history and clock are recorded in a situation. Thus the interpreter will not make use of knowledge of the opponent's last move (eg if piece captured, try recapture) in selecting its reply. It does not however consider every situation presented as new: it has plans and recognizes repetition of situations.

2. Concepts

A concept is a POP-2 function describing general relations between pieces, squares, numbers etc. Examples: rank, distance as number of king moves, block-distance, critical square, breakthrough square, different kinds of blockades, doubles, isolated and connected pawns, pin, mobility, center, shelter, good bishop, queen side majority etc. We restrict ourselves here to simple static concepts, there are no concepts which involve dynamic search, succession of states or which refer to the state of the interpreter (no concepts of overloading, desperado, encirclement, Zugzwang, initiative etc) though it is possible to have overloading as a predicate, encirclement as an action or plan etc . It is assumed above that critical squares, breakthrough squares, shelters etc, can be determined in a static manner, though in general they may be dynamic.
Concepts are used in predicates and goals.

3. Predicates

Predicates are POP-2 functions defining partial functions from states to truth-values. This is an extension of the early notion of a predicate as a partial function from situations to truth-values used in (Tan, 1972) Examples:-

(a) mate, stalemate, check, can-advance, can-capture, etc;
(b) those associated with concepts directly: has-critical-square, is-blockaded, etc;
(c) those associated with patterns: match (pattern);
(d) those referring to the state of the interpreter: has-no-plan, has-occurred-before, etc;
(e) the most important predicates are those connected with lookahead searches, they make recursive calls to the program's body of knowledge, eg: the predicate: 'starting with this situation, removing the following pieces, using all the chess-knowledge that I have, applying the following action, white will win'. With the exception of the action try (below), this is also the only place where (full or partial, forward or backward) lookahead searches can occur.

4. Actions (or action-schemes)

Partial unary operations on situations or states are called actions. Actions can be POP-2 functions or represented as a network in the same way as the whole move finding routine. They may be primitive (eg actions corresponding to moves, dummy operations, update white-list etc); or built-up from predicates and primitive actions by

conditionals (eg support, approach, letpass etc).
Associated to patterns, there are actions of the form: try (pattern, x), which means: try to reach (usually backward search) the pattern in at most x moves, if x=o the pattern must contain a suggestion on what is to be done.

5. Goals

Goals are defined by (Newell and Simon, 1972, p 807) by the characteristics:-

(i) 'a goal carries a test to determine when some state of affairs has been attained';
(ii) 'a goal is capable of controlling behaviour under appropriate conditions. The control takes the form of evoking patterns of behaviour that have a rational relation to the goal - ie methods for attaining the goal'.

The goals we have at present satisfy the first characteristic and the second to some extent. They are related directly to concepts, eg:
for black : blockade (there are different strengths of blockades), mobility of the bishop;
for white : minimize distance (white king, pawn); minimize distance (white king, bishop) etc.

The set of goals, also called feature vector, is only partially ordered, it is used for preliminary elimination of moves in a breadth-first search up to depth one. A goal in itself does not propose actions (that is why it does not quite satisfy the second characteristic), but used within an action routine it does control the choice of actions to be taken.

At present there are no mechanics for activating/de-activating and weightings of goals.

6. Patterns

There is a stock of important didactic patterns that must be recognised quickly by the program. These patterns may be geometric or defined by more general predicates. They may or may not have actions associated to them, and are used as an action: try (pattern, x) or as predicate: match (pattern).

The stock of patterns is considered fixed; non-permanent patterns created during analysis are not allowed at present.

7. States

The state of the interpreter is given by a stack of situations used to keep track of recursion, and an environment in the form of an analysis-tree. The stack is hidden and never referred to by the user.

8. Analysis-trees

This is the tree of moves considered in the analysis. Attached to its nodes we have a white-list (list of good moves for white), a black-list (list of good moves for black), and the value (win, lose or draw) of the situation corresponding to the node (if known). The white and black lists serve for communication between branches of the analysis-tree, a good killing move in one branch is often good in other branches as well (compare McCarthy's killer list, (De Groot, 1965, p 395)). The analysis-tree is the most dynamic part of the interpreter, it is grown and pruned most of the time.

9. Plans

At present there are only concrete plans extracted from the analysis-trees by pruning the insignificant branches. These plans are used to anticipate the opponent's move.

Use of abstract (containing action-schemes rather than the actual moves) plans (eg breakthrough, distribution of effort between king and bishop) during the analysis itself are being considered.

10. Network

As mentioned earlier, the interpreter is directed by a network representing the program's chess knowledge. Its nodes are records consisting of a predicate, action 1, LLINK, action 2 and RLINK, where LLINK and RLINK are pointers to other nodes. A node implies an instruction; if the predicate is true, do action 1 then follow LLINK, else perform action 2 and follow RLINK. Example of a node (omitting links): 'in case of two connected pawns, where they are abreast, if we decide to push, advance the pawn which is not on the same colour as the bishop' (Fine).

LOCAL REFERENCE

[1] Kotov A. Think like a grandmaster, Batsford 1972.

6

O B S E R V A T I O N S

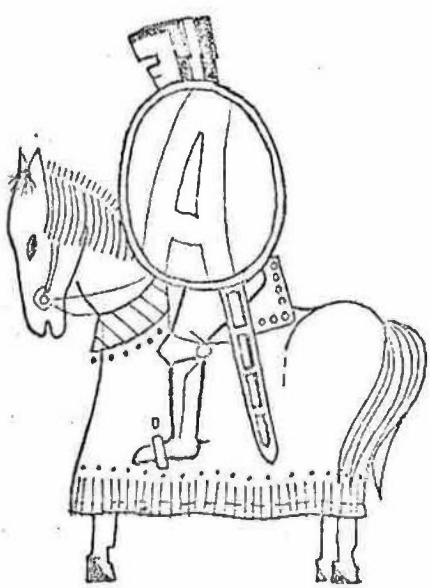
by

R Malik

107 North End Road
London
NW11

"There is nothing more difficult to take in hand, more perilous to conduct, or more uncertain in its success, than to take the lead in the introduction of a new order of things."

- MACHIAVELLI



Editor's Note -

Rex Malik is a writer who specialises in the systems and computer sciences. He is also Senior Research Associate and technical author with Professor Gordon Pask and System Research Limited.

This report is the speech I would have given had I not chosen to cut it short and lead a discussion! It is nowhere near as abrasive as my remarks at the conference, and of course being rewritten, added to, and amended to after the event probably puts my remarks into a more coherent, not to say more elegant context. I do not now, as I had to at the conference, worry about the problem of stepping out from behind the typewriter and facing an experienced audience without the shelter of editors, cold print or the unanswerable at-this-time microphone.

I must once again state that having spent a day listening to the speakers and the various points of view put forward, I was struck (with respect to my fellow speakers) with the low level of the discussion. I had better qualify what I mean by this immediately. It seems to me that ten years ago, even five years ago, the conference would have been generally discussing problems at and beyond the front end of 'art'. But in the context of today, much of the discussion was out of date, and concerned with subject matter which I would have expected an audience with the degree of expertise present to already have been familiar with; even bored with.

Against this, one must set the argument basically raised by Alex Bell that the conference was intended to bring together people, many of whom had never met, in the hope that from it something useful would spring, contacts would be made, and the place of computer chess in the scheme of things would perhaps be more closely defined. Given that the people had not been brought together before, any starting point must be useful; it gives some indication of what people know, as well as what they do not know.

My observations on the meeting come into three groups. First, the general atmosphere. It seems to me that the general interest level displayed was quite high and on the right lines. I have sat as an interested spectator writer on the sidelines of attempts to play computer chess for many years, and what I found striking was that (with what I would call first generation 'technical' knowledge) the audience should display second generation attitudes and be concerned with second generation problems. The concern seemed to be with making computers play 'people' chess, not machine to machine chess. Though it

would no doubt be of general interest if one computer chess program played well against another, the focus was on chess as a human activity and what stems from it. True, there will no doubt be some quiet jollification when and if a chess program does beat a Grand Master, it is only to be expected, but this seemed a peripheral and non-central matter. And this is an improvement, indeed if I am right and this holds across the field it denotes a major departure from historic pre-occupations. One thinks of John von Neumann and his famous predictions, goes back to Babbage; indeed it is possible to go back beyond even this, though as someone who has been involved in research into the history of computing, including mere idle speculation, I can find few traces before this which are not of the Golem or Delphic oracle variety.

Second, and also peripheral to the meeting, which in some ways I find unfortunate, the question what we should now do together was not answered at all satisfactorily. It may be that people wish to continue quite independently of each other, 're-inventing the wheel' to quote the meeting's most popular cliche. Sixty people goes a long way to dispel this. I do not believe that this was the wish of the meeting, rather that it was due to the fact that the question was never put in a way which it found attractive. But certainly a case was being made almost throughout for some organisation which would encompass those who play chess and are interested in attempts to play it by computer; those computer scientists who find the chess problem one of interest in that it provides professional intellectual satisfaction, and those who regard the chess computer problem as a suitable test-bed with which to test out deeper ideas about how we ourselves approach problems. This grouping of interests does not obviously fit into a computer professionals' society, a chess club or congress, or indeed the 'artifical intelligence' chapter.

It may be that those interested are going to have to sit down together to work out what to do. That could vary widely, but it is quite apparent to me that one thing that needs to be done (and which the SRC might somehow or other usefully undertake) would be to provide something a little more comprehensive than a bibliography which could and should be made available at least as a beginning to the participants. The situation must not occur again that many of those with a serious interest in the subject, whatever their motives, should find themselves in a position where their basic knowledge is such that effectively they still think in terms of the horse and buggy, though some others in the same meeting are already working with jet propulsion. Yet both can find much of what they want to know in the available literature - if they knew where to look.

The starting point for my third set of comments arises from the remarks made both by Alex Bell and Peter Kent. The first exhibited general dissatisfaction with the level of chess knowledge displayed by the programmers and the chess knowledge obtained from players, saying in effect 'if the chess experts could tell us what to do, we should do it'. The second during his presentation remarked not only that mini max is dead, though it could be inferred that many people had not realised it yet, but that often chess programs made the right moves for the wrong reasons: they might play legal chess, they did not play anything resembling good human chess. This was further discussed by Dr Tan, notably his unchallengeable (and unchallenged

which itself is interesting) comment that the Shannon/Turing framework is inadequate, and that we now need to look for a new one, a search in which he and some others are and have for some time been taking part.

It is not my wish to quote extensively in a report which also contains the original papers. However it does seem to me to be worth pointing out that the remarks concerned with 'where do we go from here' fall into two groups. One group is obviously that of 'chess as played by humans' and what we can learn about how people behave and operate in the context of the world of the chess board, this is also the concern of the field of congnitive studies, including computerised artificial intelligence. Thus Atkin's paper here I consider as of considerable importance; indeed his theory really ought to be tested using not only past games, but also techniques arising out of pattern recognition. One can foresee also some experiments which arise from the notion of over optimism/pessimism in relation to the real strength of the positioned pieces, and its effect on the actual game. That this as a general proposition is true, is obvious; what is not obvious is the elaboration or all the extensions, but certainly there is almost bound to be a connection between this and and the middle aged syndrome, and what that connection is might be both fun and instructive to discover. And for the middle aged syndrome, read also a large number of other problems concerned with operations in the wider outside world.

The point I am making is that a study of the world of chess using computers and computer generated techniques might now turn out to be of some very practical importance in other spheres, and should not be left in the generally bemoaned - at least this seemed to me to be the feeling - situation that the work has generally been done in spare time with minimal machine time available. Thus I should like to see for instance much more intensive work on the De Groot conclusion that Alan Bond discussed; that perception in the case of a Grand Master is almost hard wired. I am well aware that as a general propositon this can be tested elsewhere, but testing it in this field seems to present some interesting advantages, not least in that the world of chess has boundaries which can be sharply delineated.

The second part of this third grouping is concerned with the 'mechanics' of the second generation of chess programs. We are now at the level of I SPY rather than computer chess as it has been hitherto understood, and I SPY is more difficult. And if it is to be properly tackled, then obviously the machine and time requirements go up and we need to examine techniques, whether fashionable or not, which have not seriously been looked at in this field in this country before. Thus we do need to have programs which are more dynamic, which alter as the game progresses. We need not only good threat value tables, might I suggest we also need dynamic threat value tables which adjust according to not just the potentiality of the player across the board but also the actuality. Put in this way of course this is asking for a lot, and there may in fact be other ways of solving the problem, even so it does seem to me at this time to be something worth thinking about if not following up.

We are stopped by the inability of programs to generalise, and not only when they come on an 'amazing fact', by the lack of libraries,

and by the inability of the systems as yet to take a synoptic view of the board. And as was also pointed out, hardly anyone is as yet trying to solve the problem proper by searching backwards and forwards, 'the way that a Grand Master might operate'.

To end, I would say that the conference was immensely worthwhile. It seemed to me to indicate that though it might not have suspected its own existence, there was now a community present. And the creation of that is always the first step in getting something done and moving in any field. As for the SRC and its involvement? Well those in the field are in there mostly in their own time, out of interest if not love. This is not a situation so rare that it can be overlooked, particularly when the problems encountered and the possible solutions might tell us so much about ourselves. It ought to be encouraged, and the first thing might well be more programming and more computer time during official hours. Certainly it is as useful as, if not more than, much of the computing which seems to clutter up the publicly provided systems the country has available.

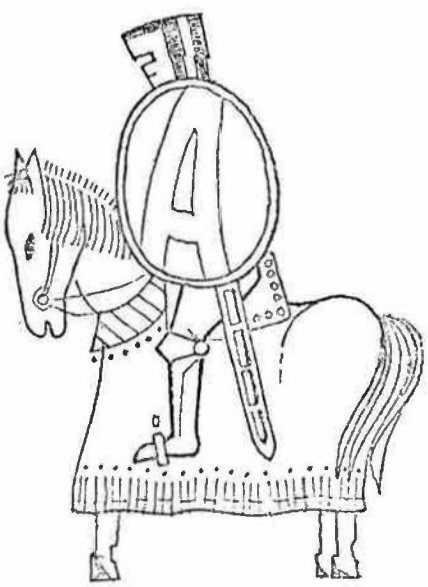
7

D E S C R I P T O R I N D E X

by

A H Bond

Queen Mary College
University of London
Mile End Road
London
E1 4NS



DESCRIPTOR INDEX

(* = recommended)

Overviews and Surveys

(Newell 72)*, (Michie 66), (Newell 59), (Slagle 71), (Bell 72), (Mittman 73)

Report of Experience with Program

(Berliner 70)*

Minimaxing

(Slagle 69)
(Gillogly 72)

α-β Pruning

Description	(Newell 59)
Functional description	(Edwards 63)
Theoretical bound	(Slagle 69)*
Dynamic ordering	(Slagle 69)

Dead Position

(Strachey 59)
(Good 65)
(Greenblatt 67)
(Berliner 70)

Plausible Move Generators

(Bernstein 58), (Newell 59), (Newell 72), (Greenblatt 67)

Data Structures for Chess

Newell and Prasad	(Newell 63), (Baylor 66)
Greenblatt	(Cerf 69)
Scott	(Scott 69)
General	(Williams 65)

Ordering by Shallow Search

(Samuel 67), (Scott 69)

Goal Seeking

(Newell 72), (Baylor 66)

Using the 'No Move' Move

(Baylor 66)*

Particular Chess Programs

Bernstein	(Bernstein 58)
Newell	(Newell 55), (Newell 59), (Newell 72)
Kotok	(Kotok 62)
Greenblatt	(Greenblatt 67), (Cerf 69)
Scott	(Scott 69)
Gillogly	(Gillogly 72)
Berliner	(Berliner 70)
Slate and Atkin	(Slate 70)
Adelson-Velsky	(Adelson 66)
Zobrist-Carlson	(Zobrist 73)
Kozdrowicki-Cooper	(Coko 73)

A Legal Move Generator in Algol 60

(Bell 70)

Endgame Players

(Baylor 65,66), (Hubermann 68)
(Tan 72)

Suggested Chess Programs

Means ends reasoning	(Pitrat 68 and 71)
Humanoid	(De Groot 64 and 65)
Method of horizons	(Botvinnik 70)

Pattern Directed Play (GO)

(Zobrist 69)

Evalution Function Based on Pattern Recogniser

(Samuel 67)

Player Based on Forcing Patterns

(positional games) (King 71)

Statistical Facts and Approaches

(Good 66)
(De Groot 66, 65)

Formal Approaches

Approach by Set Theoretic Formalism

(Banerji 69, 71)
(Dunning 69)
(Marino 66)

Topological Approach

(Atkin 72)

Reports of Chess Games Played by Computers

(Newell 72)
(Scott 69)
(Good 69)

also SIGART newsletters

Go

(Ryder 71)
(Zobrist 69)
(Thorp 64)
(Thorp 70)
(Good 65)
(Remus 62)

Kalah

(Bell 67)
(Russell 64)

Positional Games

(generalisations of noughts and crosses)

Case Institute Game Player	(Citrenbaum 70), (King 71), (Banerji 69)
Go-Moku	(Elcock 67), (Murray 68), (Konniver 63)
Qubic	(Daly 61)

Card Games

Chemin-de-Fer	(Foster 66)
Bridge	(Carley 62), (Wasserman 71)
Poker	(Findler 71), (Waterman 69)
Black Jack	(Thorp 67)

Draughts

(Strachey 52)
(Samuel 59 and 67)

Hare and Hounds

(Storey 69)

Learning in Game Players

Rote Learning	(Samuel 59), (Slate 70)
Optimisation of coefficients	(Samuel 59 and 67)
Learning of forcing patterns	(King 71), (Elcock 67)
Learning of descriptions	(Popplestone 69), (Newman 65)
Learning of heuristic rules	(Waterman 71)

Psychology of Chess

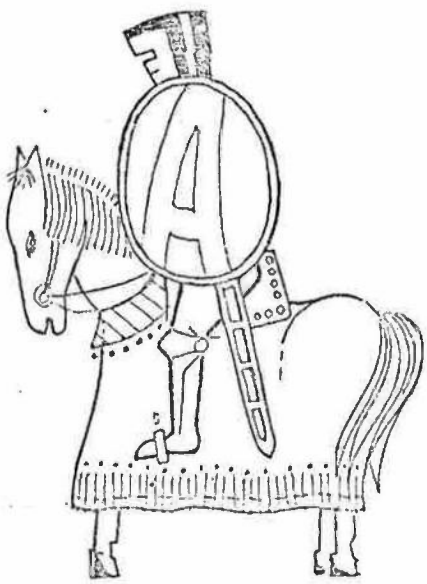
Perception	(Chase 72), (Simon 67, 69)*, (Tikhomirov 66), (Newell 72), (Pushkin 71), (De Groot 65), (Jongman 68)
Search	(Newell 65 and 72)*, (De Groot 65), (Baylor 66), (Scurrah 70), (Simon 62)
Reorganisation	(De Groot 65)*
Memory	(Binet 66)*, (Cleveland 04), (Chase 72)
Psychoanalytic	(Fine 67), (Karpman 37), (Jones 51), (Coriat 41)
Individual Differences	(De Groot 65)

Psychology of Other Games

Halma (Elithorn 70)
Go-Moku (Rayner 58)

8

REFERENCES



REFERENCES

(I possess copies of almost all these references - A H Bond)

(1) Adelson-Velskiy G M, Arlasarov V L and Uskov A G (1966). Programme Playing Chess, Report on Symposium on Theory and Computing Methods in the Upper Mantle Problem.

(2) Atkin R (1972). Multidimensional Structure in the Game of Chess, Int J Man-Machine Studies, 4, 341-362.

(3) Banerji R B (1969). An Overview of Game Playing Programs, Tech Report, Cleveland: Case W R University.

(4) Banerji R B (1969). Theory of Problem Solving, New York: Elsevier.

(5) Banerji R B (1971). Similarities in Games and Their Use in Strategy Construction, Computers and Automata Proceedings of 21st Brooklyn Polytechnic Symposium, 337-357.

(6) Banerji R B and Ernst G W (1971). Changes in Representation which Preserve Strategies in Games, IJCAI2, 651-658; Longer version technical report of same name, Case WRU.

(7) Banerji R B and Ernst G W (1972). Strategy Construction Using Homomorphisms between Games, AI 3, 223-250.

(8) Barker R (1971). Report on Human Game-Playing as Illustrated by the Game of Halma, M Sc Thesis, London: Computer Science Department, Queen Mary College.

(9) Baylor G W (1965). Report on a Mating Combinations Program, SDC Paper SP-2150.

(10) Baylor G W (1966). A Computer Model of Checkmating Behaviour in Chess, Heuristic Processes in Thinking, eds De Groot A D and Reitman W R, Moscow: Nauka.

(11) Baylor G W and Simon H A (1966). A Chess Mating Combinations Program, SJCC, 431-447.

(12) Bell A G (1967). Kalah on Atlas, MI3, 181-194.

(13) Bell A G (1970). How to Program a Computer to Play Legal Chess, CJ, 13, 208-219.

(14) Bell A G (1972). Games Playing with Computers, London: Allen and Unwin.

(15) Berliner H J (1969), Chess Playing Programs, SIGART, 17, 19-20.

(16) Berliner H (1970). Experiences Gained in Constructing and Testing a Chess Program, IEEE Symp System Sc and Cybernetics, Pittsburgh.

(17) Bernstein A and Roberts M de V, Arbuckle T and Belsky M A (1958). A Chess Playing Program for the IBM 704 Computer, WJCC, 157-158.

(18) Bernstein A and Roberts M de V (1958). Computer vs Chess Player, Scientific American, 198, 96-105.

(19) Binet A (1894). Psychologie des Grands Calculateurs et des Jouers d'Echecs, Paris: Hachette.

(20) Binet A (1893 and 1966). Mnemonic Virtuosity: A Study of Chess Players, Genetic Psych Monog, 74, 127-162;
originally prublished as:-
Les Grandes Memoires: Resume d'une Enquete sur le Jouers d'Echecs, Revue des Deux Mondes, 117, 826-859.

(21) Botvinnik M M (1960). One Hundred Selected Games, New York: Dover.

(22) Botvinnik M M (1970). Computers, Chess and Long-Range Planning. London: Longmans.

(23) Brannasky W (1927). Psychologie des Schachspiels, Berlin: De Gruyter.

(24) Carley G (1962). A Program to Play Contract Bridge, M Sc Thesis, EE MIT.

(25) Cerf V and Kline C (1969). The Greenblatt Chess Program, Unpublished term paper at UCLA.

(26) Chase W G and Simon H A (1972). Perception in Chess, Cog Psych. (See also (133).)

(27) Citrenbaum R L (1970). Efficient Representations of Optimal Solutions for a Class of Games, Thesis, Cleveland: Case WR University; Tech Report SRC-69-5.

(28) Clarke M R B (1973). Some Ideas for a Chess Compiler, Artificial and Human Thinking, eds Elithorn A and Jones D, Elsevier.

(29) Cleveland AA (1907). The Psychology of Chess and of Learning to Play It, Am Jour Psych, 18, 269-308.

(30) Coriat I H (1941). The Unconscious Motives of Interest in Chess, Psychoanalytic Review, 28, 30-36.

(31) Daly W (1961). Computer Strategies for the Game of Qubic, M Sc Thesis, Electrical Engineering MIT.

(32) De Groot A D (1964). Chess Playing Programs, 385-398.

(33) De Groot A D (1965). Thought and Choice in Chess, The Hague: Mouton.

(34) De Groot A D (1966). Perception and Memory versus Thought, Problem Solving, ed Kleinmuntz B, New York: John Wiley.

(35) Duncker K (1945 (35)). On Problem Solving, Psych Monographs, 58, no 270.

(36) Dunning C A, Ko H M and Banerji R B (1969). Some Results on Graph Interpretable Games, Tech Report, Cleveland: Case W R University.

(37) Edwards D J and Hart T P (1963). The α-β Algorithm, MIT AI Memo, 30.

(38) Eifermann R R (1972). Computer Analysis of Board Games in the Light of Street Games, NSSHT, ed Elithorn A.

(39) Elcock E W and Murray A M (1967). Experiments with a Learning Component in a Go-Moku Playing Program, MI1, 87-104.

(40) Elithorn A and Telford A (1969). Computer Analysis of Intellectual Skills, Int J Man-Machine Studies, 1. 189-209.

(41) Elithorn A and Telford A (1970). Game and Problem Structure in Relation to the Study of Human and Artificial Intelligence, Nature, 227, 1205-1210.

(42) Findler N V, Klein H, Gould W, Kowal A and Menig J (1971). Studies on Decision Making Using the Game of Poker, IFIP71.

(43) Findler N V (1971). Computer Experiments on the Formation and Optimization of Heuristic Rules, NSSHT, ed Elithorn A.

(44) Fine R (1967). The Psychology of the Chess Player, New York: Dover.

(45) Foster F G (1966). Chemin-de-Fer analysed, Computer Journal, 7, 124-130.

(46) Gillogly J J (1972).The Technology Chess Program, AI 3, 145-164; also: Technical Report 71, Pittsburgh: Carnegie-Mellon.

(47) Good I J (1965). The Mystery of GO, New Scientist, 427, 172-174.

(48) Good I J (1966). A Five Year Plan for Automatic Chess, MI2, 89-118.

(49) Good I J (1969). Analysis of the Machine Chess Game J Scott (White), ICL-1900 versus R D Greenblatt, PDP-10, MI4, 267-269.

(50) Greenblatt Richard D, Eastlake Donald E II and Crocker Stephen D (1967). The Greenblatt Chess Program, FJCC, 801-810.

(51) Greenblatt Richard D, Eastlake Donald E III and Crocker Stephen D (1967). The Greenblatt Chess Program, extended version rough draft.

(52) Greene P (1961). Networks Which Realise a Model for Information Representation, Trans Univ Illinois Symp Self Org.

(53) Harkness and Battell (1947). Article in Chess Review.

(54) Hubermann B J (1968). A Program to Play Chess End Games, Stanford Technical Memo CS 106, C S Department.

(55) Jones E (1951). The Problem of Paul Morphy: A Contribution to the Psychology of Chess, Essays in Applied Psychoanalysis, London: Hogarth.

(56) Jongman R W (1968). Het Oog van de Meester, Amsterdam: Van Gorcum.

(57) Karpam B (1937). The Psychology of Chess, Psychoanalytic Rev, 24, 54-69.

(58) King P F (1971). A Computer Program for Positional Games, Report 1107, Jennings Computer Center, Case WRU.

(59) Kister J, Stein P, Ulam S, Walden W and Wells M (1957). Experiments in Chess, JACM, 4, 174-177.

(60) Koffman EB (1967). Learning through Pattern Recognition Applied to a Class of Games, Systems Research Center Report SRC 107-1-67-45, Case IT.

(61) Koniver D (1963). Computer Heuristics for Five-in-a-Row, M Sc Thesis Mathematics, MIT.

(62) Korschelt O (1966). The Theory and Practice of GO, Tuttle, Ruttland, Vermont.

(63) Kotok A (1962), A Chess Playing Program for the IBM 7090, B S Thesis, MIT, Memo 41.

(64) Kozdrowicki E W and Cooper D W (1973). COKO III, Comm ACM, 16, 411-427.

(65) Lasker E (1960). Go and Go-Moku, New York: Dover.

(66) Levy D N L (1969). Computerised Chess: Prospects, Chess April 22nd 1969, 242-251.

(67) Levy D N L (1971). Computer Chess - A Case Study on the CDC 6600, MI6, 151-164.

(68) Luce R D and Raiffa H (1957). Games and Decisions, New York: Wiley.

(69) Maier N R F (1960). Screening Solutions to Upgrade Quality: A New Approach to Problem Solving under Conditions of Uncertainty, J Psych, 49, 217-231.

(70) Marino L R (1966). Winning and Non-Losing Strategies in Games and Control, Tech Report SRC 91-A-66-36, Case WRU, Cleveland.

(71) McKinsey J (1952). Introduction to the Theory of Games, New York: McGraw Hill.

(72) Michie D (1966). Game Playing and Game Learning Automata, Programming and Non-Numerical Computation, ed Fox L, Oxford: Pergamon.

(73) Mittman B (1973). Can a Computer Beat Bobby Fischer?, Datamation, June.

(74) Murray A M and Elcock E W (1968). Automatic Description and Recognition of Board Patterns in Go-Moku, MI2, 75-88.

(75) Newell A, Shaw J C and Simon H A (1959). Chess Playing Programs and the Problem of Complexity, IPMJ, 2, 320-335; also in: Feigenbaum E A and Feldman J A (1963), Computers and Thought.

(76) Newell A (1955). The Chess Machine, WJCC55, 101-110.

(77) Newell A (1966). On the Representations of Problems, CSRR CMU.

(78) Newell A and Prasad (1963). IPL-V Chess Position Program, Internal Memo No 63, CS Dept, Carnegie-Mellon.

(79) Newell A and Simon H A (1965). And Example of Human Chess Play in the Light of Chess Playing Programs, Progress in Biocybernetics, eds Weiner N and Schade J P, Amsterdam: Elsevier, 2, 19-75.

(80) Newell A and Simon H A (1972). Human Problem Solving, Prentice-Hall.

(81) Newman and Uhr L (1965). Bogart: A Discovery and Induction Program for Games, Proc ACM Conf 65, 176-186.

(82) Nilsson N J (1971). Problem Solving Methods in Artificial Intelligence, McGraw-Hill.

(83) Penrose J (1965). The Psychology of Chess, New Society, 29, 967-968.

(84) Philidor A D (1777). Analysis of the Game of Chess, London: Elmsley.

(85) Pitrat J (1968). Realization of a General Game-Playing Program, IFIP 68, H120-124.

(86) Pitrat J (1971). A General Game-Playing Program, Artificial Intelligence and Heuristic Programming, eds Findler N V and Meltzer B, Edinburgh University Press.

(87) Poe E A (c 1860). The Meazel Chess Automaton.

(88) Popplestone R J (1969). An Experiment in Automatic Induction, MI5, 203-218.

(89) Pushkin V A and Shershnev (1972). On Different Modes of Acquiring Information in a Person Solving Discrete Combinatorial Problems, Problems of Heuristics, ed Pushkin V N, Jerusalem: Israel Program for Scientific Translations.

(90) Rayner E H (1958). A Study of Evaluative Problem Solving, Quart J Exp Psych, 10, 155, and 10, 193;
also in: Wason P C et al, Thinking and Reasoning, Penguin.

(91) Remus H (1962). Simulation of a Learning Machine for Playing GO, IFIP62.

(92) Russell R (1964). Kalah; The Game and the Program, Stanford AI Memo No 22.

(93) Ryder J (1971). Go, Thesis, California: Stanford University.

(94) Samuel A L (1967). Some Studies in Machine Learning using the Game of Checkers II - Recent Progress, IBMJ, 11, 601-617;
also in: Annual Review of Automatic Programming, ed Halpern M, Pergamon, 6, 1-36.

(95) Samuel A L (1959). Machine Learning, Tech Rev, 62, 42-45.

(96) Samuel A L (1959). Some Studies in Machine Learning Using the Came of Checker, IBMJ, 3, 210-229;
also in: Computers and Thought, (1963), eds Feigenbaum E A and Feldman J, New York: McGraw-Hill, 71-105.

(97) Samuel A L (1960). Programming Computers to Play Games, Advances in Computers, 1, 165-192.

(98) Scott J J (1969). A Chess Playing Program, MI4, 255-266.

(99) Scott J J (1969). Lancaster vs MACHAC, SIGART, 16, 9-11.

(100) Scurrah M J and Wagner D A (1970). Cognitive Model of Problem Solving in Chess, Science, 169, 290-291. (Fuller version in Cog Psych, 1972).

(101) Selfridge O (1965). Reasoning in Game Playing by Machine, Symposium on Computer Augmentation of Human Reasoning, eds Sass M A and Wilkinson W D, Washington: Spartan.

(102) Shannon C E (1950). Automatic Chess Player, Sci Am, 182, 48-51.

(103) Shannon C E (1950). Programming a Digital Computer for Playing Chess, Phil Mag, 41, 356-375.

(104) Silver R (1967). The Group of Automorphisms of the Game of 3-Dimensional Tic-Tac-Toe, American Math Monthly, 74, 247-254.

(105) Simon H A (1966). Representation in Tic-Tac-Toe, CIP Paper No 90, Carnegie IT.

(106) Simon H A (1967). An Information-Processing Explanation of Some Perceptual Phenomena, Br J Psych, 58, 1-12.

(107) Simon H A and Simon P A (1972). Trial and Error Search in Solving Difficult Problems: Evidence from the Game of Chess, Beh Sc, II, 425-429.

(108) Simon H A and Siklossy L (1972). Representation and Meaning, Prentice-Hall.

(109) Simon H A and Barenfield M (1969). Information-Processing Analysis of Perceptual Processes in Problem Solving, Psych Rev, 76, 473-483.

(110) Slagle J R (1971). Artificial Intelligence, New York: McGraw-Hill.

(111) Slagle J R and Dixon J K (1969). Experiments with Some Programs which Search Game Trees, JACM, 16, 189-207.

(112) Slagle J R and Dixon J K (1970). Experiments with the M x N Tree Searching Program, CACM, 13, 147-154 + 159.

(113) Slate D and Atkin (1970). CDC File Printout from ULCC, London WCL.

(114) Smith R C (1969), The Schach Chess Program, SIGART, 15, 8-12.

(115) Storey S H and Maybrey M A (1969). The Game of Hare and Hounds and the Statistical Study of Literary Vocabulary, MI4, 337-348.

(116) Strachey C S (1952). Logical or Non-Mathematical Programmes, Proc ACM Conf, 46-49.

(117) Tan S T (1972). Representation of Knowledge for Very Simple Pawn Endings in Chess, Thesis, Edinburgh: Department Machine Intelligence.

(118) Thiele T N, Lemke R R and Fu K S (1963). A Digital Computer Card Playing Program, Beh Sci Vol III, 362-268.

(119) Thorp E (1967). Beat the Dealer, TABS.

(120) Thorp E and Walden W (1964). A Partial Analysis of GO, CJ, 7.

(121) Thorp E and Walden W (1970). A Compter-Assisted Study of GO on M x N Boards, TANNPS.

(122) Tikhomirov OK and Poznyanskaya (1966). An Investigation of Visual Search as a Means of Analyzing Heuristics, Soviet Psych, 5, 2-15, translated from: Voprosy Psikhologii, 12, 39-53.

(123) Turing A M (1963). A Digital Computer Applied to Games, Faster than Thought, ed Bowden B V, London: Pitman, 286-310.

(124) Vigneron H (1914). Les Automates, La Natura.

(125) Wasserman A I (1970). Realisation of a Skillful Bridge Bidding Program, FJCC70, 433-444.

(126) Waterman D (1970). Generalisation Learning Techniques for Automating the Learning of Heuristics, AI, 1, 121-170.

(127) Weizenbaum J (1962). How to Make a Computer Appear Intelligent; Five-in-a-Row Offers No Guarantee, Datamation, 24-26.

(128) Weizenbaum J and Shepherdson R C (1962). Gamesmanship, Datamation, 10.

(129) Wiener N (1948). Cybernetics, 1st edition, New York: Wiley.

(130) Williams T G (1965). Some Studies in Game Playing with a Digital Computer, Thesis, Pittsburgh: Carnegie-Mellon University.

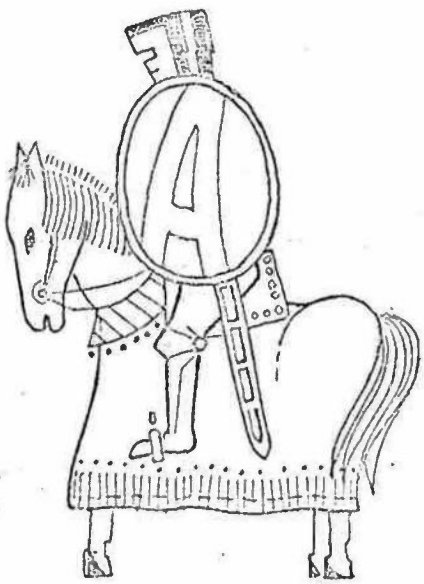
(131) Zobrist A L (1969). A Model of Visual Organisation for the Game of GO, SJCC69, 103-112.

(132) Zobrist A L and Carlson F R (1973). An Advice-Taking Chess Computer, Scientific American, 228, No 6.

(133) Simon H A and Chase W G (1973). Skill in Chess, American Scientist, 61, No 4.

9

LIST OF PARTICIPANTS



BEAL D F	
BOARDMAN R M	
BROWN D	Queen Mary College London University
BURLEY T A	
CAMPBELL G	Brunel University
CAMPBELL J A	King's College London University
CATLOW G W	AERE Harwell
CHESHIRE I M	AERE Harwell
COVINGTON J P	
CROWTHER R D	
CUTTERIDGE Dr O P D	Leicester University
DEANS D	
DIBB A T	AERE Harwell
DORAN J E	Atlas Computer Laboratory
ELDER M	Atlas Computer Laboratory
FLEMING J	
FLETCHER R	AERE Harwell
FRASER L	Southampton University
GOODEY T	Queen Mary College London University
HAILSTONE J E	Atlas Computer Laboratory
HALLOWELL P J	Rutherford Laboratory
HILSUM Miss K A	AERE Harwell
HOWLETT Dr J	Atlas Computer Laboratory
HUMBY E	ICL

JAMES B	Queen Mary College London University
JOHNSON R D	Lanchester Polytechnic
KELLY I D K	
KERMEEN S/LDR R W	RAF College, Cranwell
LARGE R	Glamorgan Polytechnic
LEIGH D J	North Staffs Polytechnic
LIGHTON R J	
LONG D H	Radio and Space Research Station
MACDONALD-ROSS M	The Open University Milton Keynes
MACDONALD-ROSS Mrs M	The Open University Milton Keynes
McEVOY J	
MANNING J R	Shoe and Allied Trades Research Associatio
MOULSDALE R	Birmingham University
POLLARD J M	Plessey Telecommunications
POWELL-EVANS D	
RAPLEY K	Operational Research Branch BEA
READ B J	Portsmouth Polytechnic
REES-JONES G	Operational Research Branch BEA
ROBERTS C L	Atlas Computer Laboratory
RYAN Dr D M	AERE Harwell
SCOTT J J	Queen Mary College London University
SHACKLETON P	
SHEARING Dr G	
SHORE Wing Cdr G B	RAF College, Cranwell
SIMMONS Dr J	Birmingham University
SOPER Dr J M	AERE Harwell
STANIER A M	Essex University
SUNSHINE K W	Portsmouth Polytechnic
WITTEN I H	Essex University
WRIGHT D J	Leicester University

Press Representatives from the 'Oxford Mail'

W PEREIRA
C POSTHLETHWAITE