Philosophical Magazine, Ser.7, Vol. 41, No. 314 - March 1950.

XXII. Programming a Computer for Playing Chess[1]

By CLAUDE E. SHANNON

Bell Telephone Laboratories, Inc., Murray Hill, N.J.[2]
[Received November 8, 1949]

1. INTRODUCTION

This paper is concerned with the problem of constructing a computing routine or "program" for a modern general purpose computer which will enable it to play chess. Although perhaps of no practical importance, the question is of theoretical interest, and it is hoped that a satisfactory solution of this problem will act as a wedge in attacking other problems of a similar nature and of greater significance. Some possibilities in this direction are: -

(1)Machines for designing filters, equalizers, etc.
(2)Machines for designing relay and switching circuits.
(3)Machines which will handle routing of telephone calls based on the individual circumstances rather than by fixed patterns.
(4)Machines for performing symbolic (non-numerical) mathematical operations.
(5)Machines capable of translating from one language to another.
(6)Machines for making strategic decisions in simplified military operations.
(7)Machines capable of orchestrating a melody.
(8)Machines capable of logical deduction.

It is believed that all of these and many other devices of a similar nature are possible developments in the immediate future. The techniques developed for modern electronic and relay type computers make them not only theoretical possibilities, but in several cases worthy of serious consideration from the economic point of view.

Machines of this general type are an extension over the ordinary use of numerical computers in several ways. First, the entities dealt with are not primarily numbers, but rather chess positions, circuits, mathematical expressions, words, etc. Second, the proper procedure involves general principles, something of the nature of judgement, and considerable trial and error, rather than a strict, unalterable computing process. Finally, the solutions of these problems are not merely right or wrong but have a continuous range of "quality" from the best down to the worst. We might be satisfied with a machine that designed good filters even though they were not always the best possible.

---

1 First presented at the National IRE Convention, March 9, 1949, NewYork, U.S.A.
2 Communicated by the Author

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

There is already a considerable literature on the subject of chess-playing machines. During the late 19th century, the Maelzel Chess Automaton, a device invented by Von Kempelen, was exhibited widely as a chess-playing machine. A number of papers appeared at the time, including an analytical essay by Edgar Allan Poe (entitled Maelzel's Chess Player) purporting to explain its operation. Most of the writers concluded, quite correctly, that the Automaton was operated by a concealed human chess-master; the arguments leading to this conclusion, however, were frequently fallacious. Poe assumes, for example, that it is as easy to design a machine which will invariably win as one which wins occasionally, and argues that since the Automaton was not invincible it was therefore operated by a human, a clear 'non sequitur'. For a complete account of the history of the method of operation of the Automaton, the reader is referred to a series of articles by Harkness and Battell in Chess Review, 1947.

A more honest attempt to design a chess-playing machine was made in 1914 by Torres y Quevedo, who constructed a device which played an end game of king and rook against king (Vigneron, 1914). The machine played the side with king and rook and would force checkmate in few moves however its human opponent played. Since an explicit set of rules can be given for making satisfactory moves in such an end game, the problem is relatively simple, but the idea was quite advanced for that period.

The thesis we will develop is that modern general purpose computers can be used to play a tolerably good game of chess by the use of suitable computing routine or "program". White the approach given here is believed fundamentally sound, it will be evident that much further experimental and theoretical work remains to be done.

2. GENERAL CONSIDERATIONS

A chess "position" may be defined to include the following data: -

(1) A statement of the positions of all pieces on the board.
(2) A statement of which side, White or Black, has the move.
(3) A statement as to whether the king and rooks have moved. This is important since by moving a rook, for example, the right to castle of that side is forfeited.
(4) A statement of, say, the last move. This will determine whether a possible en passant capture is legal, since this privilege if forfeited after one move.
(5) A statement of the number of moves made since the last pawn move or capture.

This is important because of the 50 move drawing rule.
For simplicity, we will ignore the rule of draw after three repetitions of a position.

In chess there is no chance element apart from the original choice of which player has the first move. This is in contrast with card games, backgammon, etc. Furthermore, in chess each of the two opponents has "perfect information" at each move as to all previous moves (in contrast with Kriegspiel, for example). These two facts imply (von Neumann and Morgenstern, 1944) that any given position of the chess pieces must be either: -

(1) A won position for White. That is, White can force a win, however Black defends.
(2) A draw position. White can force at least a draw, however Black plays, and likewise Black can force at least a draw, however White plays. If both sides play correctly the game will end in a draw.
(3) A won position for Black. Black can force a win, however White plays.

This is, for practical purposes, of the nature of an existence theorem. No practical method is known for determining to which of the three categories a general position belongs. If there were chess would lose most of its interest as a game. One could determine whether the initial position is a won, drawn, or lost for White and the outcome of a game between opponents knowing the method would be fully determined at the choice of the first move. Supposing the initial position a draw (as suggested by empirical evidence from master games [1]) every game would end in a draw.

It is interesting that a slight change in the rules of chess gives a game for which it is provable that White has at least a draw in the initial position. Suppose the rules the same as those of chess except that a player is not forced to move a piece at his turn to play, but may, if he chooses, "pass". Then we can prove as a theorem that White can at least draw by proper play. For in the initial position either he has a winning move or not. If so, let him make this move. If not, let him pass. Black is not faced with essentially the same position that White has before, because of the mirror symmetry of the initial position [2]. Since White had no winning move before, Black has none now. Hence, Black at best can draw. Therefore, in either case White can at least draw.

In some games there is a simple *evaluation function* f(P) which can be applied to a position P and whose value determines to which category (won, lost, etc.) the position P belongs. In the game of Nim (Hardy and Wright, 1938), for example, this can be determined by writing the number of matches in each pile in binary notation. These numbers are arranged in a column (as though to add them). If the number of ones in each column is even, the position is lost for the player about to move, otherwise won.

If such an evaluation function f(P) can be found for a game is easy to design a machine capable of perfect play. It would never lose or draw a won position and never lose a drawn position and if the opponent ever made a mistake the machine would capitalize on it. This could be done as follows.

Suppose:

f(P) = +1 for a won position,

f(P) = 0 for a drawn position,

f(P) = -1 for a lost position.

At the machine's turn to move it calculates f(P) for the various positions obtained from the present position by each possible move that can be made.
It chooses that move (or one of the set) giving the maximum value to f. In the case of Nim where such a function f(P) is known, a machine has actually been constructed which plays a perfect game [3].

With chess it is possible, *in principle*, to play a perfect game or construct a machine to do so as follows: One considers in a given position all possible moves, then all moves for the opponent, etc., to the end of the game (in each variation). The end must occur, by the rules of the games after a finite number of moves [4] (remembering the 50 move drawing rule). Each of these variations ends in win, loss or draw. By working backward from the end one can determine whether there is a forced win, the position is a draw or is lost. It is easy to show, however, even with the high computing speed available in electronic calculators this computation is impractical. In typical chess positions there will be of the order of 30 legal moves. The number holds fairly constant until the game is nearly finished as shown in fig. 1. This graph was constructed from data given by De Groot, who averaged the number of legal moves in a large number of master games (De Groot, 1946, a). Thus a move for White and then one for Black gives about 10^3 possibilities. A typical game lasts about 40 moves to resignation of one party. This is conservative for our calculation since the machine would calculate out to checkmate, not resignation.
However, even at this figure there will be 10^{120} variations to be calculated from the initial position. A machine operating at the rate of one variation per micro-second would require over 10^{90} years to calculate the first move!

Another (equally impractical) method is to have a "dictionary" of all possible positions of the chess pieces. For each possible position there is an entry giving the correct move (either calculated by the above process or supplied by a chess master.) At the machine's turn to move it merely looks up the position and makes the indicated move. The number of possible positions, of the general order of $64! / 32!(8!)^2(2!)^6$, or roughly 10^{43}, naturally makes such a design unfeasible.

It is clear then that the problem is not that of designing a machine to play perfect chess (which is quite impractical) nor one which merely plays legal chess (which is trivial). We would like to play a skilful game, perhaps comparable to that of a good human player.

A *strategy* for chess may be described as a process for choosing a move in any given position. If the process always chooses the same move in the same position the strategy is known in the theory of games as a "pure" strategy. If the process involves statistical elements and does not always result in the same choice it is a "mixed" strategy. The

following are simple examples of strategies: -

(1)Number the possible legal moves in the position P, according to some standard procedure. Choose the first on the list. This is a pure strategy.
(2)Number the legal moves and choose one at random from the list. This is a mixed strategy.

Both, of course, are extremely poor strategies, making no attempt to select good moves. Our problem is to develop a tolerably good strategy for selecting the move to be made.

3. APPROXIMATE EVALUATING FUNCTIONS

Although in chess there is no known simple and exact evaluating function f(P), and probably never will be because of the arbitrary and complicated nature of the rules of the game, it is still possible to perform an approximate evaluation of a position. Any good chess player must, in fact, be able to perform such a position evaluation. Evaluations are based on the general structure of the position, the number and kind of Black and White pieces, pawn formation, mobility, etc. These evaluations are not perfect, but the stronger the player the better his evaluations.
Most of the maxims and principles of correct play are really assertions about evaluating positions, for example: -

(1)The relative values of queen, rook, bishop, knight and pawn are about 9, 5, 3, 3, 1, respectively. Thus other things being equal (!) if we add the numbers of pieces for the two sides with these coefficients, the side with the largest total has the better position.
(2)Rooks should be placed on open files. This is part of a more general principle that the side with the greater mobility, other things equal, has the better game.
(3)Backward, isolated and doubled pawns are weak.
(4)An exposed king is a weakness (until the end game).

These and similar principles are only generalizations from empirical evidence of numerous games, and only have a kind of statistical validity. Probably any chess principle can be contradicted by particular counter examples. However, form these principles one can construct a crude evaluation function. The following is an example: -

$$f(P) = 200(K-K') + 9(Q-Q') + 5(R-R') + 3(B-B'+N-N') + (P-P') - 0.5(D-D'+S-S'+I-I') + 0.1(M-M') + \ldots$$

in which: -

(1)K,Q,R,B,B,P are the number of White kings, queens, rooks, bishops, knights and pawns on the board.

(2)D,S,I are doubled, backward and isolated White pawns.
(3)M= White mobility (measured, say, as the number of legal moves available to White).

Primed letters are the similar quantities for Black.

The coefficients 0.5 and 0.1 are merely the writer's rough estimate. Furthermore, there are many other terms that should be included [5]. The formula is given only for illustrative purposes. Checkmate has been artificially included here by giving the king the large value 200 (anything greater than the maximum of all other terms would do).

It may be noted that this approximate evaluation f(P) has a more or less continuous range of possible values, while with an exact evaluation there are only three possible values. This is as it should be. In practical play a position may be an "easy win" if a player is, for example, a queen ahead, or a very difficult win with only a pawn advantage.
The unlimited intellect assumed in the theory of games, on the other hand, never make a mistake and a smallest winning advantage is as good as mate in one. A game between two such mental giants, Mr. A and Mr. B, would proceed as follows. They sit down at the chessboard, draw the colours, and then survey the pieces for a moment. Then either: -

(1)Mr. A says, "I resign" or
(2)Mr. B says, "I resign" or
(3)Mr. A says, "I offer a draw," and Mr. B replies, "I accept."

4. STRATEGY BASED ON AN EVALUATION FUNCTION

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

More terms could be added to f(P) to account for exchanges in progress, but it appears that combinations, and forced variations in general, are better accounted for by examination of specific variations. This is, in fact, the way chess players calculate. A certain number of variations are investigated move by move until a more or less quiescent position is reached and at this point something of the nature of an evaluation is applied to the resulting position. The player chooses the variation leading to the highest evaluation for him when the opponent is assumed to be playing to reduce this evaluation.

The process can be described mathematically. We omit at first the fact that f(P) should be only applied in quiescent positions. A strategy of play based on f(P) and operating one move deep is the following. Let M_1, M_2, M_3, ..., M_s be the moves that can be made in

position P and let M_1P, M_2P, etc. denote symbolically the resulting positions when M_1, M_2, etc. are applied to P. Then one chooses the M_m which maximizes $f(M_mP)$.

A deeper strategy would consider the opponent's replies. Let M_{i1}, M_{i2}, ..., M_{is} be the possible answers by Black, if White chooses move M_i. Black should play to minimize f(P). Furthermore, his choice occurs *after* White's move. Thus, if White plays M_i Black may be assumed to play the M_{ij} such that

$$f(M_{ij}\, M_{iP})$$

is a *minimum*. White should play his first move such that f is a maximum after Black chooses his best reply. Therefore, White should play to maximize on M_i the quantity

$$\min_{M_{ij}} \quad f(M_{ij}\, M_{iP})$$

The mathematical process involved is shown for simple case in fig. 2. The point at the left represents the position being considered. It is assumed that there are three possible moves for White, indicated by the solid lines, and if any of these is made there are three possible moves for Black, indicated by the dashed lines. The possible positions after a White and Black move are then the nine points on the right, and the numbers are the evaluations for these positions. Minimizing on the upper three gives +1 which is the resulting value if White chooses the upper variation and Black replies with his best move. Similarly, the second and third moves lead to values of -7 and -6. Maximizing on White's move, we obtain +1 with the upper move as White's best choice.

In a similar way a two-move strategy (based on considering all variations out to 2 moves) is given by

$$\max_{M_i} \quad \min_{M_{ij}} \quad \max_{M_{ijk}} \quad \min_{M_{ijkl}} \quad f(M_{ijkl}\, M_{ijk}\, M_{ij}\, M_{iP}) \qquad (1)$$

The order of maximizing and minimizing this function is important. It derives from the fact that the choices of moves occur in a definite order.

A machine operating on this strategy at the two-move level would first calculate all variations out to two moves (for each side) and the resulting positions. The evaluations f (P) are calculated for each of these positions. Fixing all but the last Black move, this last is varied and the move chosen which minimizes f. This is Black's assumed last move in the variation in question. Another move for White's second move is chosen and the process repeated for Black's second move. This is done for each second White move and the one chosen giving the target final f (after Black's best assumed reply in each case). In his way White's second move in each variation is determined. Continuing in this way the machine works back to the present position and the best first White move. This move in then played. This process generalizes in the obvious way for any number of moves.

A strategy of this sort, in which all variations are considered out to a definite number of moves and the move then determined form a formula such as (1) will be called type A strategy. The type A strategy has certain basic weaknesses, which we will discuss later, but is conceptually simple, and we will first show how a computer can be programmed for such a strategy.

5. PROGRAMMING A GENERAL PURPOSE COMPUTER FOR A TYPE A STRATEGY

We assume a large-scale digital computer, indicated schematically in Fig. 3, with the following properties: -

(1) There is a large internal memory for storing numbers. The memory is divided into a number of boxes each capable of holding, say, a ten-digit number. Each box is assigned a "box number".

(2) There is an arithmetic organ which can perform the elementary operations of addition, multiplication, etc.

(3) The computer operates under the control of a "program". The program consists of a sequence of elementary "orders". A typical order is A 372, 451, 133. This means, extract the contents of box 372 and of box 451, add these numbers, and put the sum in box 133. Another type of order involves a decision, for example C 291, 118, 345. This tells the machine to compare the contents of box 291 and 118. If the first is larger the machine goes on to the next order in the program. If not, it takes its next order from box 345. This type of order enables the machine to choose from alternative procedures, depending on the results of previous calculations. It is assumed that orders are available for transferring numbers, the arithmetic operations, and decisions.

Our problem is to represent chess as numbers and operations on numbers, and to reduce the strategy decided upon to a sequence of computer orders. We will not carry this out in detail but only outline the programs. As a colleague puts it, the final program for a computer must be written in words of one microsyllable.

The rather Procrustean tactics of forcing chess into an arithmetic computer are dictated by economic considerations. Ideally, we would like to design a special computer for chess containing, in place of the arithmetic organ, a "chess organ" specifically designed to perform the simple chess calculations. Although a large improvement in speed of operation would undoubtedly result, the initial cost of computers seems to prohibit such a possibility. It is planned, however, to experiment with a simple strategy on one of the numerical computers now being constructed.

A game of chess can be divided into three phases, the opening, the middle game, and the end game. Different principles of play apply in the different phases. In the opening, which generally lasts for about ten moves, development of the pieces to good positions is the

main objective.
During the middle game tactics and combinations are predominant. This phase lasts until most of the pieces are exchanged, leaving only kings, pawns and perhaps one or two pieces on each side. The end game is mainly concerned with pawn promotion. Exact timing and such possibilities as "Zugzwang", stalemate, etc., become important.

Due to the difference in strategic aims, different programs should be used for the different phases of a game. We will be chiefly concerned with the middle game and will not consider the end game at all. There seems no reason, however, why an end game strategy cannot be designed and programmed equally well.

A square on a chessboard can be occupied in 13 different ways: either it is empty (0) or occupied by one of the six possible kinds of White pieces (P=1, N=2, B=3, R=4, Q=5, K=6) or one of the six possible Black pieces (P=-1, N=-2, ..., K=-6). Thus, the state of a square is specified by giving an integer from -6 to +6. The 64 squares can be numbered according to a co-ordinate system as shown in Fig. 4. The position of all pieces is then given by a sequence of 64 numbers each lying between -6 and +6. A total of 256 bits (binary digits) is sufficient memory in this representation. Although not the most efficient encoding, it is a convenient one for calculation. One further number lambda will be +1 or -1 according as it is White's or Black's move. A few more should be added for data relating to castling privileges (whether the White or Black kings and rooks have moved), and en passant captures (e.g., a statement of the last move). We will neglect these, however. In this notation the starting chess position is given by: -

4, 2, 3, 5, 6, 3, 2, 4;
1, 1, 1, 1, 1, 1, 1, 1;
0, 0, 0, 0, 0, 0, 0, 0;
0, 0, 0, 0, 0, 0, 0, 0;
0, 0, 0, 0, 0, 0, 0, 0;
0, 0, 0, 0, 0, 0, 0, 0;
-1, -1, -1, -1, -1, -1, -1, -1;
-4, -2, -3, -5, -6, -3, -2, -4;
+1 (=lambda)

A move (apart from castling and pawn promotion) can be specified by giving the original and final squares occupied by the moved piece. each of these squares is a choice from 64, thus 6 binary digits each is sufficient, a total of 12 for the move. Thus the initial move e4 would be represented by 1, 4; 3, 4. To represent pawn promotion on a set of three binary digits can be added specifying the pieces that the pawn becomes. Castling is described by the king move (this being the only way the king can move two squares). Thus, a move is represented by (a, b, c) where a and b are squares and c specifies a piece in case of promotion.

The complete program for a type A strategy consists on nine subprograms which we designate T_0, T_1, ..., T_8 and a master program T_9. The basic functions of these programs are

as follows: -

T_0 - Makes move (a, b, c) in position P to obtain the resulting position.
T_1 - Makes a list of the possible moves of a pawn at square (x, y) in position P.
T_2, ..., T_6 - Similarly for other types of pieces: knight, bishop, rook, queen and king.
T_7 - Makes list of all possible moves in a given position.
T_8 - Calculates the evaluating function f(P) for a given position P.
T_9 - Master program; performs maximizing and minimizing calculation to determine proper move.

With a given position P and a move (a, b, c) in the internal memory of the machine it can make the move and obtain the resulting position by the following program T_0: -

(1) The square corresponding to number a in the position is located in the position memory.
(2) The number in this square x is extracted and replaced by 0 (empty).
(3) If x=1, and the first co-ordinate of a is 6 (White pawn being promoted) or if x=-1 and the first co-ordinate of a is 1 (Black pawn being promoted), the number c is placed in square b (replacing whatever was there).
If x=6 and a-b=2 (White castles, king side) 0 is placed in squares 04 and 07 and 6 and 4 in squares 06 and 05, respectively. Similarly for the cases x=6, b-a=2 (White castles, queen side) and x=-6, a-b=+-2 (Black castles, king or queen side).
In all other cases, x is placed in square b.
(4) The sign of lambda is changed.

For each type of piece there is a program for determining its possible moves. As a typical example the bishop program, T_3, is briefly as follows. Let (x, y) be the co-ordinates of the square occupied by the bishop: -

(1) Construct (x+1,y+1) and read the contents u of this square in the position P.
(2) If u=0 (empty) list the move (x, y), (x+1,y+1) and start over with (x+2,y+2) instead of (x+1,y+1).
If lambda*u is positive (own piece in the square) continue to 3.
If lambda*u is negative (opponent's piece in the square) list the move and continue to 3.
If the square does not exist continue to 3.
(3) Construct (x+1,y-1) and perform similar calculation.
(4) Similarly with (x-1,y+1).
(5) Similarly with (x-1,y-1).

By this program a list is constructed of the possible moves of a bishop in a given position P. Similar programs would list the moves of any other piece. There is considerable scope

for opportunism in simplifying these programs; e.g., the queen program, T_5, can be a combination of the bishop and rook program, T_3 and T_4.

Using the piece programs T_1 ... T_6 and a controlling program T_7 the machine can construct a list of all possible moves in any given position P. The controlling program T_7 is briefly as follows (omitting details): -

(1)Start at square 1,1 and extract contents x.
(2)If lambda*x is positive start corresponding piece program T_x and when complete return to (1) adding 1 to square number. If lambda8x is zero or negative, return to 1 to square number.
(3)Test each of the listed moves for legality and discard those which are illegal. This is done by making each of the moves in the position P (by program T_0) and examining whether it leaves the king in check.

With the programs T_0 .... T_7 it is possible for the machine to play legal chess, merely making a randomly chosen legal move at each turn to move. The level of play with such a strategy in unbelievably bad [6].
The writer played a few games against this random strategy and was able to checkmate generally in four or five moves (by fool's mate, etc.). The following game will illustrate the utter purposelessness of random play: -

White (random)	***Black***
1. g3	e5
2. d3	Bc5
3. Bd2	Qf6
4. Nc3	Qxf2 mate

We now return to the strategy based on the evaluation f(P). The program T_8 performs the function of evaluating a position according to the agreed-upon f(P). This can be done by the obvious means of scanning the squares and adding the terms involved. It is not difficult to include terms such as doubled pawns, etc.
The final master program T_9 is needed to select the move according to the maximizing and minimizing process indicated above. On the basis of one move (for each side) T_9 works as follows: -

(1)List the legal moves (by T_7) possible in the present position.
(2)Take the first in the list and make this move by T_0, giving position M_1P.
(3)List the Black moves in M_1P.
(4)Apply the first one giving M_{11} M_1P, and evaluate by T_8.
(5)Apply the second Black move M_{12} and evaluate.
(6)Compare, and reject the move with the smaller evaluation.
(7)Continue with the third Black move and compare with the retained value, etc.

(8) When the Black moves are exhausted, one will be retained together with its evaluation. The process is now repeated with the second White move.
(9) The final evaluation from these two computation are compared and the maximum retained.
(10) This is continued with all White moves until the best is selected (i.e. the one remaining after all are tried). This is the move to be made.

These programs are, of course, highly iterative. For that reason they should not require a great deal of program memory if efficiently worked out.

The internal memory for positions and temporary results of calculations when playing three moves deep can be estimated. Three positions should probably be remembered: the initial position, the next to the last, and the last position (now being evaluated). This requires some 800 bits. Furthermore, there are five lists of moves each requiring about 30x12=360 bits, a total of 1800. Finally, about 200 bits would cover the selections and evaluations up to the present calculation. Thus, some 3000 bits would suffice.

6. IMPROVEMENTS IN THE STRATEGY

Unfortunately a machine operating according to the type A strategy would be both slow and a weak player. It would be slow since even if each position were evaluated in one microsecond (very optimistic) there are about 10^9 evaluations to be made after three moves (for each side). Thus, more than 16 minutes would be required for a move, or 10 hours for its half of a 40-move game.

It would be weak in playing skill because it is only seeing three moves deep and because we have not included any condition about quiescent positions for evaluation. The machine is operating in an extremely inefficient fashion - it computes all variations to exactly three moves and then stops (even though it or the opponent be in check). A good human player examines only a few selected variations and carries these out to a reasonable stopping point. A world champion can construct (at best) combinations say, 15 or 20 moves deep. Some variations given by Alekhine ("My Best Games of Chess 1924-1937") are of this length. Of course, only a few variations are explored to any such depth. In amateur play variations are seldom examined more deeply than six or eight moves, and this only when the moves are of a highly forcing nature (with very limited possible replies). More generally, when there are few threats and forceful moves, most calculations are not deeper than one or two moves, with perhaps half-a-dozen forcing variations explored to three, four or five moves.

On this point a quotation from Reuben Fine (Fine 1942), a leading American master, is interesting: "Very often people have the idea that masters foresee everything or nearly everything; that when they played h3 on the thirteenth move they foresaw that this would be needed to provide a loophole for the king after the combinations twenty moves later, or even that when they play 1. e4 they do it with the idea of preventing Nd5 on Black's twelfth turn, or they feel that everything is mathematically calculated down to the smirk

when the Queen's Rook Pawn queens one move ahead of the opponent's King's Knight's Pawn. All this is, of course, pure fantasy. The best course to follow is to note the major consequences for two moves, but try to work out forced variations as they go."

The amount of selection exercised by chess masters in examining possible variations has been studied experimentally by De Groot (1946, b). He showed various typical positions to chess masters and asked them to decide on the best move, describing aloud their analyses of the positions as they thought them through. In this manner the number and depth of the variations examined could be determined. Fig. 5 shows the result of one such experiment. In this case the chess master examined sixteen variations, ranging in depth from 1/2 (one Black move) to 4-1/2 (five Black and four White) moves. The total number of positions considered was 44.

From these remarks it appears that to improve the speed and strength of play the machine must: -

(1)Examine forceful variations out as far as possible and evaluate only at reasonable positions, where some quasi-stability has been established.
(2)Select the variations to be explored by some process so that the machine does not waste its time in totally pointless variations.

A strategy with these two improvements will be called a type B strategy. It is not difficult to construct programs incorporating these features.
For the first we define a function g(P) of a position which determines whether approximate stability exists (no pieces en prise, etc.). A crude definition might be:

g(P) = 1 if any piece is attacked by a piece of lower value or by more pieces than defences or if any check exists on a square controlled by the opponent
g(P) = 0 otherwise

Using this function, variations could be explored until g(P)=0, always, however, going at least two moves and never more, say, 10.

The second improvement would require a function h(P,M) to decide whether a move M in position P is worth exploring. It is important that this preliminary screening should *not* eliminate moves which merely look bad at first sight, for example, a move which puts a piece en prise; frequently such moves are actually very strong since the piece cannot be safely taken.

"Always give check, it may be mate" is tongue-in-check advice given to beginners aimed at their predilection for useless checks. "Always investigate a check, it may lead to mate" is sound advice for any player.
A check is the most forceful type of move. The opponent's replies are highly limited - he

can never answer by counter attack, for example. This means that a variation starting with a check can be more readily calculated than any other. Similarly captures, attacks on major pieces, threats of mate, etc., limit the opponent's replies and should be calculated whether the move looks good at first sight or not. Hence h(P,M) should be given large values for all forceful moves (check, captures and attacking moves), for developing moves, medium values for defensive moves, and low values for other moves. In exploring a variation h (P,M) would be calculated as the machine computes and would be used to select the variation considered. As it gets further into the variation the requirements on h are set higher so that fewer and fewer subvariations are examined. Thus, it would start considering every first move for itself, only the more forceful replies, etc. By this process its computing efficiency would be greatly improved.

It is believed that an electronic computer incorporating these two improvements in the program would play a fairly strong game, at speeds comparable to human speeds. It may be noted that a machine has several advantages over humans: -

(1)High-speed operations in individual calculations.
(2)Freedom from errors. The only errors will be due to deficiencies of the program while human players are continually guilty of very simple and obvious blunders.
(3)Freedom from laziness. It is all too easy for a human player to make instinctive moves without proper analysis of the position.
(4)Freedom from "never". Human players are prone to blunder due to over-confidence in "won" positions or defeatism and self-recrimination in "lost" positions.

These must be balanced against the flexibility, imagination and inductive and learning capacities of the human mind.

Incidentally, the person who designs the program can calculate the move that the machine will choose in any position, and thus in a sense can play an equally good game. In actual facts, however, the calculation would be impractical because of the time required. On a fair basis of comparison, giving the machine and the designer equal time to decide on a move, the machine might well play a stronger game.

7. VARIATIONS IN PLAY AND IN STYLE

As described so far the machine once designed would always make the same move in the same position. If the opponent made the same moves this would always lead to the same game. It is desirable to avoid this, since if the opponent wins one game he could play the same variation and win continuously, due perhaps to same particular position arising in the variation where the machine chooses a very weak move.
One way to prevent this is to leave a statistical element in the machine. Whenever there are two or more moves which are of nearly equal value according to the machine's calculations it chooses from them at random. In the same position a second time it may then choose another in the set.

The opening is another place where statistical variation can be introduced. It would seem desirable to have a number of the standard openings stored in a slow-sped memory in the machine. Perhaps a few hundred would be satisfactory. For the first few moves (until either the opponent deviates from the "book" or the end of the stored variation is reached) the machine play by memory. This is hardly "cheating" since that is the way chess masters play the opening.

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

Finally we may note that a machine of this type will play "brilliantly" up to its limits. It will readily sacrifice a queen or other pieces in order to gain more material later of to give checkmate provided the completion of the combination occurs within its computing limits.

The chief weakness is that the machine will not learn by mistakes. The only way to improve its play is by improving the program. Some thought has been given to designing a program which is self-improving but, although it appears to be possible, the methods thought of so far do not seem to be very practical. One possibility is to have a higher level program which changes the terms and coefficients involved in the evaluation function depending on the results of games the machine has played. Slam variations might be introduced in these terms and the values selected to give the greatest percentage of "wins".

8. ANOTHER TYPE OF STRATEGY

The strategies described above do not, of course, exhaust the possibilities, In fact, there are undoubtedly others which are far more efficient in the use of the available computing time on the machine. Even with the improvements we have discussed the above strategy gives an impression of relying too much on "brute force" calculations rather than on logical analysis of a position. it plays something like a beginners at chess who has been told some of the principles and is possessed of tremendous energy and accuracy for calculation but has no experience with the game. A chess master, on the other hand, has available knowledge of hundreds or perhaps thousands of standard situations, stock combinations, and common manoeuvres which occur time and again in the game. There are, for example, the typical sacrifices of a knight at f7 or a bishop at h7, the standard mates such as the "Philidor Legacy", manoeuvres based on pins, forks, discoveries, promotion, etc. In a given position he recognizes some similarity to a familiar situation and this directs his mental calculations along the lines with greater probability of success.

There is reason why a program based on such "type position" could not be constructed.

This would require, however, a rather formidable analysis of the game. Although there are various books analysing combination play and the middle game, they are written for human consumption, not for computing machines. It is possible to give a person one or two specific examples of a general situation and have him understand and apply the general principles involved. With a computer an exact and completely explicit characterization of the situation must be given with all limitations, special cases, etc. taken into account. We are inclined to believe, however, that if this were done a much more efficient program would result.

To program such a strategy we might suppose that any position in the machine is accompanied by a rather elaborate analysis of the tactical structure of the position suitably encoded. This analytical data will state that, for example, the Black knight at f6 is pinned by a bishop, that the White rook at e1 cannot leave the back rank because of a threatened mate on c1, that a White knight at a4 has no move, etc.; in short, all the facts to which a chess player would ascribe importance in analysing tactical possibilities. These data would be supplied by a program and would be continually changed and kept up-to-date as the game progressed.
The analytical data would be used to trigger various other programs depending on the particular nature of the position. A pinned piece should be attacked. If a rook must guard the back rank it cannot guard the pawn in front of it, etc. The machine obtains in this manner suggestions of plausible moves to investigate.

It is not being suggested that we should design the strategy in our own image. Rather it should be matched to the capacities and weakness of the computer. The computer is strong in speed and accuracy and weak in analytical abilities and recognition. Hence, it should make more use of brutal calculation than humans, but with possible variations increasing by a factor of 10^3 every move, a little selection goes a long way forward improving blind trial and error.

ACKNOWLEDGEMENTS.

The writer in indebted to E.G. Andrews, L.N. Enequist and H.E. Singleton for a number of suggestions that have been incorporated in the paper.

October 8, 1948.

[1] The world championship match between Capablanca and Alekhine ended with the score Alekhine 6, Capablanca 3, drawn 25.
[2] The fact that the number of moves remaining before a draw is called by the 50-move rule has decreased does not affect the argument.
[3] Condon, Tawney and Derr, U.S. Patent 2,215,544. The "Nimotron" based on this patent was built and exhibited by Westinghouse at the 1938 New York World's Fair.
[4] The longest game is 6350 moves, allowing 50 moves between each pawn move or capture. The longest tournament game on record between masters lasted 168 moves, and the shortest four moves. (Chernev, Curious Chess Facts, The Black Knight Press, 1937.)

[5] See Appendix.
[6] Although there is a finite probability, of the order of 10^{-75}, that random play would win a game from Botvinnik. Bad as random play is, there are even worse strategies which choose moves which actually aid the opponent. For example, White's strategy in the following game: 1. f3 e5 2. g4 Qh4 mate.

APPENDIX. THE EVALUATION FUNCTION FOR CHESS

The evaluation function f(P) should take into account the "long term" advantages and disadvantages of a position, i.e. effects which may be expected to persist over a number of moves longer than individual variations are calculated. Thus the evaluation is mainly concerned with positional or strategic considerations rather than combinatorial or tactical ones. Of course there is no sharp line of division; many features of a position are on the borderline. It appears, however, that the following might properly be included in f(P): -

(1) Material advantage (difference in total material).
(2) Pawn formation:
  (a) Backward, isolated and doubled pawns.
  (b) Relative control of centre (pawns at e4, d4, c4).
  (c) Weakness of pawns near king (e.g. advanced g pawn).
  (d) Pawns on opposite colour squares from bishop.
  (e) Passed pawns.
(3) Positions of pieces:
  (a) Advanced knights (at e5, d5, c5, f5, e6, d6, c6, f6), especially if protected by pawn and free from pawn attack.
  (b) Rook on open file, or semi-open file.
  (c) Rook on seventh rank.
  (d) Doubled rooks.
(4) Commitments, attacks and options:
  (a) Pieces which are required for guarding functions and, therefore, committed and with limited mobility.
  (b) Attacks on pieces which give one player an option of exchanging.
  (c) Attacks on squares adjacent to king.
  (d) Pins. We mean here immobilizing pins where the pinned piece is of value not greater than the pinning piece; for example, a knight pinned by a bishop.
(5) Mobility.

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

REFERENCES.

Chernev, 1937, *Curious Chess Facts*, The Black Knight Press.

De Groot, A.D., 1946a, *Het Denken van den Schaker* 17-18, Amsterdam; 1946b, Ibid., Amsterdam, 207.

Fine, R., 1942, *Chess the easy Way*, 79, David McKay.

Hardy and Wright, 1938, *The Theory of Numbers*, 116, Oxford.

Von Neumann and Morgenstern, 1944, *Theory of Games*, 125, Princeton.

Vigneron, H., 1914, *Les Automates*, La Natura.

Wiener, N., 1948, *Cybernetics*, John Wiley.