CHAPTER 8

GAME-PLAYING AND GAME-LEARNING AUTOMATA

D. MICHIE

1. INTRODUCTION

Mechanized game-playing is studied partly for fun and partly for its value as a model for decision-taking in real life, for example in control problems of chemical engineering, or in medical diagnosis, or in animal behaviour. In the last case the two players are the animal, whose moves are called "responses", and its environment, whose moves are called "stimuli" (see Michie, 1962).

A game is a sequence of choices, each choice being made from a number of discrete alternatives. Every sequence terminates in an outcome, and every outcome has a value. It is conventional to assign outcome values from the point of view of the opening player, so that in Chess, for example, there are three outcome values, which we may denote $+1$, 0, and -1 (a win for White, a draw, and a win for Black respectively). Chess belongs to the class of two-person games with perfect information (i.e. both sides are allowed full view of the board) without chance moves. Other examples are Draughts (Checkers), Noughts and Crosses (Tic-tac-toe), Go and Nim. The last two are two-valued games, since drawn positions do not occur. Many-valued games also exist.

2. A GAME AS A GRAPH

A game of the type which we have defined can be represented by a directed graph, as has been done in Fig. 1 for a simple version of Nim. Terminal nodes have been labelled with the corresponding outcome values, on the presumption that a description of the position itself (including which player's turn it is to play) is all we need to know in order to assign a value to it. Although this is true of Nim it is not necessarily the case in general. The value of a position, even a terminal position, could be dependent on the route followed in arriving at it. If there were a rule in Chess that making more than 100 moves automatically loses the game, it would be fruitless for a player to point out, after making his hundredth move, that he could force mate in one.

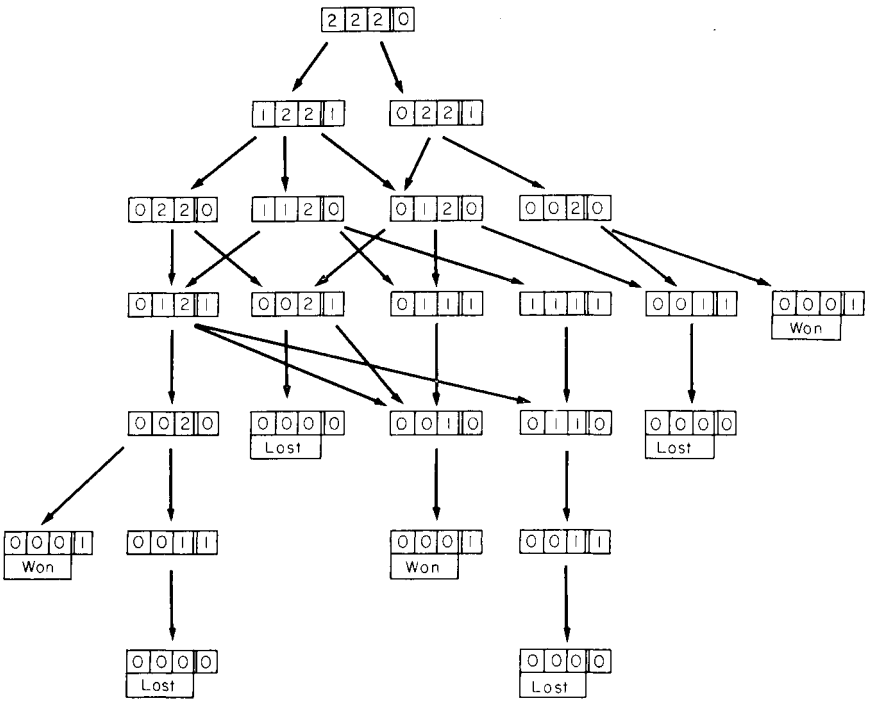

FIG. 1. Representation of an ultra-simple version of the game of Nim by means of a directed graph. The initial state here consists of three heaps of two coins each. The fourth digit in the numerical coding of positions is 0 or 1 according to whether it is the first or second player's turn to play. Moves are made alternately, and consist in the removal of one or more coins from any one heap. The object is to be the player who removes the last coin.

In order to handle this complication it is customary to depict a game not by a generalized graph, in which the nodes stand for positions, but by the special kind of graph known as a *tree*. The nodes are now made to stand not for positions but for *partial plays*. A partial play embodies the total past history from the start up to the given stage and hence is not simply a static photograph of the current position. A graph of positions can be converted in a straightforward manner into a tree of partial plays, as has been done in Fig. 2 for the Nim example.

3. THE 'FOREGONE CONCLUSION' THEOREM

The tree representation permits the demonstration of the fundamental theorem that every game is, in a certain easily definable sense, a foregone

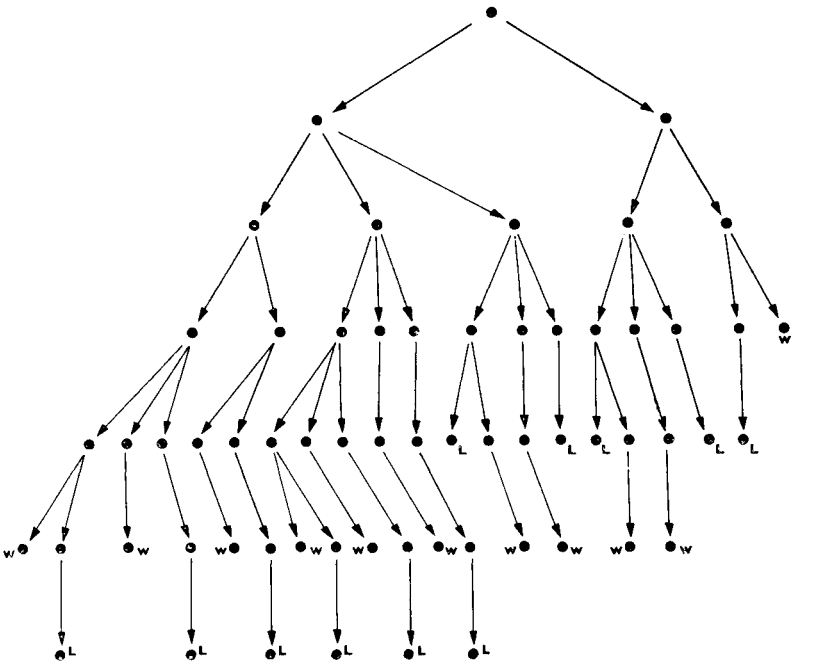

FIG. 2. The graph of the game shown in Fig. 1 expanded into a tree of partial plays. The terminal nodes have been labelled with outcome values: W for a won game (for the first player) and L for a lost game.

conclusion. The theorem applies just as powerfully to "deep" games such as Chess or Go as it does to Noughts and Crosses, where the truth of the theorem is fairly obvious to most adults. It states that, if we assume that both sides follow an optimal strategy, values can be assigned not only to the terminal nodes—these are already assigned by the rules of the game—but to every other node, including the starting node. This initial value is the *value of the game*, and shows what the outcome would be, provided that neither side were to make a mistake.

The method of "backing up" the terminal values to higher and higher levels of the tree is important to grasp, since it foreshadows the "minimaxing" procedure employed in mechanized systems. The terminal spots of the complete tree of the game can be labelled with the values of the corresponding outcomes (win, draw, lose). The pre-terminal nodes can then be assigned values by the following rule: if the node represents our side's choice, then label it with the maximum of the values of the family of terminal nodes dependent from it. If it represents opponent's choice, then label it with the minimum. The assumption is, of course, that our side will play so as to

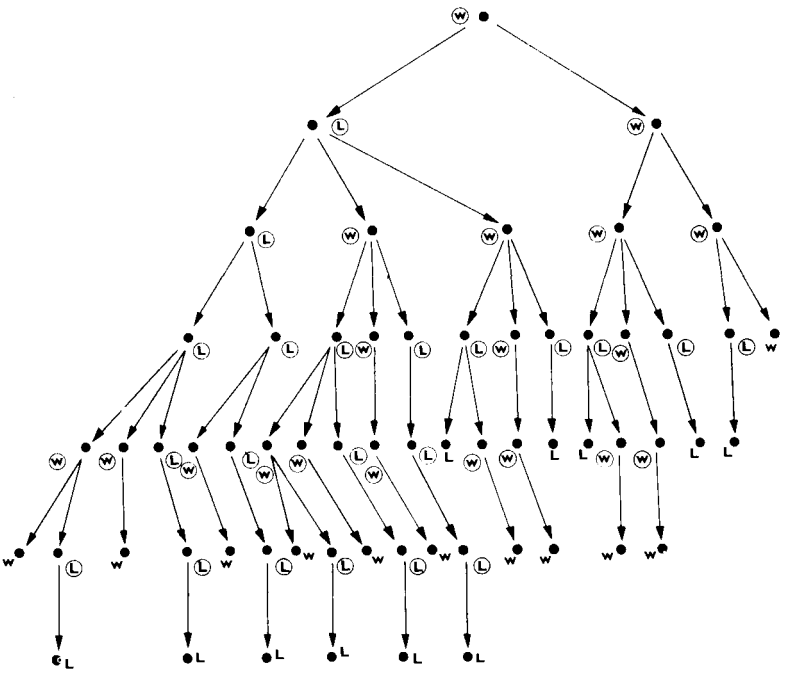

FIG. 3. The tree of partial plays shown in Fig. 2, with all nodes labelled back to the origin by application of the minimax rule. Values obtained by the "backing up" process are ringed. The value of the game is thus established as a win for the first player.

maximize the value of the outcome and conversely for the opponent. The process can obviously be repeated, thus "backing-up" the assigned values to higher and higher values of the tree. Ultimately the initial node will be labelled. This represents the *value* of the game. The process is illustrated in Fig. 3.

4. TWO FALLACIES

It is possible to be misled by the foregone conclusion theorem into thinking that the mechanization of game-playing presents no particular problem. But it is one thing to assert a general theorem about games of this structure and quite another thing to be able to say what the value of any particular game is. Nobody knows whether Chess is a forced win for White, or for Black, or is a draw, nor is anyone ever likely to know: the number of partial plays is certainly greater than the number of elementary particles in the observable universe, so that the task of constructing the required tree is

unthinkable. In the case of a relatively trivial game such as Noughts and Crosses, or the 3 + 3 + 3 version of Nim, total enumeration is feasible, so that an optimal strategy could take the form of an exhaustive list of positions, with a recommended move entered against each.

This is a convenient point at which to consider another misconception of a more subtle kind. We have seen that it is only practicable exhaustively to specify an optimal strategy for a game of relatively trivial dimensions. "Optimal" is here used in a technical sense, as describing any strategy which invariably selects the highest-valued alternative, in the case of the opening player, or the lowest-valued in the case of the second player. Such a strategy guarantees that the outcome cannot be worse than the value of the game. Is it optimal in a different and more practical sense, in ensuring that the expected outcome is as favourable as possible?

The answer is No, for a reason familiar to game-players in connexion with "complicating the position" and the setting of "traps". Against a weaker player it can sometimes pay to make an inferior move with the idea of tempting, bluffing, or bewildering him into making mistakes. More generally, the object of a game is to secure the best result possible against the actual opponent confronting us across the board, rather than the hypothetical opponent defined by the minimax rule. It will be essential to keep this distinction in mind when we later discuss the learning, as opposed to the mere implementation, of effective strategies.

5. COMPRESSED STRATEGIES

The idea of an exhaustive enumeration is useful as a base-line for attempts at compressing the information into a manageable compass. The game of Nim affords a famous example of an elegant compression, namely the rule of play based on conserving the parity of a binary representation of the current position. Compression of a strategy is analogous to the mental processes known as abstraction and generalization, and it is at this point that the challenge to the mechanizer becomes interesting. A compressed strategy is typically based on the evaluation of intermediate positions by means of abstracted features, a process first explicitly propounded in the present context by A. M. Turing, who was responsible for the first detailed thinking about the design of game-playing machines (see Turing 1953).

6. TURING'S IDEAS

It is sometimes thought that Turing's interest in mechanized game playing was the spare time frivolity of a man who reserved his serious thoughts for

worthier topics. This was not the case. He had the conviction that the development of high-speed digital computing equipment would make possible the mechanization of human thought processes of every kind, and that games constituted an ideal model system in which studies of machine intelligence could first be developed. This view is commonplace today although at the time it was regarded askance by many. I shall not stay longer in the pre-history of the subject but will list the basic ideas which Turing contributed to mechanized game-playing.

(1) *Minimax evaluation.* This is an extension of the method referred to in our earlier description of the use of the "tree" representation to assign a value to every choice-point and so establish the 'foregone conclusion' theorem.

Turing pointed out that the method of "backing-up" could be applied to sub-terminal evaluations made on some general strategic grounds without

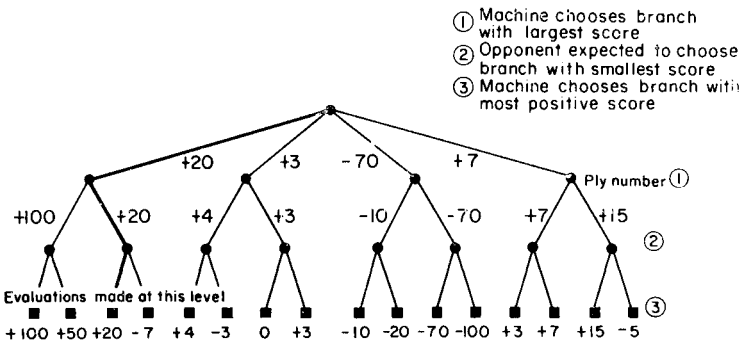

FIG. 4. Simplified diagram showing how the evaluations are backed up through the "tree" of possible moves to arrive at the best next move. The evaluation process starts at (3). (Reproduced from Samuel, 1960, p. 167)

necessitating complete backward tracing from an astronomical number of terminal spots. The crudest of such evaluations would be those provided by the value of own and enemy pieces, rating, for example, in the case of chess, Q = 10, R = 5, B = $3\frac{1}{2}$, Kt = 3, P = 1. Other components of a strategic score might be numerical expressions of such features as mobility, control of central squares, advancement of pawns, etc. Given some means of attaching a score to any position we have a simple scheme for a game-playing machine: look forward along the tree to a certain depth, evaluate all the positions at this depth, and then "back-up" the evaluation by the minimaxing method described earlier. The process is illustrated in Fig. 4.

(2) "*Dead positions*" *and variable lookahead.* Turing defined a "dead position" as one which not only lay beyond the given depth of lookahead,

but could not be reached from within this depth by a capture-recapture sequence. The point of the "dead position" category was to introduce, although in a crude form, the idea of a variable depth of lookahead. The notion was to analyse ahead to a fixed depth, for example 2, and then follow all possible capture-recapture sequences until "dead positions" were reached, before finally making evaluations. Subtler variations on this theme have been subsequently embodied into the program of A. L. Samuel (1959) for Draughts (Checkers). The quality of play which resulted from Turing's application of ideas (1) and (2) was poor, being in part limited by the lack of adequate high-speed computing equipment. The earliest published description of an automaton designed according to these principles seems to be that of Turing (1953), where a complete record of a specimen game is given. This was not, however, Turing's earliest essay in the field. As early as 1944 he was discussing with various friends and colleagues the idea of making computers play chess. In 1948, as a *jeu d'esprit*, he and D. G. Champernowne devised a one-move analyser, the TUROCHAMP. In the same spirit, S. Wylie and D. Michie designed a rival, the MACHIAVELLI.

A tournament between the two machines was attempted at that time, but never completed. Many years later, the MACHIAVELLI was tested in combat with Smith's One-Move Analyser, (the SOMA) designed by J. Maynard Smith (see Maynard Smith and Michie 1961 for the full game record).

In order to give the reader a picture of the kind of board evaluations used in these early automata, the specifications of a SOMA-MACHIAVELLI hybrid are reproduced in the Appendix, contributed by J. Maynard Smith, at the end of this chapter. This machine, when allowed a lookahead of 2, has a standard of play equal to that of a mediocre human player (Maynard Smith and Michie 1961). The design of the TUROCHAMP is no longer extant.

Turing attempted to program both the TUROCHAMP and the MACHIAVELLI on the Manchester Ferranti Mark I in order to play them off against each other automatically, but he never completed it. Shannon (1950) made proposals, not worked out in detail, for the design of a chess-playing program, and this stands as the earliest published treatment of the topic.

It would, however, be unsatisfactory to leave this discussion of chess without exhibiting some specimen of reasonably competent machine play. Unfortunately this is not possible, since there seem to be no survivors of the various chess programs known to have been written (see Newell, Shaw and Simon (1958) and Samuel (1960)) which are capable of really competent chess. On this criterion it appears that no significant advance has been made since the early days, in spite of a widespread and strongly-held belief to the contrary.

(3) *Self-improving behaviour obtained through parameter-adjustment.* We are today familiar in many and varied contexts with the idea of a program which improves its performance in the light of the results obtained, by varying the coefficients or weights attached to the various terms of some linear evaluation function. In bare outline Turing proposed this system also, but its full development had to await Samuel's work on the mechanization of Draughts (Checkers), an outline account of which will now be given.

7. SAMUEL'S WORK WITH THE GAME OF DRAUGHTS (CHECKERS)

During the early 1950's the game of Draughts was programmed by C. Strachey—a considerable *tour de force* at that time—but the work was not taken much beyond the solution of the purely programming problems involved.

It was, however, the first working computer program to combine position-evaluation with a minimaxing lookahead. Evaluation was crude, consisting only of a count of own and enemy pieces. But the lookahead went to a minimum depth (or "ply" in A. L. Samuel's terminology) of 6 , and contained an important means of extending this range. In the forward analysis captures were not counted as moves, so that forcible variations involving exchanges permitted an extended lookahead. This is reminiscent of Turing's suggestion that lookahead might be continued until a "dead position", i.e. one not offering any capture opportunities, was reached. Both schemes aim at directing as high a proportion of the search as possible to the less luxuriantly ramified parts of the tree. This is precisely what expert human players learn to do, exploring only forcible variations to any great depth and otherwise relying largely on their powers of static evaluation. The difference between the Chess or Checkers master and the strong club player does not lie, as people often suppose, in the ability to explore a forward tree of many thousands of nodes and keep track of them somehow in the short-term memory, but rather in a superior instinct of how to restrict the amount of searching that has to be done. The story is told of the chess master who, when asked how many moves ahead he looked, replied "One, the right one"!

Building upon Strachey's foundation, A. L. Samuel (1959) has subsequently engaged on the most thorough and illuminating exercise which is on record. In 1959 Samuel's program was defeating its author with fair regularity. It has subsequently improved its standard with practice, and in 1962 it defeated a former checkers champion of Connecticut, R. W. Nealey. Five further games have since been played, of which four were drawn and one won by Mr. Nealey.

The record of the first game is reproduced below. The annotations were made by Dr. Samuel. Mr. Nealey's comments, as quoted by the *IBM Research News*, are as follows:

> Our game . . . did have its points. Up to the 31st move, all of our play had been previously published except where I evaded "the book" several times in a vain effort to throw the computer's timing off. At the 30–26 loser and onwards, all the play is original with us, so far as I have been able to find. It is very interesting to me to note that the computer had to make several star moves in order to get the win, and that I had several opportunities to draw otherwise. That is why I kept the game going. The machine, therefore, played a perfect ending without one misstep. In the matter of the end game, I have not had such competition from any human being since 1954, when I lost my last game.
>
> Nealey (WHITE) vs. Samuel Checker Program (BLACK), July 12, 1962, at Yorktown, New York. Mr. Nealey was given the option and chose to defend. The Old Fourteenth opening was followed.

11	15		
23	19		
8	11		
22	17		
4	8		
17	13		25–22 would restrict Black's variety of play a little more.
15	18		
24	20		Lee's Old Fourteenth, Var. 9. 11–15 is the trunk move.
9	14		
26	23		Doran's Var. 100 listed as an even game.
10	15		
19	10		
6	15		
28	24		Doran lists 23–19 as giving an easier game for White.
15	19		An aggressive move for Black.
24	15		
5	9		
13	6		
1	19	26	
31	22	15	
11	18		Still in Lee's Var. 9 and Doran's Var. 100
30	26		This is probably a poor move on Mr. Nealey's part.
8	11		A good reply maintaining control of the center.
25	22		
18	25		
29	22		
11	15		
27	23		
15	19		
23	16		

Let us now consider the new features added by Samuel to Turing's original framework of ideas. Of these, the first is what Samuel calls *rote-learning*. At its simplest level this consists in no more than saving the score for future reference whenever a position is evaluated. If the same position is encountered in subsequent play, time which would otherwise be spent on

re-evaluating the position is saved, and thus is available for alternative computations such as extending the lookahead distance.

Rote-learning, however, as so defined, leads immediately to a most important means of exploiting past evaluations. The point is at the same time subtle and simple. Many of the evaluations which are saved have themselves been obtained by "backing-up" from spots at a lower level in the game tree, as was described earlier. In a subsequent play of the game such a position might again be encountered, this time in the lookahead from the current position on the board. The effect of having saved its value is then that the *effective* lookahead has been extended. The point is illustrated in Fig. 5.

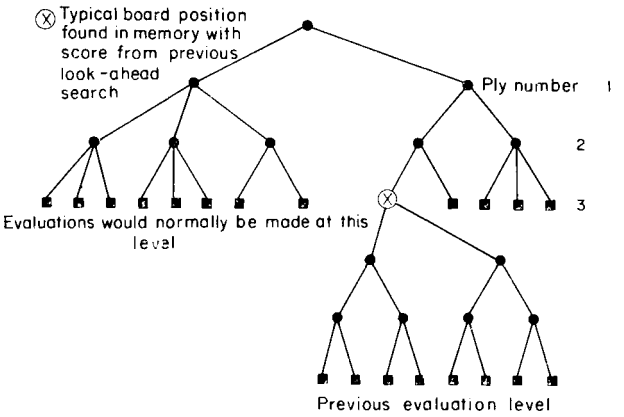

FIG. 5. Simplified representation of the rote-learning process, in which information saved from a previous game is used to increase the effective ply of the backed-up score. (Reproduced from Samuel, 1960, p. 183.)

A further important feature of rote-learning as implemented by Samuel causes the program to tend to delaying tactics when losing, and the opposite when winning. I shall not here go into the details of this, nor of many other ingenious devices which Samuel employed in creating the full operating system, but will proceed directly to the learning-by-generalization system which Samuel ultimately combined with the rote-learning routine. As mentioned before, this was based on a linear scoring function consisting of a number of terms each intended to express some strategic feature of the game, together with weighting coefficients. During the learning procedure the program caused two phantom players, Alpha and Beta, to play against each other, and adjustments to be made in the weighting coefficients according to the relative success of the two sides. Fig. 6 shows the early phase of a learning process of this kind. Terms whose coefficients gravitated to zero or near-zero values were dropped from the list and replaced by new terms selected from a

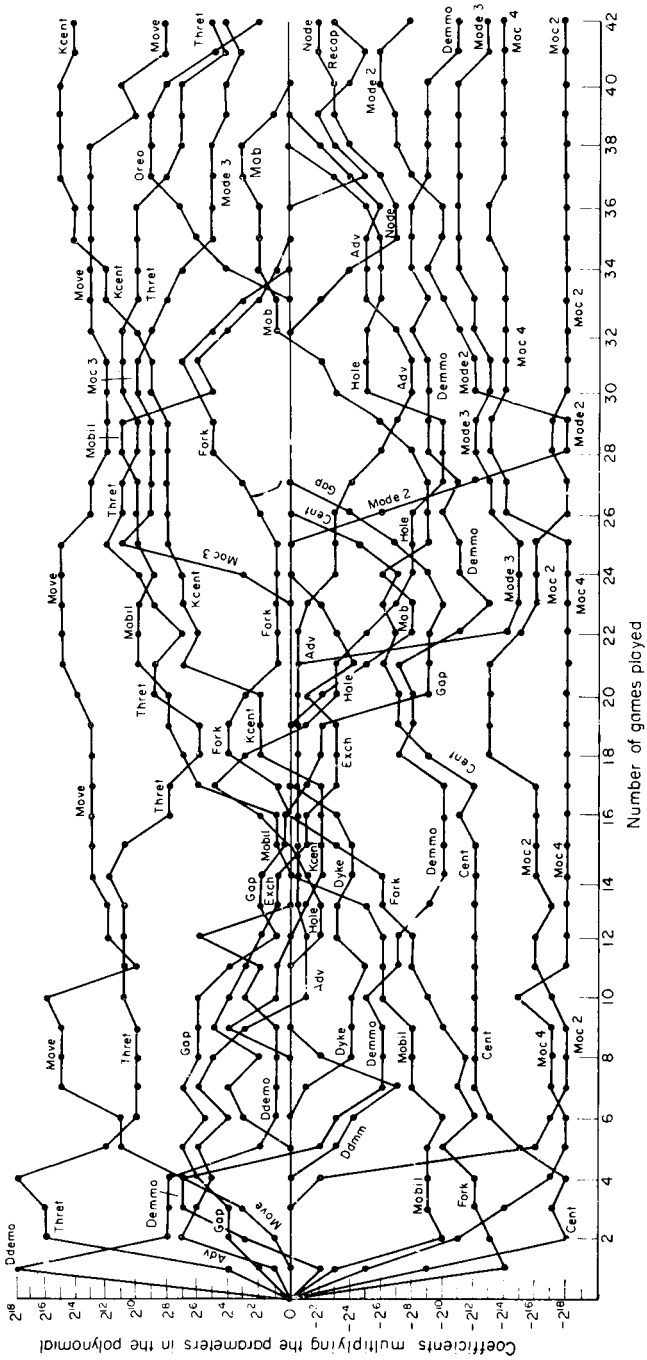

FIG. 6. Second series of learning-by-generalization tests. Coefficients assigned by the program to the more significant parameters of the evaluation polynomial plotted as a function of the number of games played. Two regions of special interest migh: be noted: (1) the situation after 13 or 14 games, when the program found that the initial signs of many of the terms had been set incorrectly, and (2) the conditions of relative stability which are beginning to show up after 31 or 33 games. (Reproduced from Samuel, 1959.)

reserve list. Each term stands for some strategic feature of the game. Thus ADV = "advancement", KCENT = "king centre control" etc. The method used to modify the coefficients attached to the various terms was based on the idea of bringing about as good agreement as possible between the values calculated for different positions connected by the same minimax chain. For details of this and other features of the program, the reader is referred to Samuel's published descriptions.

The fact that Samuel's list of terms was man-made and that no means was devised whereby the program could generate new terms for itself is significant, for it is precisely at this point that current research on "artificial intelligence" is encountering great difficulty. Deductive processes are in principle easy to mechanise. But the intellectual processes involved in induction, with their aura of "creativity", "originality", "concept-formation", etc., are difficult to capture within a formal framework.

It would be unprofitable to pursue these questions here. Rather, we shall now turn from games of such complexity that schemes of classification and evaluation are positively forced upon the would-be mechaniser, and consider the following question. In simple games for which individual storage of all past board positions is feasible, is any optimal learning algorithm known?

8. THE "TWO-ARMED BANDIT" PROBLEM

The surprising answer to the last question is "No". In other words, it is beyond our present understanding to design an automaton guaranteed to make the most efficient possible job of learning even games more trivial than Noughts and Crosses. The difficulty lies in costing the acquisition of information for future use at the expense of present expected gain. A means of expressing the value of the former in terms of the latter would lead directly to the required algorithm. We can briefly illustrate the point by scaling the problem down to the simplest task definable as a two-person game. Such a game will have two outcomes (win, lose), two stages (one choice for each player) and two alternative moves at each choice-point. The structure of the game is shown in Fig. 7. The learning problem is that of the opening player, whose task is to gain the greatest possible number of wins in N successive plays of the game on the basis of no prior knowledge of his opponent's behaviour.

In the special case where the opponent responds to M_1, with losing and winning moves in the proportions p_1, $(1 - p_1)$ and to M_2 with losing and winning moves in the proportions p_2, $(1 - p_2)$, where p_1 and p_2 are stochastic variables of fixed, but initially unknown, values, the problem is identical with a well-known problem as the "two-armed bandit" problem. Here, in place of the opponent with his two modes of response we have two gambling

machines characterized by the stochastic variables p_1 and p_2 representing mutually independent probabilities of pay-off. Given some assumption about the prior probability distributions of p_1 and p_2 (e.g. that the prior distributions are uniform), the problem is to devise a policy for choosing between the machines on successive trials in such a way as to maximize the expected monetary gain over N trials, or alternatively over an infinite succession of trials during which the value of the pay-off is suffering a steady exponential decay. At first sight the problem looks easy, since there is no particular

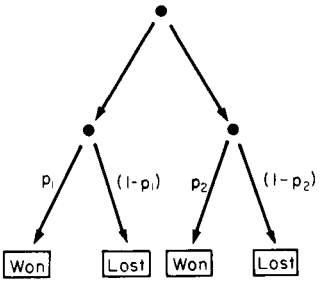

FIG. 7. The formal structure of the simplest possible two-person game, in which each player is allowed one move, each move being a choice of two alternatives. The second player, if he were a non-imbecilic human, would infallibly win so trivial a game. Imagine, however, a fallible player, whose replies to the two possible openings are made with probabilities p_1, p_2, initially unknown to the first player. Then the first player's task of learning to play the game is equivalent to the two-armed bandit problem (see text).

difficulty of forming *estimates* of p_1 and p_2 from the results of past trials, which we may denote by $\hat{p}_1$ and $\hat{p}_2$. It now seems common sense to choose M_1 or M_2 according as $\hat{p}_1 > \hat{p}_2$ or $\hat{p}_1 < \hat{p}_2$. On reflection this can be seen to be the right answer to the wrong question, as though we had been asked for a policy which would always maximize the expected gain on the *next* trial, instead of the expected gain accumulated over all remaining trials. It can be seen, even on intuitive grounds, that in some such case as

M_1: 2 trials, 1 success,

M_2: 99 trials, 50 successes,

M_1 is the better choice for the next trial rather than M_2 because of the much greater information increment, from which cash benefit can be extracted in future choices.

The two-armed bandit problem is still unsolved, and *a fortiori* so is the problem of optimizing the design of a game-learning automaton for even the simplest of games.

9. CONCLUDING REMARKS

Given that the key problems of mechanizing the learning both of easy and of difficult games seem still a long way from solution, one may ask whether the further experimental pursuit of the subject is truly worth the effort, when so many more pressing applications of computer science claim attention. Any answer to this question must be sought in the special challenge offered by games to develop and perfect new technique, and to isolate in pure form the logical structure underlying real-life systems. In view of the prevalence of multi-stage decision processes in practical contexts, the use of games as testing-grounds for automatic methods seems assured of continuance.

10. REFERENCES

MAYNARD SMITH, J. and MICHIE, D. (1961) Machines that play games, *New Scientist*, **12**, 367–9.

MICHIE, D. (1962) Puzzle-learning versus game-learning in studies of behaviour, pp. 90–100 of *The Scientist Speculates* (I. J. Good, ed.) Heinemann, London.

NEWELL, A., SHAW, J. C. and SIMON, H. (1958) Chess-playing programs and the problem of complexity, *IBM Journal of Research and Development*, **2**, 320–35.

SAMUEL, A. L. (1959) Some studies in machine learning using the game of checkers, *IBM Journal of Research and Development*, **3**, 211–29.

SAMUEL, A. L. (1960) Programming computers to play games, pp. 165–92 of *Advances in Computers*, vol. 1 (F. L. Alt, ed.) Academic Press, London and New York.

SHANNON (1950) Programming a digital computer for playing chess, *Phil. Mag.* **41**, 356–75.

TURING, A. M. (1953), pp. 288–95 of *Faster than Thought* (B. V. Bowden, ed.), Pitman, London.

APPENDIX

JOHN MAYNARD SMITH

Rules of SOMAC*

(SOMA with features taken from the MACHIAVELLI)

1. GENERAL (A)

It is supposed that SOMAC is playing the White pieces. If there are n legal moves, calculate the values $V_1, V_2, \ldots, V_n$ of White's position after each of them, and make the move giving the maximum value. If two or more moves give equal values, choose at random. In practice it is easier to calculate $V_1 - V_0$, etc., where V_0 is value of White's position before moving.

* Numbered notes (1) (2), etc., explain how moves are calculated. Lettered notes (A), (B), etc., comment on effects of rules, and are at the end.

The value of White's position is equal to:

	Note
Value of White's pieces*—Value of Black's pieces	(1)
+Value of squares attacked by White pieces	(2)
—"Exchange Value" of White pieces	(3)
+"Exchange Value" of Black pieces	(3)
−5 for every White piece (*not* pawns) which, although not attacked by a Black pawn, can be so attacked by a single Black move (i.e. by an advance or capture by a black pawn) (B)	(4)
$+5/x$ if Black is in check, where x is the number of legal moves available to Black (C)	
+5 if O — O and/or O — O — O is a legal move; +16 if O — O has been played; +14 if O — O — O has been played (D)	(5)

Note (1) *Value of pieces*

$$P = 10;\ N = B = 30;\ R = 50;\ Q = 90 \text{ or } 100;\ K = 1000$$

The value of a Queen is 90 if it is not on the back two ranks and the player has not castled; otherwise Q = 100. Thus if White, before castling, moves Q from the back two ranks, this reduces the value of his position by 10: if he then castles this increases the value of his position by 10. (E)

Note (2) *Value of squares attacked*

+1 for every square, occupied or not, attacked by a White piece (2i)
+1 extra for every attack on one of the four central squares
+2 extra for every attack on a square adjacent to the Black King (2ii)

Note that the squares attacked by each White piece are enumerated separately and added, so that if a square is attacked by r White Pieces, this adds r to the value of his position.

Note (2i)

A piece attacks a square (occupied by a Black or White piece or unoccupied) if it could capture a piece of opposite colour on that square: but this rule is modified by "*pins*" and "*transparencies*" as follows: (F).

Pins. A White piece "X" does not attack a square if, should it move to that square, *either*, a Black piece attacks a White piece (other than X) of Value higher than itself, which it did not attack before "X" was moved, *or* a Black piece attacks a White piece (*not* a pawn; *not* X) which it did not attack before X was moved, and which occupies a square not itself attacked by any other White piece.

* Unless otherwise stated "Pieces" includes pawns and King.

Transparencies. In calculating squares "attacked", certain pieces are assumed, contrary to the rules of the game, to be allowed to move through squares occupied by other pieces of the same colour, which are then said to be transparent, as follows:

R is transparent to R,
R is transparent to Q on files and ranks only,
B is transparent to Q on diagonals only,
Q is transparent to R and B,
P is transparent to B and Q, along diagonals, but only for one square beyond pawn, and only if B or Q are moving forwards along the diagonal.

Note (2ii)
A square is "adjacent" to the King if it is attacked by the King.

Note (3) *Calculation of exchange values* (*G*)

(In what follows, a piece is said to be attacked by another piece if it occupies a square attacked by that piece: squares attacked by Black pieces are calculated, *mutatis mutandis*, in the same way as squares attacked by White pieces.)
Step A.
Calculate "Swap-off Value" S for all pieces, of either colour which are *en prise* i.e. attacked by a piece of opposite colour.

Swap-off value of a White piece (*H*)

(reverse this procedure for a Black piece)

A White piece of Value v_0 is attacked by n White pieces of value v_1, $v_2, \ldots, v_n$ in ascending order of value and by N Black pieces of values u_1, $u_2, \ldots, u_N$ in ascending order of value (3i).
Calculate:

$$w_1 = v_0 \qquad b_1 = v_0 - u_1$$

$$w_2 = v_0 - u_1 + v_1 \qquad b_2 = v_0 - u_1 + v_1 - u_2$$

and so on, until the series $w_1, b_1, w_2, b_2, w_3 \ldots$ terminates (it cannot terminate before v_0).

(i) if $n \geqslant N$ (series ends at b_N),
S = largest value of b, or, if smaller, the smallest value of w preceding this.

(ii) if $n < N$ (series ends at w_{n+1}),
S = smallest value of w, or, if larger, the largest value of b preceding this.

Note (3i). If a piece X only attacks the square by virtue of the assumed transparency of another piece Y, then even if $v_x < v_y$ (i.e. Value of X < Value of Y) v_x cannot precede v_y in this series.

Step B

Calculation of exchange value of White pieces = V_{EW} (*I*)

Ignore negative and zero values of S.

V_{EW} = Largest positive value of S on any White piece +5 for each additional White piece with a positive S. (I)

Step C

Calculation of exchange value of Black pieces = V_{EB} (*J*)

If only one Black piece has a positive value of S, $V_{EB} = +5$.

If r Black pieces have positive values of S, $V_{EB} = 5(r - 1)$ + Second highest Value of S. (J)

In calculating r, any White piece which is *en prise* with a zero or positive value of S is deemed to attack only one Black piece, which is taken to be that piece which maximises r.

If $r > 1$, but would have been 1 or 0 had it not been that a White piece with a negative S value $= -S'$ attacked two or more Black pieces, then V_{EB} is taken as *either* the value as calculated above, *or* as S', whichever is less.

Note (4). Only one value of −5 is scored for a single White piece, even if it could be attacked by two or more Black pawn moves.

Note (5). If a player has castled, castling is no longer a legal move: so the net gains for O − O and O − O − O are +11 and +9 respectively.

Comments on SOMAC

A. *General* The "currency" is based on a value of 10 for a pawn, of 1 for an attack on a square by a piece, and of 5 for having the short-term "initiative". These relative values are about right, judging by the way SOMAC plays.

B. White scores −5 if his opponent can gain the initiative (i.e. force White to move a particular piece) by a pawn move. A several-move analyser could probably dispense with this rather arbitrary addition.

C. This is a rule from MACHIAVELLI. It ensures that SOMAC will play a mating move if one is available ($x = 0$; $5/x = \infty$). If SOMAC himself is in check, he will move out of check if he can, because in check he has an "exchange value" of $\simeq -1000$. There is no built-in rule to say "I am checkmated".

D. A one-move analyser cannot be persuaded to castle without a special inducement.

E. This, though rather inelegant, is to prevent the Q moving out too soon. Without it, White would, for example, after 1:P–K4, P–K4, play 2:Q–B3?

instead of 2:Kt–KB3. It might be simpler and perhaps more effective, but equally inelegant, to impose a penalty for moving Q before, say, 2 minor pieces have been moved.

F. It would not be worth allowing for pins and transparencies if it were only a matter of calculating the value of the squares attacked; but it is vital in calculating the exchange values. The rule on pins works quite well. The rule on transparencies has snags. The main one is that in which White has castled, and has a K, Q and 2R's only on the back rank. In this case, since the rooks are transparent to the Queen, there is little to encourage the Q to move off the back rank. But this is perhaps a minor snag.

G. This is really the main part of the machine as far as combinational Chess is concerned. It could almost certainly be greatly improved.

H. The Swap-off value of a piece is in effect what a player would lose if his opponent captured it.

I. This is simple. It assumes Black will capture a White piece if he gains material. The +5 is for initiative.

J. This assumes that Black will move or defend the piece with the largest *S*. The 5 is for initiative. The difficulties arise because of "inter-locking" between pieces, with one piece defending two others, etc.