LOGICAL OR NON-MATHEMATICAL PROGRAMMES

By

C. S. Strachey, M.A.
National Research Development Corporation, London, England

A large part of any mathematical programme is concerned with operations which are not strictly mathematical at all. Examples of this are the input and output routines, the arrangements for calling in various sub-routines when required and, above all, the general organization of the problem as a whole. It is an interesting fact that these non-mathematical parts of the programme often take many more instructions than the mathematics proper. As an example, in a problem which involved the step by step integration of a simple non-linear differential equation, the actual integration cycle used 70 instructions, the arrangements to print out the results and to stop the integration at the required point used a further 48 instructions and a printing sub-routine of 64 instructions - a total of 112 instructions. The problem required this integration to be performed a large number of times with different parameters; the organization involved in doing this, and in arranging that the parameters should be fed into the machine in the simplest possible form used no fewer than 250 instructions and sub-routines totalling about 150 instructions. This may be a rather extreme case, but it is generally true to say, I think, that the non-mathematical parts of the programme use far more instructions than one would at first sight expect, and that a relatively large part of effort in preparing the programme is spent dealing with these non-mathematical operations.

The reason for this is twofold. Firstly these operations, being essentially logical in nature and not mathematical, are exceedingly diverse. There is, so far as I know, no really satisfactory notation in use for dealing with them and they are therefore correspondingly difficult to think about. The second reason, is that the machines have been designed principally to perform mathematical operations. This means that while it is perfectly possible to make them do logic, it is necessarily a rather cumbersome process. The difficulty, of course, is to decide what further functions would be of assistance with the logic. The B-line facility of the Ferranti machines is certainly a big step in the right direction, but there is still a long way to go. I do not propose to go any further into this question here, except to say that I think it is a point worth a good deal of consideration in the design of any future machines.

Instead, I want to discuss a slightly different group of programmes: those which are wholly or almost wholly non-mathematical in nature. These programmes are extremely varied and I shall therefore confine myself to describing a few typical ones very briefly.

The first and most generally known type I want to mention is the floating point routine. It is often of great assistance to be able to use a floating binary or decimal point, and for machines which have not got this facility built into the hardware, the simplest way to do this is to use an interpretive floating point routine. A routine of this type takes a single 'instruction' (sometimes called a coded instruction) which is usually of the same general form as the normal instructions and interprets it as the sequence of ordinary instructions needed to carry out the floating point operation called for. A routine of this type makes it as simple to code a programme for floating point arithmetic as for fixed point arithmetic, but it has the disadvantage of being rather slow. A single floating point operation usually takes about 60 single operation times. This means that if speed is important, and it usually is, it is best to arrange that as much as possible shall be carried out by normal uncoded instructions. It is remarkable how far this can be carried - a very large part of the programme consists of counting cycles, performing discriminations and transferring numbers to and from the accumulator and multiplier register. If the floating point programme is arranged so that these can all be done by normal instructions, remarkably few coded instructions will be necessary. This means arranging for the stores used as the floating point accumulator and multiplier registers to be always in the standard form so that direct transfers are possible, and choosing the standard form so that the numerical part of the number is in the most significant and the exponent in the least significant part of the combined number so that normal discriminations on sign can be performed. With these precautions, an interpretive floating point programme need only take about ten times as long as the corresponding programme using a fixed point.

Another interesting group of logical programmes are those intended to assist the programmer. There is one just coming into use at Manchester which will perform the entire coding of a problem from a sequence of quasi-algebraic symbols. In order to use this programme properly, it is still necessary to have a good working knowledge of the machine, and the symbolism it uses is not exactly that of normal mathematics, but it certainly makes the preparation of simple programmes an extremely easy and painless affair. This programme is

a first and very important step, I think, towards the ideal stage of getting a machine to accept ordinary mathematical equations in the normal symbolism and from them to prepare its own programme. Programmes of this type whose function is to take over some of the work now done by the programmer will, I hope, become increasingly common.

Finally I should like to describe in slightly more detail a programme I have just completed which makes the machine play a game of draughts.

The game of draughts occupies an intermediate position between the extremely complex games such as chess, and the relatively simple games such as Nim or Noughts-and-Crosses for which a complete mathematical theory exists. There are several programmes in existence which will play the simple games but most of them use the complete mathematical theory so that the outcome of the game is no longer uncertain. In spite of a good deal of newspaper comment to the contrary, I do not believe that a programme has yet been constructed which will play a complete game of chess. The nearest approach I know of is a programme for the Manchester Machine by Prinz which will solve two move chess problems, subject to certain restrictions. This programme might be adapted to play a complete game, but it would be quite intolerably slow.

Chess presents two quite different sorts of difficulty which do not appear in the simple games. The first is that there are many types of men and moves so that at any stage there are a great many possible choices. The second is that as there is no complete theory of the game, it is necessary for the machine to look ahead for a number of moves and choose its move by some scheme of valuing the resulting position. The second of these difficulties is of much greater theoretical [illegible] than the first, but it is the first, I think, which has made it impossible so far to produce a chess-playing routine.

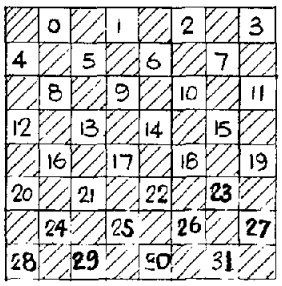
For this reason, I have considered the much simpler game of draughts. In this game the moves are relatively simple, but it is still necessary to make the machine look ahead and choose its moves by a valuation scheme. I have succeeded in making a programme for the Manchester Machine which will in fact play a complete game of Draughts at a reasonable speed. This programme is a fairly typical example of a large logical programme.

Representation of a Position

Draughts is played on the 32 white squares of a chessboard. For convenience we number these from 0 to 31 as shown in Fig. 1.

There are only two kinds of pieces for each player - men, who can only move forwards, and Kings who can move forwards or backwards. We can represent a position completely by three 32-digit binary numbers which we shall

BLACK



WHITE

FIG. 1

call B, W and K. Each digit in these numbers represents a square in the natural order - that is to say the least significant digit represents square 0 and the most significant digit square 31. The number B gives the positions of the black men and black Kings, W gives the positions of the white men and Kings, while K gives the positions of the Kings alone of both colours. In each of the numbers a 1 indicates the presence and a 0 indicates the absence of the appropriate type of man. Thus for example the position in fig. 2 would be represented by

B = 1100 0000 0000 0000 0000 0000 0000 0000
W = 0000 0000 0000 0000 0000 0000 0000 0011
K = 0100 0000 0000 0000 0000 0000 0000 0010

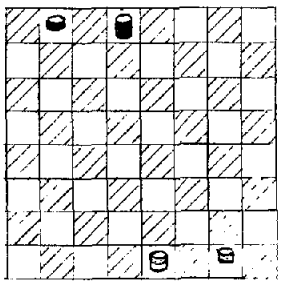


FIG. 2

The positions of the Black Kings are given by the word B & K while the empty squares are given by $\sim$ B & $\sim$ W. This representation is not the most economical possible - it allows eight possible representations for each square of which only five are ever used - but it is simple and convenient.

Moves

If we indicate a move by the change in the square number, we see that there are six possible types of non-capture move $\pm$ 3, $\pm$ 4 or $\pm$ 5 (e.g. 14 - 17, 17 - 14, 14 - 18, 18 - 14, 9- 14, 14 - 9) but at most four of these can be made from any square.

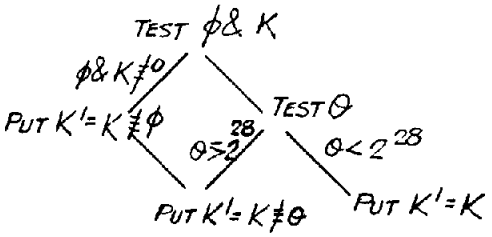
Let us first consider the + 4 moves. These are possible (if the position allows it) from every square on the board except 28 to 31, so that the number M_4 = 1111 1111 1111 1111 1111 1111 1111 0000 represents the squares from which a + 4 move is possible.

If we want to find the possible + 4 moves for black, the number B & M_4

will give the black men which are on sailable squares, while B & M_4 x 2^4 will give the positions to which they will move. The move is only possible if the final square is empty, so that Y = (B & M_4) x 2^4 &~B &~W will give the squares to which black men can actually make a +4 move.

The moves can then be considered individually by taking the non-zero digits of Y one at a time. If 0 is one of these, so that it represents the square to which the black man is moved, $\phi = 0 \times 2^{-4}$ represents the square from which it has been moved. Thus B', the resulting black position, will be given by B' = B ≠ 0 ≠ ϕ

W will be unaltered, but to find K' we must determine if the moved was a King, and if not (i.e. if ϕ & K = 0) whether it has become a King during the move. This latter will be the case if $0 \geq 2^{28}$ - i.e. if it has been moved to the last rank. Thus to find K' we perform the operations:-



If we want to find the +4 moves for white, we must remember that as these are backwards moves, they can only be made by the white Kings so that the expression for the group of possible moves becomes

$$Y = (W \,\&\, K \,\&\, M_4) \times 2^4 \,\&\sim B \,\&\sim W$$

An exactly analogous procedure will apply for the other possible type of move. We can shorten the process somewhat by noticing that while a ±4 move is possible from every square only the combinations (+5, -3) or (+3, -5) are possible it is therefore possible to combine the +3 and +5 moves together without ambiguity and take, for a black move,

$$Y = \{(B\&M_3) \times 2^3 \neq (B\&M_5) \times 2^5\} \&\sim B \,\&\sim W$$

There are thus four types of non-capture moves to be considered for each position:- +4, -4, +3 or +5, -3 or -5. Single stage capture moves (i.e. moves capturing only one piece) consist of two non-capture moves in the same direction, but of different types (e.g.+4 followed by +3 or +5) with the final position empty, and the intermediate one occupied by an opponent's piece. Thus one type of capture move for black will be given by

$$Y = [\{(B\&M_3) \times 2^3 \neq (B\&M_5) \times 2^5\} \&W\&M_4] \times 2^4 \&\sim B\&\sim W$$

These are the moves which start with +3 or +5, followed by +4. There will be three other similar types of capture move.

There is a further difficulty about capture moves. This is that a capture move is not necessarily completed after one stage; indeed in order to avoid being 'huffed', it is essential not to terminate the move until no further captures are possible with the same piece. This feature, which is peculiar to draughts, introduces a very considerable complication into the programme, and there is unfortunately no time to discuss it fully. In outline, however, the procedure is to enter the move finding sequence repeatedly until no further captures can be found.

Valuation of Positions and Strategy

The chief interest in games playing routines is probably in the development of a suitable type of strategy which will allow the machine to play a reasonably good game and at the same time to play reasonably rapidly. It should be possible to graft almost any type of strategy into the move finding scheme outlined above to produce a complete draughts-playing routine and then to evaluate the effectiveness of the strategy by direct experiment. I have done this with two rather simple types of strategy so far, and I hope to be able to try some rather more refined strategies in the future.

For demonstration purposes, and also to ensure that a record of the game is kept, and to take certain precautions against machine error, the move finding sequence and its associated strategy have been combined with a general game playing routine which accepts the opponent's moves, displays the positions, prints the move, and generally organizes the sequence of operations in the game. It is rather typical of logical programme: that this organizing routine is in fact longer than the game-playing routine proper. As its operations, though rather spectacular, are of only trivial theoretical interest, I shall not describe them here.

The first, and simplest strategy to try is the direct one of allowing the machine to consider all the possible moves ahead on both sides for a specified number of stages. It then makes its choice, valuing the final resulting positions only in terms of the material left on the board and ignoring any positional advantage. There is an upper limit to the number of stages ahead that can be considered owing to limitations of storage space, - actually six moves, three on each side, are all that can be allowed. In practice, however, time considerations provide a more severe limitation. There are on an average about ten possible legal moves at each stage of the game, so that consideration of one further stage multiplies the time for making the move by a factor of about 10. The machine considers moves at the rate of about 10 a second, so that looking three moves ahead (those of its own and one of its opponent's), which takes between one and two minutes, represents about the limit which can be allowed from the point of view of time.

This is not sufficient to allow

the machine to play well, though it can play fairly sensibly for most of the game. One wholly unexpected difficulty appears. Consider the position in fig.3.

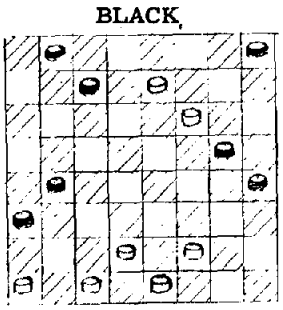


FIG. 3

In this position, the machine is aware that its opponent is going to King next move. Now a King is more valuable than a man - the actual values were 3 for a King and 1 for a man - so that if the opponent Kings the machine effectively looses 2 points. The only way it can stop this is by offering a man for sacrifice, because then, by the rules of the game the sacrifice must be taken at once. If it does this, it will loose only one point, and as it is not looking far enough ahead, it cannot see that it has not prevented its opponent from Kinging but only postponed the evil day. At its next move it is still faced with the same difficulty which it tries to solve in the same way, so that it will make every possible sacrifice of a single man before it accepts as inevitable the creation of an opponent's King. This, of course is a fatal flaw in the strategy - and not one it would have been easy to detect without actually trying it out. An opponent who detected this behaviour - and it is extremely conspicuous in play - would only have to leave his man on the point of Kinging indefinitely, the machine would then sacrifice all its remaining men as soon as the opportunity offered.

In order to avoid this difficulty, the second strategy was devised. In this the machine continues to investigate the moves ahead until it has found two consecutive moves without captures. This means that it will be able to recognise the futility of its sacrifice to prevent Kinging. It is still necessary to impose an over-riding limit on the number of stages it can consider, and once more, considerations of time limit this. However, as no more is continued for more than two stages unless it leads to a capture, it is possible to allow the machine to consider up to four stages ahead without it becoming intolerably slow. This would mean that it would consider the sacrifice of two men to be of equal value to the creation of an opponent's King, and as there is a random choice between moves of equal value, it might still make this useless sacrifice. This has been prevented by reducing the value of a King from 3 to $2\frac{7}{8}$.

With this modified strategy the machine can play quite a tolerable game until it reaches the end game, it has always seemed probable that a wholly different strategy will be necessary for end games. The game given below, which is the first ever played using the strategy, brings this point out very clearly.

MACHINE	STRACHEY
11 - 15	23 - 18
7 - 11	21 - 17
8 - 12	20 - 16 a
12 - 21 (16)	25 - 16 (21)
9 - 14 ! b	18 - 9 (14)
6 - 20 (16,9) c	27 - 23
2 - 7 d	23 - 18
5 - 8	18 - 14
8 - 13 e	17 - 3 (13)
4 - 13 (8)	14 - 9
1 - 5 f	9 - 6
15 - 19	6 - 1 (K)
5 - 9	1 - 6 ?g
0 - 5 ! h	6 - 15 (10)
11 - 25 (22,15)	30 - 21 (25)
13 - 17	21 - 14 (17)
9 - 18 (14)	24 - 21
18 - 23	26 - 22
23 - 27	22 - 17
5 - 8 i	17 - 14
8 - 13	14 - 9
19 - 23	9 - 6
23 - 26 j	31 - 22 (26)
27 - 31 (K)	6 - 2 (K)
7 - 10	2 - 7
10 - 15	21 - 16 ?k
3 - 10 (7)	16 - 9 (13)
10 - 14	9 - 6
15 - 9	6 - 2 (K)
31 - 27 m	2 - 6
27 - 31 m	6 - 10
31 - 26 n	10 - 17 (14)
19 - 23	29 - 25
26 - 31 p	

Notes

a An experiment on my part - the only deliberate offer I made. I thought, wrongly, that it was quite safe.
b Not forseen by me.
c Better than 5 - 21 (9,17)
d A random move (zero value). shows the lack of a constructive plan.
e Another random move of zero value. Actually rather good.
f Bad. Ultimately allows me to make a King. 10 - 14 would have been better.
g A bad slip on my part.
h Taking full advantage of my slip.
i Bad, unblocks the way to a King.
j Sacrifice in order to get a King (not to stop me Kinging . A good move, but not possible before 19 - 23 had been made by chance.
k Another bad slip on my part.
m Purposeless. The strategy is failing badly in the end game
n Too late.
p Futile. The game was stopped at this point as the outcome was obvious.