6

EXPERIMENTS WITH A LEARNING COMPONENT IN A GO-MOKU PLAYING PROGRAM

E. W. ELCOCK and A. M. MURRAY
UNIVERSITY OF ABERDEEN

INTRODUCTION

This paper is a report on some preliminary work undertaken as part of a longer term study of the problems which arise in designing and implementing digital computer programs which 'learn'.

A program has been written which learns to play the board game 'Go-Moku' using a particular learning mechanism to be described later. The program is to be regarded as an experimental tool by means of which the particular learning mechanism can be investigated in some depth.

Go-Moku is a simple but not a trivial game with an intellectual content comparable with a game of draughts (checkers). Opinions have sometimes been expressed that there is nothing to be learnt (no pun intended!) by programming simple games. With this we disagree. Present knowledge of programming learning is such that it is useful to experiment with programs operating in a simple task environment. It is not so much what game the program learns as how it learns it. It is emphasised that the object of the present work is not to write a program which plays a difficult game better than anyone or anything has played it before, but to isolate and investigate particular aspects of a learning process which might be valid over a range of ill-structured problems. For the record, however, the current learning programs learn to play a good (basically defensive) game. The modifications currently being made to the program should give it a learning capacity to become unbeatable.

GO-MOKU: THE RULES AND AN EXAMPLE

For readers unfamiliar with the game of Go-Moku, the rules are given below. An example game is given to illustrate the kind of motivation that might lie behind move making. It is hoped that this will provide a background for discussion of the learning program which follows.

The game is played on a 19×19 square mesh (Fig. 1). Player $b(w)$ has a supply of indistinguishable black (white) pieces. The players take it in turns to play a piece on a lattice point. The winner is the first to complete a

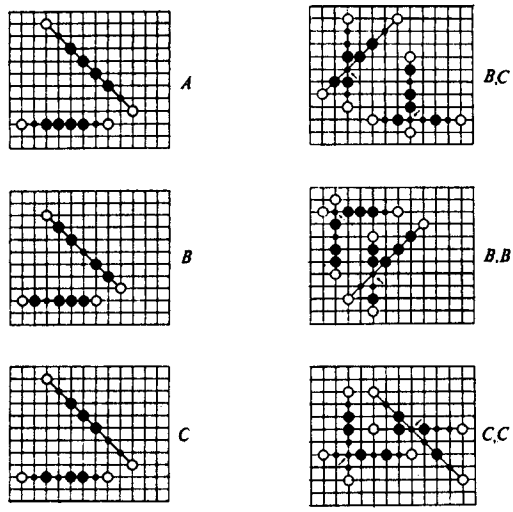

FIG. 1.

(horizontal, vertical or diagonal) line of five and only five adjacent pieces of his colour.

One or two elementary remarks might be useful. Pattern A (Fig. 1) is a winning pattern. It can be made into a five by adding a piece at either end: the opponent cannot kill both. It follows that patterns such as B and C are *forcing* patterns: the opponent must reply to them immediately. It follows that patterns such as B,B C,C and B,C are winning patterns with black to play. By playing moves such as those indicated he simultaneously creates two forcing patterns.

At a more advanced level of play the advantage of being first player is compensated for by adding a further rule, to the effect that the opening player may not play a move which is a component of two forcing patterns of the type C, i.e., moves of the kind indicated in the patterns C,C.

This additional rule has not been implemented in the work reported here.

And now for some comments on the example game illustrated in Fig. 2.

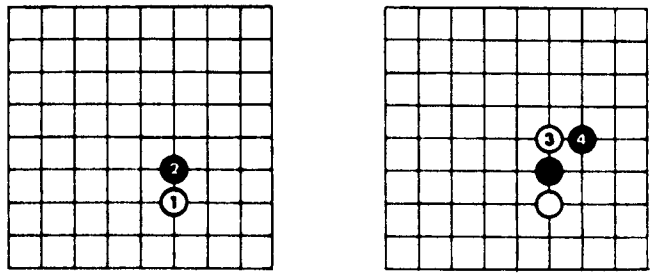

FIG. 2.

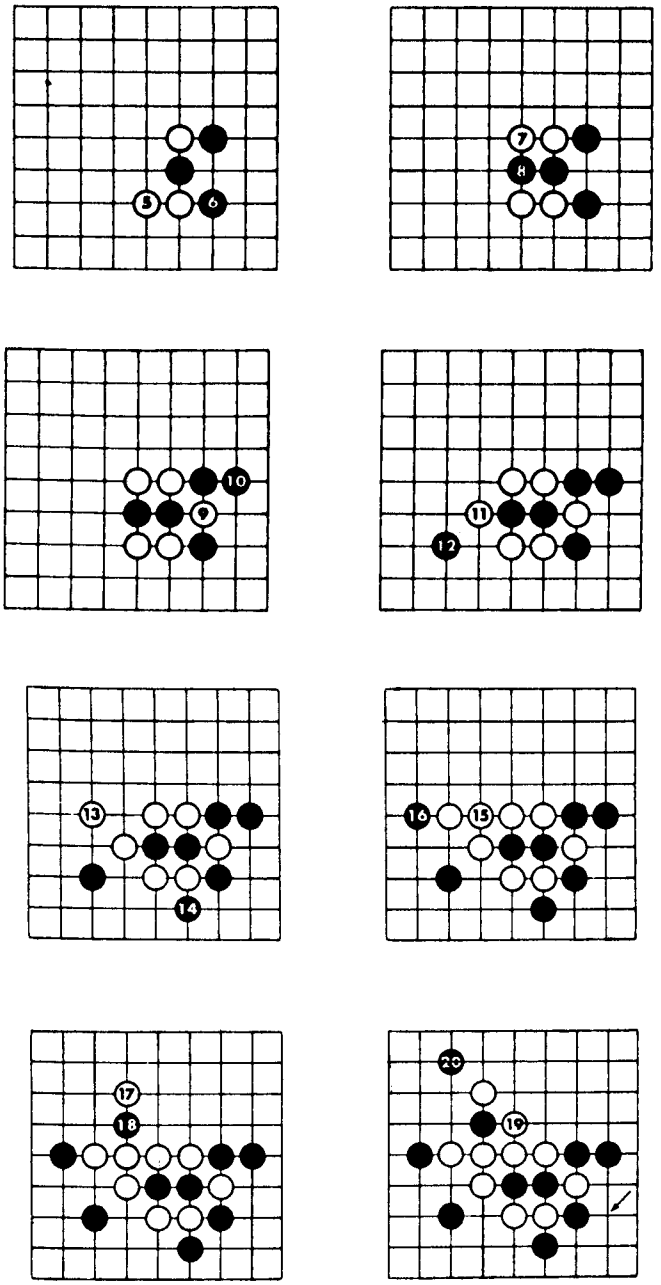

FIG. 2 (*cont.*)

The opening moves are characterised by the fact that it is usually dangerous to play too far away from one's opponent. White's moves 3, 5, 7 and 9 are motivated by a desire to kill his opponent's tentative line patterns whilst still preserving his own. The move 9, in fact, achieves this object. At this

point white has active open patterns along the two diagonal lines through 9, whilst black has no active open patterns.

Black now plays 10 to kill one of White's open patterns—a weak defensive move.

White plays 11 to create a very strong (as we shall see) group of forcing moves around the left margin of the played pieces.

Black (unwisely) ignores the threat and plays the (weak) development move 12.

White can now win by a sequence of forcing moves: 13 (making up a pattern of type C) forcing 14; 15 (type B) forcing 16; 17 (type CC) forcing 18; 19 (making up a winning pattern of type A) forcing 20, etc.

Informal description of the learning mechanism

We shall start by considering just what is involved in writing a program to play Go-Moku. Most of what follows is applicable to two-person board games generally, and many of the basic ideas are not new. Indeed, our debt to previous workers (particularly Shannon, Samuels, Newell, Simon and Shaw) is so pervasive that no attempt will be made to give detailed acknowledgements. As far as we know, however, the particular learning mechanism isolated and investigated in the present program, the method we have called 'backtrack analysis', is new.

At any stage of the game there is a large number of possible moves (all vacant lattice positions). The first problem is to describe the current board situation in such a way as to make possible:

(i) adequate discrimination between possible moves;
(ii) an ordering (evaluation) of discriminated moves.

A further general requirement on the description (and indeed *all* processing) is that it should be sensible in the demands its implementation makes on computer time. This (debatable) requirement was not made for financial (or social) reasons, but was one which we felt made sense in relation to the learning process as a whole: if the learning mechanism was not powerful enough for the learning program to meet this requirement, then it had no place in the immediate project.

Let us suppose that, as a result of our own experience of playing Go-Moku, we have formed certain ideas about the kinds of description which might be appropriate.

If we use a description which is sufficiently simple for the ranking of discriminated moves to be obvious, then it is a straightforward business to write a program to play on the basis of this ranking. We have, in fact, implemented such non-learning programs to play Go-Moku. The strength of a playing program, however, resides largely in the power of the descriptive machinery. If more powerful descriptions are used, allowing more subtle discriminations between possible moves, it becomes increasingly difficult and

sometimes quite impracticable to rank the discriminated moves by direct inspection, and consequently impossible to make use of the description at all in a non-learning program.

At this point it becomes imperative, if one is to progress, to extend the concept of the game-playing program to include an automatic generation of an ordering of discriminated moves. An immediately attractive way of doing this is to get the program to analyse its played games and obtain the required ordering as a result of its accumulated experience of the analyses. There are various ways in which the descriptive and analytic components of the program might be designed to make this possible. One outstandingly successful method is that used by A. L. Samuel in his now classic draughts-playing program. The method isolated and studied in the present work is a deductive backtrack analysis of the kind a human game-player might make. Informally described, it goes as follows:

A game between the learning program (LP) and an opponent is played to a win. In a sense the LP now asks itself: 'Where did I (or my opponent) go wrong?' The LP performs a backtracking *analysis* to find the critical board situation from which it sees the win as inevitable. The LP then attempts to describe the particular *abstracted* board situation in its formal descriptive language. If the formal language has been suitably designed, then this description will automatically *generalise* the abstracted board situation. Finally the ordering of discriminated moves is adjusted in the light of this analysis, abstraction and generalisation.

It should be emphasised that the component of the LP which generates the ordering is *not* tied to any one formal description of the current state of the board, except in the minor demands it makes on the way this description is presented during implementation.

If the primary task *was* simply to design a game-playing program, then the LP not only makes it possible to use a powerful descriptive language, but gives one the facility for experimenting with different descriptive languages in a search for the most powerful. It is hoped that these features of the present work will carry over into less trivial tasks, and that the deductive backtrack analysis will prove a useful component in other learning programs.

THE LEARNING PROGRAM

As indicated in 'Informal description of the learning mechanism' (p. 90), the aim is to isolate and study the backtrack analysis component (BAC) as a learning mechanism for generating an ordering of discriminated moves.

The BAC operates in the context of a formalism for describing possible moves in such a way as to allow a description of a particular move to be elaborated through a sequence of increasingly sophisticated levels of discrimination. This formalism will be defined and discussed in 'Description of possible moves' (below). 'Program play' (p. 94) discusses, briefly, how in a game played by the LP a move is selected using an ordered list of

described moves. 'Backtrack analysis' (p. 94) discusses in detail how this ordering is generated by the BAC.

Description of possible moves

A description of a class of possible moves is made up of a number of component sub-descriptions, referring to particular features. It is supposed that, for any one feature, the possible values of the feature can be ordered by a relation 'better than' ('contains'). For example, consider the (imprecisely) defined feature: 'the maximum number of played pieces of the same colour in a possible winning 5-pattern which includes the move'. The possible values of this feature are 1, 2, 3 or 4. The relation 'better than' might be taken as integer magnitude.

An ordering is now defined over the values of a description. If D' and D'' are two values of a description, the relation 'better than' holds between D' and D'' if and only if it holds between each component feature.

More formally: An n-description, D^n, of a class of possible moves, is an ordered n-tuple of components d_i, $1 \leqslant i \leqslant n$. For each i, $1 \leqslant i \leqslant n$, there is an ordering (written $\supset^i$, and loosely thought of as 'better than' or 'contains') defined over the set of values of d_i.

An ordering (written $\supset$ and again thought of as 'is better than') is defined between two descriptions $D_\alpha{}^m$ and $D_\beta{}^n$ if and only if $m=n$, when

$$D_\alpha{}^n \supset D_\beta{}^n$$

if and only if

$$d_{\alpha 1} \supset^1 d_{\beta 1},\ d_{\alpha 2} \supset^2 d_{\beta 2},\ \ldots,\ d_{\alpha n} \supset^n d_{\beta n}.$$

It should be noted that **not** $(D_\alpha{}^n \supset D_\beta{}^n)$ does not imply $D_\beta{}^n \supset D_\alpha{}^n$.

Both in the interests of giving flesh to this skeleton, and of providing background in terms of which the actual performance of a particular LP can be discussed later, an example sequence of actual descriptions used in an LP follows.

An actual description

Begin by defining:

A 7-pattern is the pattern of played pieces on a line of seven adjacent lattice points.

A black 7-pattern is an integer, N, where N has the value 'if an end point is black or an interior point is white then zero, else one plus number of black pieces' (i.e., zero if the interior 5 cannot be made into a winning black pattern else a measure of completion of the pattern).

A line pattern on a given line through a given vacant lattice point is the pattern of played pieces on the given line through the lattice point.

A black line pattern is an ordered pair of integers, $n : r$, where n has the value of maximum $[N]$ over the set of black 7-patterns for which the given vacant lattice point is an interior point, and r is the number of successive

1-translations of a 7-pattern that can be made keeping $N=N_{max}$ (see examples).

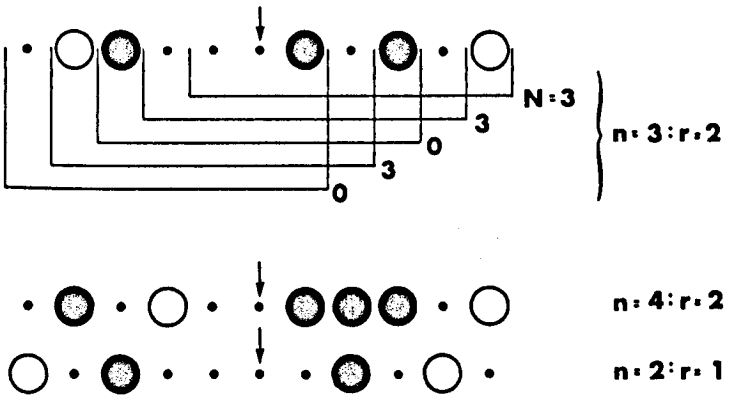

FIG. 3.

Fig. 3 shows some examples of line patterns: r gives a measure of the number of different winning patterns to which a piece played on the vacant lattice point, together with a largest group of already played pieces on the same line, can contribute.

If an ordering is defined over the set of ordered pairs n,r then a black line pattern is a description D^1. We will take the ordering to be first by magnitude of the integer n and then by r. This description allows adequate discrimination of play on a given lattice point with regard to the given line. If and only if a move completes a winning pattern on the line then it has the description 5 : 1. The description also gives a sensible ordering: a 4 : 2 (a winning move) is 'better than' a 4 : 1 (not a winning move); a 3 : 2 is 'better than' a 2 : 3.

There are four black line patterns through any vacant lattice point.

The level 1 description, D^1, of a possible move is defined to be the best (see 'Description of possible moves' (p. 92)) black line pattern through the given vacant lattice point.

The level 1 description cannot give an adequate (sufficient) description of winning moves such as B,B C,C and B,C (Fig. 1) which depend on the pattern of played pieces on two lines intersecting in the considered lattice point. Such patterns can be described by extending the description of a possible move by another D^1.

The level 2 description, D_2, of a possible move is defined to be the second best line pattern through the given lattice point. The description $D_1 \times D_2$ (a D^2) is now capable of discriminating and ordering moves such as those of B,B, C,C and B,C (Fig. 1).

This kind of description can be and has been extended further but the above two levels provide an adequate background for the next two sections. More sophisticated levels of description will be given when needed.

Program Play

In a particular game, move selection is essentially governed by the LP's current list of subgoals. It is supposed that these subgoals are descriptions of moves from which a win is expected to be inevitable. The current list of subgoals has been generated in a way that will be discussed in detail in the next section. The list governs which of the described possible moves in the given game-state is finally selected for play. There is an implicit ordering of subgoals on the list according to the number of moves away from a win. This ordering is natural: it allows the most economical of a number of alternative winning moves to be selected. Equally important, it makes the choice between attacking and defensive moves straightforward; if black is to play and white has a higher level subgoal than black, then black must play (defensively) on the lattice point which gives white this higher subgoal.

Thus, the task is to select a move from a list of possible moves. Each move on the list is processed to find its description. The list of subgoals is now scanned by level and, for each subgoal, a comparison is made between the description of the subgoal and the described possible moves. The highest level acceptable attacking or necessary defensive move is selected.

Finally, there is a default move. If the search procedure does not find a move which achieves a subgoal then a default procedure is called to select a move. The default procedure may use quite different criteria or no criterion at all (e.g., selection using a pseudo-random number generator).

THE BACKTRACK ANALYSIS

A number of backtrack analysis components have been experimented with at various levels of complexity. One of the simpler, implemented originally as part of a feasibility study for the present LP, has formed a basis for later elaborations. We will begin with a discussion of this simple BAC and its limitations.

A simple BAC

This simple BAC works in the context of a fixed (static) description of possible moves. The flow is given in Fig. 4.

The BAC is only called when the LP has *lost* a game. This is important since it then follows that the loser's moves were played on the basis of the LP's current list of subgoals, and therefore need not be re-examined during the backtrack—they are the 'best' moves the LP could make. Removal of this restriction requires a more sophisticated BAC.

The BAC is entered with a list of played moves in reverse order, i.e., winning move at head of list. 'Level' refers to the ordering of subgoals on the list of subgoals by number of moves away from completion of the winning pattern. The level of the last subgoal played in the BA is held in 'old level'. When, in the course of the BA, a move which is not a subgoal

is encountered, the description is added to the list of subgoals at a level one below 'old level'—it being assumed that the new subgoal is a necessary and sufficient step in the formation of the higher one.

In general, this last assumption is clearly not justifiable. It is this requirement, that the description added to the subgoal list should be necessary and

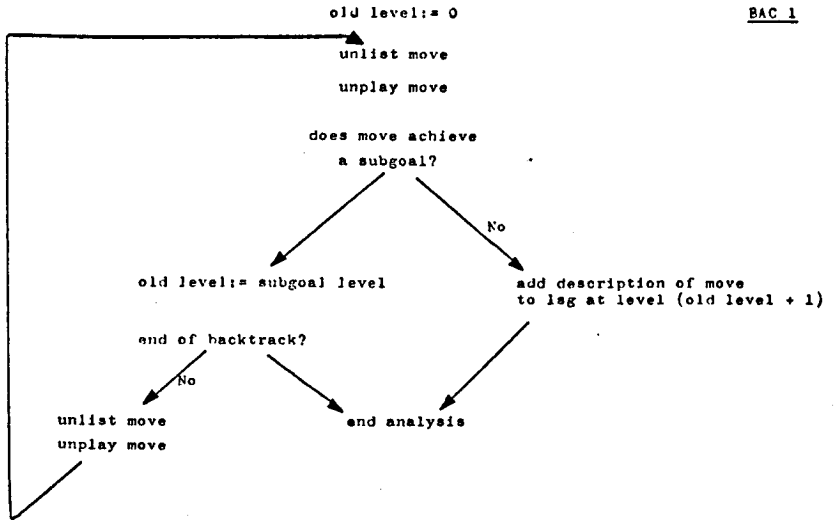

FIG. 4.

sufficient to ensure a transition to a higher subgoal, that lies at the heart of the design of a BAC. We will approach the problem through a sequence of elaborations of BAC 1; each elaboration going some way further to meet the requirement.

Some extracts from an actual teaching sequence with an LP using BAC 1 and 'random' play as a default move are given in the Appendix. These extracts are given both for their intrinsic interest and to provide a base-line from which to consider results obtained with later BACs.

BAC 2 and BAC 3: minimal descriptions

These BACs still operate in the context of a fixed description of possible moves. They go some way towards meeting the requirement that the descriptions of particular subgoals on the list of subgoals should be *necessary* (minimal).

Consider a played game in which a particular move achieves a new subgoal for which a *sufficient* description can be given. If D_{min} is the necessary (minimal) description of the subgoal, and D_{move} is the description of the move, then in general we have $D_{move} \supset D_{min}$ (D_{move} 'contains' D_{min}) but *not* $D_{move} = D_{min}$. We would like to have D_{min} added to the list of subgoals. BAC 1 adds D_{move}.

If D^1, D^2 are descriptions such that $D^1 \supset D^2 \supset D_{min}$ and $D^1 \neq D^2 \neq D_{min}$ then, with BAC 1, if the sequence of teaching games is such that D^1 is encountered before D^2, both D^1 and D^2 will be added to the list of subgoals. Both D^1 and D^2 describe the same subgoal and neither is a minimal description. Experiments with BAC 1 showed that this is what usually happens. Without attempting to give a precise explanation: it is a consequence of the fact that games in a teaching sequence tend to become less and less 'open' as the program learns.

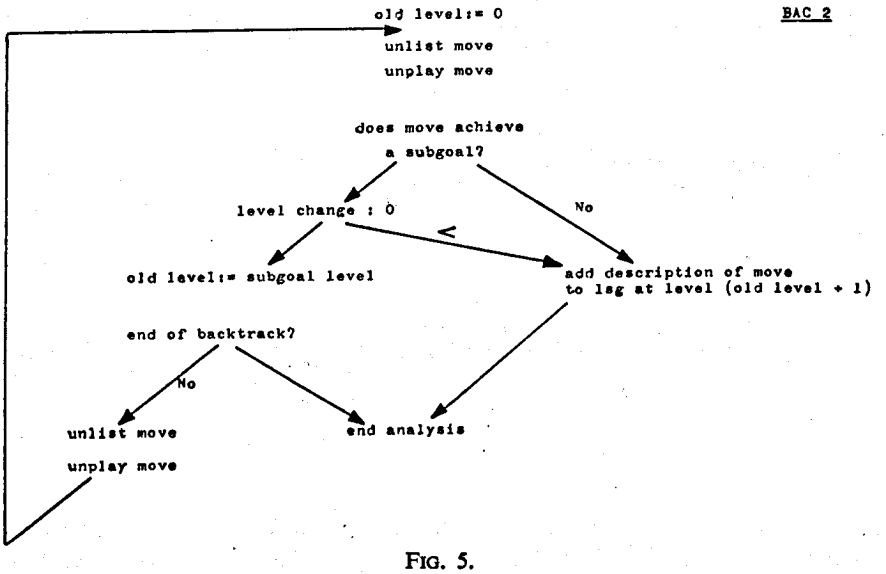

FIG. 5.

BAC 2 finds *minimal* descriptions by a straightforward modification of the action of the component, 'Add description of move to list of subgoals at level (old level+1)', of BAC 1. Suppose the description D_{new} is to be added at a given level to the list of subgoals. D_{new} is added and, as part of the re-ordering of the list of subgoals, all those descriptions at the given level which 'contain' D_{new} are deleted. By this simple device, and over a sequence of teaching games, the descriptions on the list of subgoals approach minimal descriptions.

The experiments with BAC 1 also showed that, when subgoals are insufficiently described, their partial descriptions are often discovered for the first time in a BA at a level lower than their 'true' level. In BAC 2 if, on backtracking, the level change (old level—level+1) between two successive subgoals in the backtrack sequence is found to be negative, then the list of subgoals is amended to bring the lower subgoal up to a level one below the first.

Finally, it is interesting to try to obtain more information from each BA and not simply end the analysis at the first re-organisation of the list of subgoals.

Consider the BA when the white move (n) is discovered as achieving a new subgoal. The list of subgoals is modified. Would the LP move ($n-1$) still make sense if LP had been playing from the new list of subgoals? Yes: if and only if D_{LP} ($n-1$) 'contains' $D_W(n)$. Now this is most unlikely to happen for an LP whose default move selection is random. If, however, the default move selection is 'Play the best D^2 (own if equal black and white alternatives) on the board', then this situation can and frequently does occur. The modification of BAC 2 which makes this extended analysis possible is called BAC 3 and its flow is given in Fig. 6.

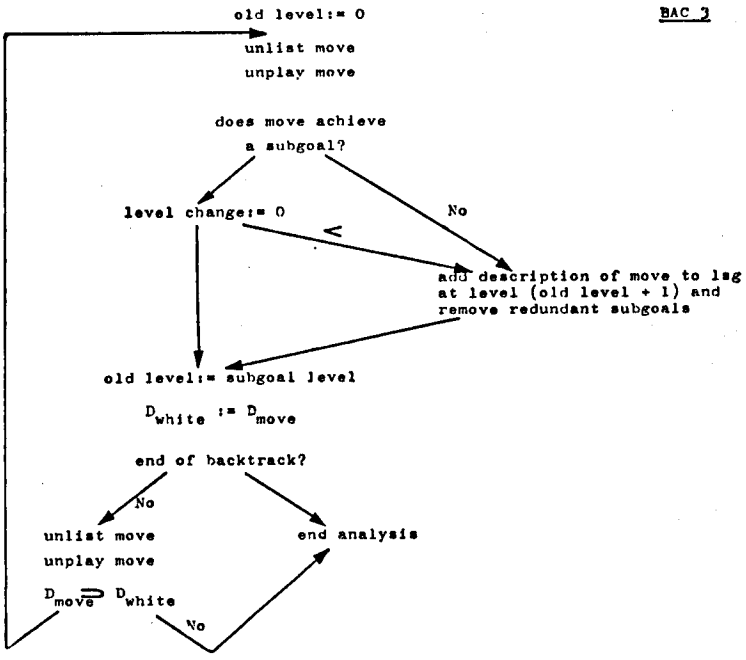

FIG. 6.

The selection of results given and discussed in the Appendix are from a teaching sequence with an LP using BAC 3 and a default move of the kind indicated above. It should be emphasised, however, that the final list of subgoals obtained in a teaching sequence is not significantly dependent on the default move selection.

BAC inadequate descriptions

Learning programs based on BAC 1, 2 or 3 all reach a stage at which they can learn nothing more from backtrack analyses of lost games. All minimal descriptions of subgoals which can be adequately described have been obtained. The opponent wins (*when* he wins!) by a move which achieved a described, but *inadequately* described, subgoal on the list of subgoals: the backtrack analysis cannot even recognise that there is a critical board situation, let alone resolve it.

The work going on at present is aimed at extending the BAC so that in a BA:

(i) inadequate descriptions of subgoals are recognised as such and,
(ii) following such recognition, it is optional to extend the formal descriptive language, in the light of the BA, in an attempt to make an adequate description of the new subgoal possible.

Recognition of inadequate descriptions

There are a number of ways by which the LP might detect whether or not a description is inadequate. We will indicate briefly two quite different approaches.

(i) When the LP is playing a game, it keeps a record of any of its own attacking moves played from its current list of subgoals (lsg). The BAC performs a similar analysis to BAC 3 but, in addition, now:

If the LP has lost, then the BAC looks at the record of own-moves which achieved an attacking subgoal. These subgoals were supposed to be winning subgoals. Their presence on the record indicates that the descriptions must be inadequate.

If the LP has won, then the BAC simply backtracks through the LP's moves, monitoring the sequence of subgoal levels. If a subgoal is encountered at a higher level than the previous subgoal in the backtrack, then this last subgoal is inadequately described, in that achieving it has not inevitably created a subgoal at a higher level.

(ii) Here the idea is that when, on backtracking, the BAC recognises a new subgoal, the new subgoal is subjected to an *immediate*, though not infallible, test for inadequacy. The test consists in generating a number of 'minimal' (in the sense of number of played pieces) examples of board situations which answer the description of the new subgoal. Starting from an example board situation, the BAC now plays alternate own and 'opponent's' moves, using the LP's current lsg to select both, to see if a win can be achieved in the number of moves implied by the level at which the new subgoal has been discovered. Depending on the results of this test for a number of 'randomly' generated examples, the new subgoal can be added to the lsg with the appropriate indication 'adequate' or 'inadequate'.

Method (i) has the advantage that its incorporation in the BAC is essentially independent of the particular descriptive language of the LP. It has the disadvantage that recognition of inadequacy may possibly not be obtained in the same game-analysis in which the new subgoal is discovered and added to the lsg. Indeed, recognition may be deferred several games, with undesirable implications for play and analysis in the intervening games.

With method (ii) the recognition of inadequacy can often be made immediately a new subgoal is discovered. On the other hand, a direct interpretation of method (ii) implies that each time the descriptive language is changed or extended, the processing unit in the BAC which generates 'random' examples has also to be changed or extended.

Both methods have been implemented and tested in the context of the simple descriptive language used for illustration in this paper. Both are being studied further in the context of more sophisticated descriptions (cf. 'Extension of inadequate descriptions' below).

A variant of method (ii) which overcomes the disadvantage of *direct* generation of 'examples' is also being investigated. A single minimal example is generated from the *particular* critical board situation met in the BA. Starting from this critical board situation, a piece '*x*' on the board is 'unplayed'. If the description of the move which achieves the new subgoal is unchanged as a result of unplaying '*x*' then '*x*' is left off the board; else '*x*' is replayed. This is done for all the pieces of the critical board situation. The pieces remaining on the board represent a minimal example of the described new subgoal. The example is now played to completion by the BAC as before. This variant of (ii), like (i), has the advantage that its incorporation in the BAC is independent of the particular descriptive language of the LP.

Extension of inadequate descriptions

BAC 1, 2 and 3 operate in the context of a *fixed* description of possible moves. With a BAC which is capable of recognising when a description is inadequate, it is desirable to be able to operate in the context of an *extendable* description. Thus, when the BAC indicates that a description of a new subgoal is inadequate, it should be possible, either immediately or at a convenient later 'break point', to add a new processing unit to the descriptive component; the BAC operates in what is essentially a conversational mode.

With a description scheme of this kind, it is interesting that the lsg can be used, not only in selecting a move from a list of promising moves, but also to control the amount of processing that goes into the actual description of particular promising moves. Briefly, in selecting a move from the list of possible moves, the moves on the list might first be described at the simplest level of description. During the comparison of these described moves with the descriptions on the lsg, contracted lists of 'promising' moves (moves whose simple descriptions contain the relevant component of an extended description of a subgoal) might be formed. These lists would be processed to the appropriate next level of description (which might be different for different lists) if and only if no subgoal is encountered at the simpler level of description. In this way the costly processing associated with obtaining sophisticated descriptions can be automatically controlled through the lsg. This is clearly feasible when the ordering of the lsg by (i) number of moves away from a win, and (ii) complexity of description, go hand and hand. The necessary control when this criterion is not satisfied is more difficult.

Work is in progress on a BAC which incorporates the components discussed in 'Recognition of inadequate descriptions' (p. 98) and which will be used to investigate the problems of operating in this kind of 'extendable description' mode.

ACKNOWLEDGEMENT

The work of the project has been supported in part by a grant from the SRC and the authors would like to take this opportunity of thanking the SRC for this assistance.

APPENDIX: EXTRACTS FROM RESULTS BAC 1 and 3

Note: the symbol * marks moves selected through use of the list of subgoals.

GO-MOKU 24 : 8 : 65 (BAC 1): Series 1

Newgame 1: current list of subgoals is: 5 : 0, 0 : 0
white begins: sequence of moves is:
8, 8 (8, 7) 9, 8 (6, 8) 10, 8 (7, 6) 11, 8 (12, 8*) 7, 8 wins

Game analysis is:

move 7, 8 achieves subgoal 5 : 0, 0 : 0 at level 1: backtrack
move 11, 8 does not achieve subgoal: description 4 : 1, 1 : 3 added to list of subgoals at level 2

Newgame 2: current list of subgoals is: 5 : 0, 0 : 0
4 : 1, 1 : 3

white begins: sequence of moves is:

8, 8 (7, 7) 9, 9 (9, 6) 10, 10 (6, 7) 11, 11 (12, 12*) 10, 9 (8, 7) 9, 7 (7, 6) 7, 9 (8, 9*) 6, 10 (5, 11*) 10, 6 wins

Game analysis is as follows:

move 10, 6 achieves subgoal 5 : 0, 0 : 0 at level 1: backtrack
move 6, 10 achieves subgoal 4 : 1, 1 : 3 at level 2: backtrack
move 7, 9 does not achieve subgoal: description 3 : 2, 3 : 1 added to list of subgoals at level 3

new list of subgoals is: 5 : 0, 0 : 0
4 : 1, 1 : 3
3 : 2, 3 : 1

. . . . . .

Newgame 8: current list of subgoals is: 5 : 0, 0 : 0
4 : 1, 1 : 2 4 : 1, 1 : 3
3 : 2, 3 : 1
4 : 0, 2 : 1 3 : 2, 2 : 3
3 : 2, 2 : 2
2 : 3, 2 : 3

white begins: sequence of moves is:

8, 8 (8, 7) 9, 9 (9, 8*) 10, 10 (7, 7*) 11, 11 (12, 12*) 10, 9 (10, 8*) 10, 11 (10, 12*) 11, 9 (8, 9*) 9, 11 (8, 11*) 8, 12 (7, 13*) 12, 8 wins

Game analysis is: . . . . . .

. . . . . .

Comments

Game 1. The LP plays 'at random' until it recognises (too late!) the winning move 12, 8. The LP plays 12, 8 but white wins with 7, 8. During the BA the LP recognises the move 11, 8 as creating a subgoal (the winning pattern of Fig. 1), and adds the description (4 : 1, 1 : 3) of this move to its list of subgoals.

Game 2. This game is longer than necessary. The sequence of moves up to the LP's move (12, 12*) are simply to illustrate that the simple win of game 1 cannot be repeated.

Game 8. The LP now has an extensive list of subgoals. White however wins by playing 11, 9 at move 13. After white plays 10, 11 at move 11, moves 11, 9 and 9, 11 are winning moves for white. The LP plays 10, 12. The reason is that all three moves achieve (inadequately) described subgoals at the same level (4) on the list of subgoals (lsg). The descriptive language would need to be extended before the BA could resolve such board situations and learn anything significant from this game. Note however:

(i) how the program play is now almost entirely determined by the lsg (move selection from the lsg is indicated by the output of * after the move);
(ii) the existence of non-minimal subgoals on the lsg. Thus, at level 2, 4 : 1, 1 : 3 4 : 1, 1 : 2 and neither is the minimal subgoal for this level (compare with results of BAC 3).

GO-MOKU 2 : 9 : 65 (BAC 3): Series 2

Newgame 2: current list of subgoals is: 5 : 0, 0 : 0
4 : 1, 2 : 0

white begins: sequence of moves is:

7, 7 (8, 7) 8, 6 (6, 8) 8, 8 (6, 6) 9, 9 (10, 10) 9, 5 (10, 4) 9, 7 (9, 6) 10, 8 (11, 9*) 10, 6 (11, 5*) 11, 7 (12, 6*) 12, 8 (8, 4*) 13, 9 wins

Game analysis is:

move 13, 9 achieves subgoal 5 : 0, 0 : 0 at level 1: backtrack
move 12, 8 achieves subgoal 4 : 1, 2 : 0 at level 2: backtrack
move 11, 7 does not achieve subgoal: description 3 : 2, 3 : 2 added to list of subgoals at level 3: backtrack
move 10, 6 does not achieve subgoal: description 3 : 2, 2 : 3 added to list of subgoals at level 4

new list of subgoals is: 5 : 0, 0 : 0
4 : 1, 2 : 0
3 : 2, 3 : 2
3 : 2, 2 : 3

. . . . . .

Newgame 5: current list of subgoals is: 5 : 0, 0 : 0
4 : 1, 2 : 0
3 : 2, 3 : 2
3 : 2, 2 : 3
2 : 3, 1 : 3
1 : 3, 1 : 3
2 : 3, 1 : 2

white begins: sequence of moves is:

7, 7 (8, 7*) 8, 6 (7, 6) 6, 8 (7, 8) 5, 9 (9, 5*) 4, 10 wins

Game analysis is:
move 4, 10 achieves subgoal 5 : 0, 0 : 0 at level 1: backtrack
move 5, 9 achieves subgoal 2 : 3, 1 : 3 at level 5
description 4 : 1, 1 : 3 added to list of subgoals at level 2
new list of subgoals is: 5 : 0, 0 : 0
4 : 1, 1 : 3
3 : 2, 3 : 2
3 : 2, 2 : 3
2 : 3, 1 : 3
1 : 3, 1 : 3
2 : 3, 1 : 2
. . . . . .

Newgame 10: current list of subgoals is: 5 : 0, 0 : 0
4 : 1, 1 : 3
4 : 0, 3 : 1 3 : 2, 3 : 2
3 : 2, 2 : 0
2 : 3, 1 : 3
1 : 3, 1 : 3
2 : 3, 1 : 2

white begins: sequence of moves is:
7, 7 (8, 7*) 8, 8 (7, 8) 9, 9 (6, 6*) 10, 10 (11, 11*) 10, 8 (9, 6) 6, 9 (10, 9*) 9, 8 (12, 8*) 11, 9 (12, 10*) 8, 10 (11, 7*) 9, 10 (9, 11*) 7, 10 (11, 10*) 6, 10 wins

Game analysis is:
move 6, 10 achieves subgoal 5 : 0, 0 : 0 at level 1: backtrack
move 7, 10 achieves subgoal 4 : 1, 1 : 3 at level 2: backtrack
move 9, 10 achieves subgoal 2 : 3, 1 : 3 at level 5
description 3 : 1, 3 : 1 added to list of subgoals at level 3
new list of subgoals is: 5 : 0, 0 : 0
4 : 1, 1 : 3
3 : 1, 3 : 1
3 : 2, 2 : 0
2 : 3, 1 : 3
1 : 3, 1 : 3
2 : 3, 1 : 2
. . . . . .

Comments

Game 2. Illustrates the continuation of the BA after the first new subgoal found. The condition $D_{\text{move}} \supset D_{\text{white}}$ (see Fig. 6) is satisfied for move 11, 5 but not for move 11, 9: the BA therefore stops at move 11, 9.

Game 5. This game is dramatically lost by the LP because it did not recognise a particular example of the winning pattern of Fig. 1. The minimal description of a move achieving this subgoal is 4 : 1, 0 : 0. A non-minimal description learnt in Game 1 is 4 : 1, 2 : 0, and this non-minimal description is on the lsg at the start of Game 5. The move 5, 9 of Game 5, which actually achieves the winning pattern A of Fig. 1, has the description 4 : 1, 1 : 3 which 'contains' the minimal description (4 : 1, 1 : 3 $\supset$ 4 : 1, 0 : 0) but does *not* 'contain' the description on the lsg at level 2 (4 : 1, 1 : 3 $\not\supset$ 4 : 1, 2 : 0). In fact (see analysis record) the move 5, 9 is only recognised as achieving a subgoal at level 5 (4 : 1, 1 : 3 $\supset$ 2 : 3, 1 : 3). The LP has a 'better' move 7, 8, which achieves a higher level attacking subgoal and this is selected for play. The LP learns its lesson however that the BA recognises

that the new subgoal 4 : 1, 1 : 3 should be added at level 2 *not* level 5. In adding the new subgoal at level 2 it notes that the previous subgoal at this level 'contains' the new subgoal, and so is redundant and removed from the lsg. The lsg therefore now has a more minimal description of the winning pattern C of Fig. 1.

Game 10. A similar but more striking example of the convergence to minimal descriptions. White's move 9, 10 achieves a winning pattern of the kind of Fig. 1. The minimal description of this pattern is 3 : 1, 3 : 1. The description on the lsg at this stage is the non-minimal description 3 : 2, 3 : 2. It follows that, during play, the LP does not recognise move 9, 10 as achieving a subgoal at level 3. It is worth noting that there is another description (4 : 0, 3 : 1) on the lsg at level 3. This is a (minimal) description of a winning pattern of the kind CC Fig. 1. Neither description at level 3 is contained within the other. When, as a result of the BA of this game, the description 3 : 1, 3 : 1 is added to the lsg at level 3, the non-minimal description 3 : 2, 3 : 2 is deleted. In addition the minimal description 4 : 0, 3 : 1 is also recognised as being now redundant (4 : 0, 3 : 1 $\supset$ 3 : 1, 3 : 1) and level 3 contracted to a single description which allows all winning patterns of the kind CC Fig. 1 to be detected.