

Implementing Search Heuristics Using the AL1 Advice-Taking System

Ivan Bratko
Faculty of Electrical Engineering and
J. Stefan Institute, University of Ljubljana
61000 Ljubljana, Yugoslavia

The AL1 System (Advice Language 1) was developed as a vehicle for conveying expert knowledge, or advice, to a problem solver. Using chess end-games as an experimental problem domain, this paper concentrates on the question: How to implement chess heuristics using the mechanisms of AL1? The spirit of advice programming as a sort of "tree-search engineering" is illustrated by developing an AL1 piece-of-advice for solving an example problem from the king-knight vs. king-rook chess ending. Experiments indicate that an advice program for this ending, based on comparatively simple concepts, and the human chess master perform comparably.

1. INTRODUCTION

Two main themes motivated the development of the AL1 system (Advice Language 1 [3,6]): (1) To obtain a linguistic vehicle for conveying expert knowledge, or advice, to a problem solver, and (2) to facilitate formal proving of the correctness of problem solving strategies defined by this knowledge. The second of the above two aspects is considered in [1] while here, using chess end-games as an experimental problem domain, we concentrate on a question of "advice programming": How to implement chess heuristics using the mechanisms of AL1? AL1 as a language for describing strategies is an assertional language, as opposed to procedural languages. Characteristic principles and mechanisms employed in AL1 are:

- the division of problem domain into subdomains (separate "advice tables" handle separate subdomains);
- production rules;
- goals, subgoals (goals to be achieved, or maintained, as prescribed by a "piece-of-advice");
- constraints on operators application ("move-constraints" to eliminate meaningless moves from consideration in advance).

The approach used in our experiments is based on the use of benchmark problems and their solutions for the development of an AL1-like representation of appropriate heuristics. The use of an example problem typically results in the

specification of advice, sufficient to solve the problem. The specification of such advice should be mainly based on the concepts already known; new concepts are to be introduced only if practically necessary.

2. TREE-SEARCH ENGINEERING IN AL1: AN EXAMPLE

In this section we illustrate the spirit of "advice programming" by an example problem, i.e. a king-knight- king-rook chess position, together with its solution. As shown by Kopec and Niblett [5] correct play in the KNKR ending is much harder than commonly believed even by chess master's standards. The weaker side, i.e. the knight's side, has to fight for getting, or keeping, its pieces together in order to prevent the opponent's fundamental long-term threat which is: force the king and the knight to separate further and, when the knight is definitely cut-off from its king, attack the knight with both pieces and capture it.

Fig. 1 is a good example of what can happen. The natural move is Ke3 as it brings the Black king towards the knight and centralizes the king. But Kg3 is surprisingly the only move that saves the position. The tree in Fig. 1 illustrates a refutation strategy for Ke3. The depth of this refutation-tree is 20 ply, i.e. the depth at which the knight gets captured. How can we formulate advice in AL1 so that we avoid the 20 ply deep search which is prohibitive.

Two basic mechanisms control the search in AL1: better-goals and holding-goals, both being predicates on board positions. The result of game-tree search in AL1 is a "forcing-tree" (a concept introduced by Huberman [4]) which represents a

detailed strategy for the achievement of better-goals while preserving holding-goals. More precisely: a forcing-tree is a subtree of the game-tree rooted in a current board position. All nodes of the forcing-tree, except the root node, satisfy a given holding-goal; all terminal nodes satisfy a given better-goal, and no other node satisfies the better-goal. For each nonterminal "them"-to-move node a forcing-tree contains all legal successor positions; for each nonterminal "us"-to-move node a forcing-tree contains exactly one successor position. Thus the concept of forcing-tree corresponds to the concept of solution-tree in AND/OR trees [e.g. 7].

The following simple concepts suffice to construct an advice which comparatively efficiently solves the problem of Fig. 1:

- our piece attacked ("our" is the weaker side's);
- "knight-safety" which will be for the present purpose defined as:
N-safety = distance between their king and knight minus distance between our king and knight (distance is measured in king-moves);
- N-lost-2p: we to move cannot prevent the loss of the knight in the next two plies;
- N-mobility: the number of safe squares attainable by the knight in one move;
- N-in-corner: the knight in a corner of the board.

A piece-of-advice that solves our problem of Fig. 1 can be stated as follows: Consider the legal moves in the current position in order of decreasing N-safety. Play the first one of those moves which guarantees that for the next 6 plies:

- (1) we do not get mated, and, unless we can capture their rook;
- (2) we do not lose the knight;
- (3) after the first us-move, N-safety may deteriorate with respect to the current position only if it can be restored on the next move;
- (4) the knight is never placed in a corner of the board.

For the sake of the search efficiency we add a better-goal which prescribes the investigation of "dangerous" opponent's moves only: attacking and knight-constraining moves.

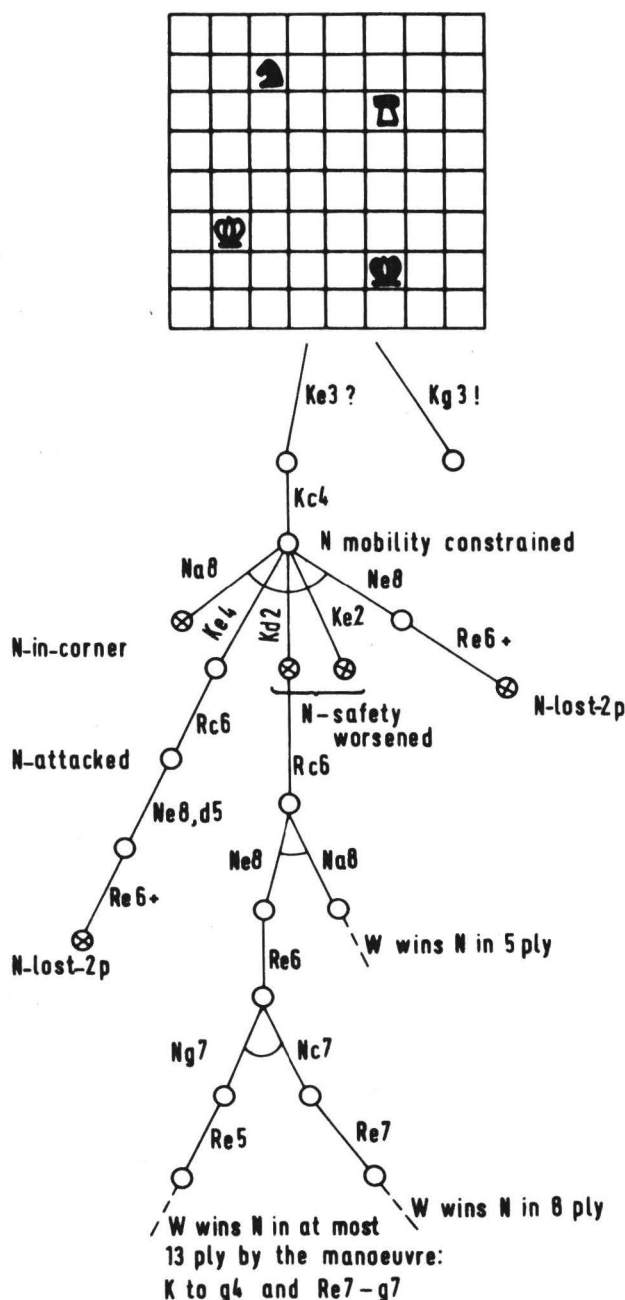


Fig. 1: A refutation-tree for Ke3? . Some obvious branches are omitted.

The translation of such a piece-of-advice into AL1 is straightforward. In an actual implementation, the advice considered above was slightly generalized in order to broaden its applicability.

An advice program for the KNKR ending [2] was developed along the above illustrated lines. The program contains 16 pieces-of-advice which are different combinations of the concepts, considered in the previous example, or similar ones.

Experimental testing of the program [2] and its behavioral comparison with the human chess master convincingly indicate that the program and the human master perform comparably. In fact, the program's performance was slightly better than the master's in these tests. When applying a piece-of-advice, the program never searched deeper than 6 ply, and typically examined a few hundred of nodes, up to about one thousand. The effective branching factor was thus between 2 and 3.

3. CONCLUDING REMARKS

Of interest are the following observations:

(1) Advice, constructed for solving some particular example problem, was usually applicable to many other problems if only some effort was made to state the concepts used in the advice in a general form. An example is in Fig. 2.

(2) Experience obtained in these experiments indicates that, when constructing such knowledge based programs, it is very important to keep the structure of the knowledge neat. This also facilitates formal correctness proofs for strategies, as investigated in [1].

(3) Example of Fig. 1 shows how an automatic facility for learning from examples could be added to the AL1 system in a straightforward manner. In this context, the task of learning from an example can be viewed as one of searching for a compact specification which extracts a proper forcing-tree from the game-tree. This facility has not been programmed; however, part of the KNKR advice program was efficiently constructed by carrying out this process manually [3].

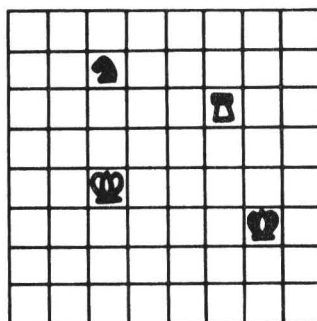


Fig. 2: A position that can occur from the one in Fig. 1; Black to move. The natural 1 ... Kg4 fails to 2 Rc6 Ne8 3 Re6 Nc7 4 Re7 Na8 (or Na6) 5 Rb7 and the knight is stalemated and lost in a few moves. The same advice as for the position in Fig. 1 finds here the only, study-like defence: 1 ... Kh4! .

ACKNOWLEDGEMENT

The author would like to thank professor D.Michie for his continuous efforts in Advice Language research.

REFERENCES

- [1] Bratko I. (1978) Proving correctness of strategies in the AL1 assertional language, Information Processing Letters, Vol. 7, No. 5, 223-230.
- [2] Bratko I. (1979) An advice program for the king-knight vs. king-rook ending, MIP-R-123, University of Edinburgh: Machine Intelligence Research Unit.
- [3] Bratko I., Michie D. (1979) A representation for pattern-knowledge in chess end-games, in Advances in Computer Chess 2 (ed. M.R.B.Clarke), Edinburgh University Press (in press).
- [4] Huberman B.J. (1968) A program to play chess end-games, Technical report no. CS106, Stanford University: Computer Science Department.
- [5] Kopec D., Niblett T. (1979) How difficult is the KNKR ending? in Advances in Computer Chess (ed. M.R.B.Clarke) Edinburgh University Press (in press).
- [6] Michie D. (1976) An advice-taking system for computer chess, Computer Bulletin, Ser. 2, 10, 12-14.
- [7] Nilsson N.J. (1971) Problem Solving Methods in Artificial Intelligence, McGraw-Hill.