EG — A CASE STUDY IN PROBLEM SOLVING WITH KING AND PAWN ENDINGS[1]

Crispin Perdue and Hans J. Berliner
Carnegie-Mellon University
Pittsburgh, PA 15213

We present an overview of the design of a program that playr, simple chess endings with pawns and details of interesting aspects. The program evaluates positions according to production-like rules and also generates moves through the mediation of rules that produce "strategies". Effects of the design are discussed, partly through examples. The design affects the application of standard chess programming principles, among them use of cutoffs, the definition of a repeated position, and the comparison of values of positions. We also describe problems and solutions of problems concerning concepts peculiar to this type of design, especially the concept of search within the context of pursuing a particular strategy.

Introduction

King and pawn endgames are an appealing subject for study for several reasons. Even in very simple pawn endings straightforward searching may not deliver a correct answer. A reason to stop searching may not appear within 15 ply, and even at branching factors of 7 this is too much search. See (Newborn, 1977). These endings are also easily divided into easier and harder classes of problems. The easiest problems can be solved by less sophisticated means and yet they can be solved better by the more powerful techniques that *are* essential in solving the more complex endings.

Because objectives in the king and pawn endings tend to be few in number, but may be several moves away, we felt this was an opportunity to explore analysis of a "strategic" nature. We have also found it possible to substitute calculations for substantial amounts of searching in many situations.

EG has developed over a period of time and changed as it has done so. This development has left behind a series of object lessons in the design of such a program which are more important than the fact that it solves certain problems.

In the presentation, parts of the program will be presented in simplified form. These views will be refined or altered as needed further on.

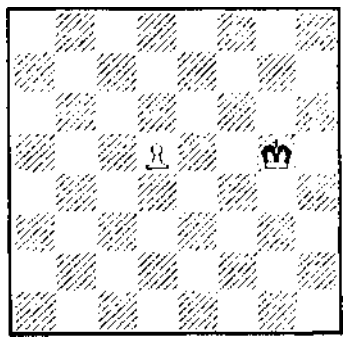
This work was supported by the Advanced Research Projects Agency of the Office of the Secretary of Defense (contract F44620-73-C-0074) and is monitored by the Air Force Office of Scientific Research.

A first view — evaluating positions

1 he information most readily available to the would-be chess programmer relates descriptions of the king and pawn endings positions to facts about them. As available from technical books on the endings such as (Averbakh and Maizelis, 1974), it can usually be described for the computer in terms of ordinary geometrical relations and the "functional" relations of the king and pawn endings (bearing relations on pieces and squares). The descriptions are usually stated in such a way that it is easy to test whether or not a description holds. EG has a set of rules whose condition parts are in this form which it uses to evaluate positions.

Let us look at the rule of the square, Example K If the king is inside the square shown after his move, he will be al„)le to capture the pawn by the time it promotes. The test condition of the rule of the square appears in the condition parts of several rules for evaluating positions.

Example 1:



Uhi te to move

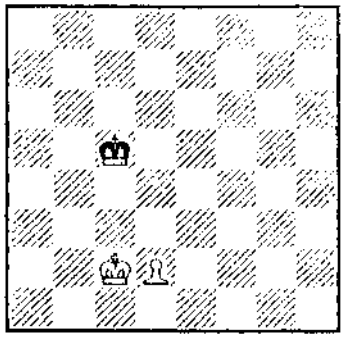
An exceedingly simple yet powerful evaluation rule states that in positions where white has a king and pawn against black's king, white need never lose. (Of course he actually CANNOT lose.) In general we will always refer to the side with a distinct advantage as "white". This convention simplifies the statement of rules.

Looked at from a very simple point of view, EG searches moves proposed by move suggestion rules. Any branch of the search terminates when it reaches a position whose game-theoretic value is known to be either good enough or bad enough to be definitely accepted or rejected by the side on move at the root of the search tree. (For instance, in a position where it is known from the start that one side

cannot win, a drawn position must be considered good enough for that side.) Every position is evaluated by applying the position evaluation rules to it when it is first reached. The values are represented by a pair of values from the set: {black win, draw, white win). The pair represents a range within which EG is sure the game-theoretic value of the position lies. In the simple endings we have investigated a good player can expect to completely analyze a position, so we have tried successfully so far to avoid heuristic values, instead obtaining exact values by analysis.

The information about values of positions thai is obtained from static evaluation is used in several ways to control search. The most obvious use of the information is to prevent search at positions (other than the root) where the true value of the position is Known. The availability of this kind of information can save a great deal of effort, as in Example 2.

Example 2:



White to move

Even though the principal variation from this position to the promotion of the pawn is *71* ply deep, one of the basic "patterns" (position descriptions) for endings with a king a pawn against a king applies after just one ply of search. EG would try the (unique) winning move before any others, but even if they were suggested, any of the other legal moves in this position would be rejected after 1 to 2 ply of search, when positions known to be drawn would be reached.

The winning move is K-B3. The condition met by the position it yields is typical. It says that when one side has exactly one pawn and the other side has only a king, if the pawn is not at the edge of the board, the side with the pawn "has the opposition" *and* his king is more advanced than the pawn, the side with the pawn can force a win.

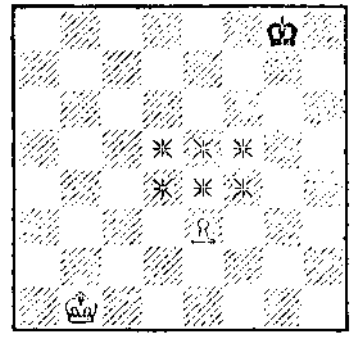
To "have the opposition" is a chess term for a situation where two kings face each other frontally, as after white's correct move, and with an odd number of squares between them. The side not on move is said to be the one having the opposition.

Since the statically determined values of positions are consistent with the game theoretic values, it is possible to use these values to return similarly consistent values from the search. Let us call the first value of the range indicated by a value the "lower value" and the second one the "upper value". The lower value of a node may be set to the maximum of the lower values of its daughter nodes, and its upper value may be set to the maximum of its daughters' upper values.

Sfarchinig_and. strategies

Only moves suggested for a particular reason are considered in any position. This restricts the search further. Example 3 illustrates the structure of EG and some issues of searching with this design.

Example 3:



White to move

1. K-B2 K-B2 2. K-Q3 K-K3 3. K-K4 [win]
K-K2 3. K-K4 K-K3 [draw]
1. K-B2 K-B2 2. K-Q3 K-K3 3. K-K4 [win]
K-K2 3. K-Q4 K-Q3 [draw]

Each of the nodes with no descendants is followed by its value. In this example, the range of uncertainty of each is nil 1 7 nodes are searched including the root. How does FG decide what to try?

All moves are generated by what we call strategies. As part of the problem solver, strategies combine the characteristics of means and goals; a strategy specifies goal information and also is a means for obtaining the situation specified. EG being written in LISP, strategies are implemented as S-expressions which are evaluated and also inspected by EG. When a strategy is evaluated in the context of a particular node in a search tree it produces a list of moves, possibly empty. This list of moves goes into a buffer. If a move is needed and the buffer is empty, a new round of strategy suggestion occurs. If no new strategies are proposed, EG tries no more moves from the node.

The list of moves generated by a strategy at a particular node in a search basically depends on its arguments, which help to define the goal, and on the chess position at the node. Usually either one move or no moves are generated at one time by a strategy.

In Example 3, EG considers six squares to be critical for the white king to reach, shown starred. (Advancing the pawn can only be harmful.) White must bring the king to onr of these or he will only achieve a draw. Of the six, only those nearest to the white king need to be considered; if they fail all the others will also. In this position there *sre* two squares at a distance of three from the white king: Q4 and K4. Two strategies are suggested, one to reach Qfl and one to reach K4. Only moves generated for either one of these two strategies are considered.

White does not adhere strictly to his initial strategy (march to Q4). There is what amounts to a hierarchy of strategies, and the rule which suggests taking of the opposition in the critical situations can take precedence over the basic strategy. Actually, this arrangement has proved to interfere with the improvement of the program. We will be in a better position to explain this once feasibility testing has been presented. (See the section on Calculations.)

In this common type of position where the pawn is not on a rook file black marches toward a square on the file of the white pawn: either the square of the white pawn or *onr* of the squares in front of it. There is a feasibility test based on distances which can declare squares to be impossible for black to reach if white tries to prevent his occupation of them. Black heads for the square of those not ruled out by the distance test which is closest to the pawn. If he fails to achieve a draw, he tries squares as far as the third square in front of the pawn. If all of these fail, the strategy fails, which in this case means the position is lost.

Blac.kV. basic strategy of reaching K4 suggests an inferior move at his second turn before the correct one is suggested. Black's correct moves are made because EG recognizes that when white succeeds because he gets the opposition, there are special techniques black can use to prevent this. (The evaluation rules that involve the opposition are written to record the fact that it occurred and the move suggestion rules check for such a message. 7 his is the same kind of mechanism that is in the Causality Facility described in (Berliner, 1974), but here the causes of failure recognized and the counter-strategies tend more in the direction of being special cases.) Black tries to just mark time for one move by moving to a square adjacent to the one he was on when white succeeded by obtaining the opposition. In this example he happens to make the correct move on the first try. In some other examples ECs behavior may be somewhat less appropriate.

More search control issues

Repetition of positions

The reader may have noticed that several positions are repeated. It is necessary to allow this; the reason is that in different contexts different variations may be tried from the same position. One component of the context is a statement of the strategy being pursued. Another is a He ^script ion of events, such as the occurrence of the pattern of the opposition, for which there are particular counter measures. In particular, the same position may be reached during the pursuit of two different strategies. That is exactly what happened in this case. Nodes in the search tree can be considered the same if the positions are the same in the usual sense and the contexts are also the same.

Searching in context

If a strategy fails somewhere other than the root node of the search, EG will usually try fewer strategies than it would if that position were a root node. Typically, the strategies in simple king and pawn positions have the property that if the strategy generates any moves at all, one of them will be at least as good for its goal as the best move- generated by any strategy not generating moves diiected to that end. This implies that if a strategy is tried at a node, it does not need to be tried anywhere in the search trees rooted at siblings of the nodes generated by the strategy. Eor this reason we calf it the exclusionary property. This method of reducing tree search is based on a suggestion in (Berliner, 1974). Other aspects of control of the tree search are also directly related to concepts described there.

In Example 3 both of white's basic strategies are suggested at the root node and they have the exclusionary property, so white does not initiate branching except at the root node. If the exclusion did not occur, white would try two moves at each of depths 2 and 3.

Strategies that fail

If in some position a strategy cannot make a consistent move or it is known that the value it was aiming for is not obtainable, we say that it has failed in that position. (Each strategy has associated information giving the minimum value it is designed to achieve if successful, effectively a standardized piece of goal information.) We make the assumption that the principal variation will end at a leaf at which a strategy has succeeded. Since the minimum value can be chosen to be the absolute minimum value, it is easy enough to choose the value conservatively, and some cutoffs can be made this way.

When a strategy fails, it may do so at a position where the static evaluator is uncertain of the value of the position. If some other strategy is not a failure at the node, the failure of the strategy is hardly more significant than, say, suggestion of an illegal move. If all proposed strategies fail, the "success" assumption affects the value

backed up. Under that assumption, the variation could be discarded, and as a practical matter that would usually be equivalent to giving the position the absolute minimum value. In practice, we subordinate this assumption to the assumption that the static evaluation is consistent. Instead of returning the absolute minimum value, we return the minimum value indicated by the static evaluation. This provides a consistency check in those cases where the principal variation is generated by a strategy that fails.

As has been mentioned above, with the "interruption" scheme sometimes positions are played correctly only because one strategy suggests the first moves of a variation, at which point another strategy suggests the best continuation. It is hard to ensure that the interruption will occur if the interrupting strategy has the exclusionary property, though under some circumstances it will.

Kin£ and two pawns versus kin£

The main problem in endings with king and two pawns against a king is to use the information about the pawns individually along with the information applying only to the two pawns together. It has turned out to be fairly simple to use the existing information about endings with a king and pawn against a king.

The problem of evaluation of positions was handled by generalizing the evaluator for king and pawn versus king endings a little bit. It was given parameters and expanded somewhat. One typical (and important) change was that those positions leading to stalemate with only one pawn on the board almost always lead to wins with two pawns. The rules for detecting these classes of positions were modified to distinguish between positions with and without an extra pawn which can make a move for the simple purpose of forcing black to move. The evaluator for positions with a king and two pawns versus king has a rule which says with minor exceptions that if a position can be recognized as a win for either pawn "by itself" it is a win with the two pawns together. Suggestion of strategies and moves is handled in a similar way. EG solves almost all endings with king and two pawns versus king. The known exceptions involve potential stalemates and a few unusual positions. There seems to be a tendency for a few unusual kinds of positions to be overlooked when rules are being written. •

Unfortunately, it does not appear that things can be this easy in general. To be used in more complex positions, knowledge sources like the position evaluators would have to be able to give much more information, certainly things such as reasonable bounds on the time needed to win with a particular pawn and the squares each side needs to have available to it.

Calculations

EG's ability to detect properties of positions without searching is distinctly better than has been indicated up to now. It has the ability in many cases to detect that patterns can be achieved without going through a search to reach a position where the pattern actually applies. The calculations are based on time and distance.

For example EG assumes in king and pawn against king situations that if the white king is "as close" to the pawn as the black king and "closer" to one of the three squares marked with stars in Example 3, white can win the ending. Conversely if the black king is "closer" to the pawn than the white king he will be able to capture the pawn and draw.

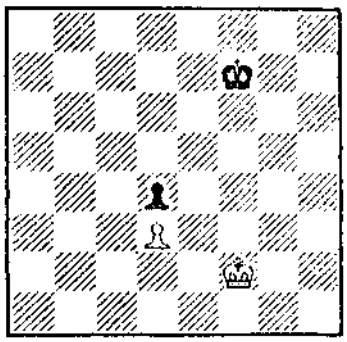
Two algorithms have been developed which handle time and distance measurements with great precision within their domains of applicability. They both operate by marking squares of the chess board with indications of distance. They each do their analysis with all other pieces stationary by assumption.

One is a variation of the A* algorithm for finding minimum cost paths, described in (Nilsson, 1971). The variation is that this algorithm finds ALL minimum cost (shortest) paths for a king of a given color between two given points on a board rather than the one path given by A*. The other algorithm is less interesting from *an* algorithmic point of view," but more interesting in the assumptions behind its use. Its purpose is to determine which squares of the board the white king can reach before the black king and which squares the black king can reach first. The idea is that each side starts out claiming the square that its king is on. The sides then alternate in claiming all the unclaimed squares adjacent to squares already claimed by them and not adjacent to any square already claimed by the other side. The algorithm terminates when neither side can claim any more squares.

This divides the board up into "spheres of influence" of the kings and it provides a very useful approximation for the set of squares which each king can reach despite the efforts of the other. Certainly each king can reach all of the squares indicated by this algorithm. However, if one of the kings only needs to reach one of a set of two or more squares, he may be able to do so even when the algorithm does not indicate it. This analysis is quite effective and it solves very much the same set of problems that people Solve by "counting" analysis on the chessboard. See [f]Botvinnik, 1970) for a proposal to apply this type of analysis to the movements of all types of pieces for analysis purposes.

Example 4 is one where EG's calculation abilities can be used and are, but some searching is still done. If the black king were at N2, black would not try the first variation. If the white king were at N2, white would not try the second variation shown.

Example 4:



Black to move

1. ... K-K3 2. K-B3 K-Q4 3. K-B4 K-B4
4. K-K4 [draw win]
1. ... K-B3 2. K-K2 K-N4 3. K-Q2 K-N5
4. K-B2 K-N6 5. K-N3 K-B6 6. K-B4 [lose]
2. K-B3 K-B4 [draw]

The values given are all relative to white. The initial position is evaluated as [draw win], at least a draw for white. Black abandons the first variation upon realizing it cannot succeed at its aim of achieving a "loss". White's strategy of attacking the black pawn from the other side is fairly reasonable, but fails. The last variation, which black uses to select his move, is good play for both sides.

Feasibility and interruption

In many positions the calculations can also determine when simple strategies will fail. We call this the use of calculations for feasibility testing. Feasibility testing is an important means for reduction of search in the endings we have seen. In the design of the program we found that feasibility testing interacts badly with careless reliance on interruption of strategies by higher-ranked strategies.

The problem is that when a strategy may be interrupted in unspecified ways by unspecified other strategies, it is impossible to reject in advance any moves it might generate. This is the problem referred to in the description of Example 3.

Since feasibility testing is so useful, the use of interruptions has been modified to make allowed interruptions explicit in the representation of each strategy. This has required better understanding of the situations where we had relied on interruption of strategies before.

The fact that we need to eliminate arbitrary interruptions does not mean eliminating searching. It only means that arbitrary interruptions cannot be implicitly allowed in all strategies. It is not necessary in all strategies. To solve more difficult problems with limited search will require the ability to automatically generate strategies in which limited and specified types of interruptions are allowed.

More representation issues

A view of the kpk endings

We now turn to a large view of the king and pawn versus king endings. EG's behavior in playing out an ending can he viewed through an analogy to navigation, f G has a pretty good idea of which way to go at any given point, but is still subject to error. To avoid error, it has memorized a number of landmarks in the area, and it corrects itself by watching for them. That is, it backtracks when a clearly wrong landmark is reached and stops searching when a satisfactory one is found. (The analogy works best when EG is playing the side which succeeds in the ending.) A "landmark" is a position meeting the conditions for one of the static evaluation rules. Figure 1 is a "map" which portrays something of how EG interacts with its environment in the king and pawn versus king endings.

Each node in the graph represents an informally defined (lass of positions, each associated with a few related types of landmarks, i.e. known patterns or types of positions. We give either a verbal description of the main kind of pattern or a generalized description of the set of patterns associated with each node. (There are really a number of additional types of known positions which cover special cases, especially where the basic rules would give incorrect information. These are left out to keep the figure simple.) EG may start a search from a position not belonging to any node and search may pass through positions belonging to no node while navigating from one node to another. The nodes also mostly have self-loops which are not shown.

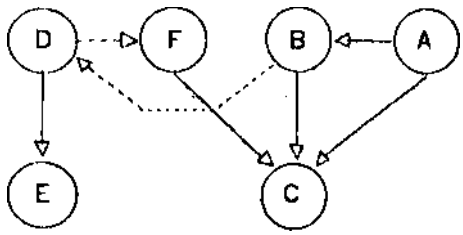


Figure 1: States of King and Pawn vs King Endings

Descriptions of nodes:

A: Positions where white has the win and the white king can deny the black king access to the squares of the file of the pawn which are in front of the pawn.

B: Positions where white has the win but black has access to the file of the pawn.

C: Positions where the pawn has been safely promoted.

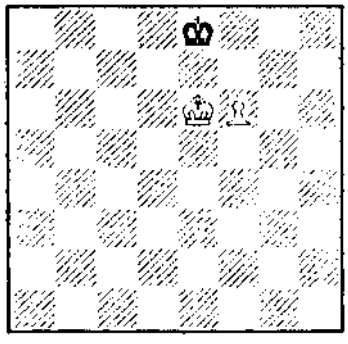
D: Positions where the pawn is safe, but no win is

possible with correct play by both sides. (See Example 6.)

E: Stalemated positions.

F: Special case positions with the white King on the sixth rank, as in Example 5.

Example 5:



Black to move

The clashed arrows represent transitions corresponding to blunders a tyro might make because of unfamiliarity with these endings. He might also fail to get the most out of a position not belonging to a node.

The existence of this graph model of the king and pawn versus king ending problem was not assumed during the development of the program. The graph does not coi respond directly to any structure in the program. It does summarize some things nicely.

Making progress

For example, notice that the search does not terminate only at positions known to be better than the initial position. A variation is considered satisfactory if it leads to a position as good as can be expected to result from the root node. It would appear that EG should have trouble with "endless" looping. It could loop among positions in a node or among nodes. Mow does it know how to make progress? The answer at present is that the search is sufficiently well-directed that the issue doesn't occur. This has been somewhat surprising to us, and we have considered ways of giving EG "a better sense of direction". More on this later.

Putting up resistance

Game-playing programs sometimes exhibit quirks of behavior when choosing between moves of nearly equal value. Behavior may look particularly strange when a program is choosing between bad alternatives where the loss comes immediately with one and sometime later with the other. As described so far EG would choose arbitrarily between any two variations leading to equal degrees of failure for it. Where one way of failing is complex and the other is immediate and simple, the result may look foolish. Tor example, in some king and pawn versus king positions white might unnecessarily abandon the pawn. If the position weren't winnable, white wouldn't care whether he had the pawn or not.

To try to prevent this problem, one might add some slight credit to failing variations where the failure is recognized deeper in the tree or create a measure of the complexity of the search that must be performed to determine that the variation fails. We have responded by distinguishing individual positions from one another, rather than looking at search depth or complexity. This is convenient for us because we can use the evaluation rules that already exist. We have been able to express our subjective preferences successfully by this method.

In the king and pawn versus king endings we differentiate positions where black can force white to show a little understanding of the ending by using the opposition properly and those where the question will not come up. We also distinguish between the utterly dead draws and positions where white retains his pawn and can force black to play correctly for the stalemate.

Making these fine distinctions can lead to problems with searching. The search looks for "the best move", and this can lead to substantial amounts of searching to decide which of two moves is slightly better than the other in a situation where a truly better move has yet to be tried.

To resolve this problem we run the search in two phases. In the first phase the search only concerns itself with the major issues of winning and losing. If after the first phase an issue of putting up resistance remains, the search is extended (redone in fact) to decide that finer point.

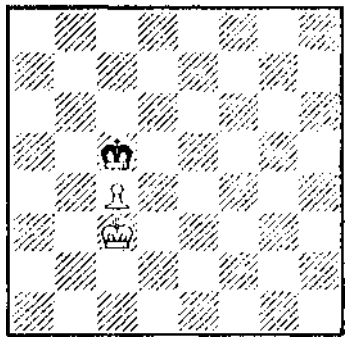
A paradox

There are situations in chess where making progress in a winnable position is a significant problem, as in the ending of bishop and knight against a lone king. A program has been written which uses search to resolve those problems (Hueberman, 1968). It deals strictly with situations which are already known wins for one side, so its representation is designed to keep track of progress. Its static evaluation yields conservative minimum measures of progress and the role of search is to select a move by the side with the advantage which definitely leads to progress despite resistance. Information ranking won positions according to progress is quite appropriate, so it appears we could encode this information on the existing

That ending is known to be winnable, yet it is thought that the more unfavorable positions require over 30 moves for the mate. See Averbakh and Maizelis, 1974.

numeric preference scale much as is done with the "resistance" information. But in a sense we cannot do this. It is interesting to see that the preferences of black and white cannot necessarily be represented on a single linear scale. Consider the positions in class F such as Example 6:

Example 6:



either side to move

How does each side compare the value of this situation with the same position one or two squares further advanced, e.g. pawn at QB5? We believe that white prefers the position with the more advanced pawn if he has *any* hope that black may blunder. Black's opinion may not be quite as clear. Let us suppose that his preferences are expressed on the same scale as white's. If this is true he must prefer the less advanced position.

Thus if two strategies are tried, both maintain the draw, *and* one ends with the pawn less advanced than in the other, black will choose the one with the less advanced pawn. We see no reason why black should behave this way. If he has confidence in his understanding of the situation he has nothing to gain by delaying -- he will lengthen the time it takes him to reach his objective with no other effect. We feel that black at least must not prefer to slow down the game in such a situation, so his preferences are not the inverse of white's.

The situation makes sense if the players have models of each other. We are assuming that white has assigned black a nonzero probability of failing to find the stalemate. Black, we are assuming, does not do so.

The form of rules in EG

The form taken by rules of EG has proved a success. Calls on complicated functions are allowed, notably those described in the section on calculations, so it would be difficult to determine in any definite way the absolute power of the rules. Nevertheless, we can talk about the power of the rules relative to the functions they call. The rules have proved generally comfortable to use and powerful enough for our purposes.

The rules are interpreted directly by LISP, but we restrict ourselves to a subset of the control structure and almost bar side effects entirely. Side effects are limited to use for the purpose of user-specified elimination of common subexpressions. The control structure consists of conditionals, mapping functions, find quantification over lists.

The mapping functions are the standard MAPCAR and MAPCAN functions. Treating lists as sets, they allow the calculation of images of sets *under* transformations, unions over sels of sets, and selection of subsets containing elements with specified properties. The existential quantifier is extended to indicate an element of the list having the property if such an element exists. There is also a universal quantifier. We do sometimes sort elements of a list and pick the best with some property, but this could be simulated with the other operations, although inefficiently. We rarely do violate the rules.

These operations with the booleans and "primitive" functions (and predicates) effectively give us the basic operations of set theory for finite sets, implemented as lists. This seems to be a very natural form in which to stale rules and create lists of things like strategies, and it is powerful enough to be used.

References

Averbakh, Y., and Maizelis, I., Pawn Endmgs, Chess Digest, Inc., 1974.

Berliner, H. J., Chess as Problem Solving, Doctoral Thesis, Carnegie-Mellon University, 1974.

Botvinnik, M., Ccimj^yJerSj Chess, arid Long Range Planning, Springer-Verlag, 1970.

Hueberman, B., "A Program to Play Chess Endgames", Technical Report CS-106, August 1968, Stanford University, 1968.

Newborn, M., "PEASANT: An Endgame Program for Kings and Pawns", in P. Frey, ed., Chrjss Skill in Man and M^hine, Springer-Verlag, 1977.

Nilsson, N. J., Problem-Solving Methods in Artificial LntcilJRence, McGraw-Hill, 1971.

Pople, H. E., , "A Goal-Oriented Language for the Computer", in Representation and Meaning, H. A. Simon and L. Siklossy eds., Prentice-Hall, Inc., 1972.

[J] For a much more ambitious implementation of a related idea, see (Pople, 1972).