ISG 1977-2

11

Inductive Learning in a Hierarchical Model for Representing Knowledge in Chess End Games

Pericles Negri

Department of Computer Science
University of Illinois at Urbana-Champaign

INTRODUCTION

The end game is the part of the chess game for which, in contrast with the other parts of the game, a good deal of knowledge is available that permits the player to find the correct move, if it exists. The problem then is to define precisely what this knowledge is, how it is organized and how it could be implemented in a computer so that the machine can play the end games efficiently without recurring to brute force searches.

We present here a model for this knowledge that is based on the concept of goals and subgoals. The primary goal of either White or Black is to win or to draw the game. This fundamental goal can be expressed as a logical function of some position predicates that represent geometrical or logical relationships among the men on the board, and some simpler subgoals that can be of offensive or defensive nature. In their turn these subgoals can be expressed as a function of their own subgoals and so on until very simple subgoals are reached that can be expressed exclusively as a function of primitive position predicates. To play the game in a determined position, the player tries to achieve some combinations of the subgoals of the primary goal determined by the logical function. This is done by trying to achieve simpler subgoals and so on through the very simple subgoals of the bottom of the structure. Whenever the last ones fail to be achieved, as characterized by the position predicates, the player goes back in a bottom-up direction and tries to achieve alternative subgoals.

Attached to the functions that define the primary goal in terms of its subgoals, there are special procedures that generate the move in case some combinations of these subgoals can be achieved.

Finally, it should be emphasized that the depth of the structure is quite arbitrary and different players will likely have different models for the same end game. Experienced players have probably less deep structures than unexperienced players because they eliminate some intermediate subgoals (what makes the functions more complex).

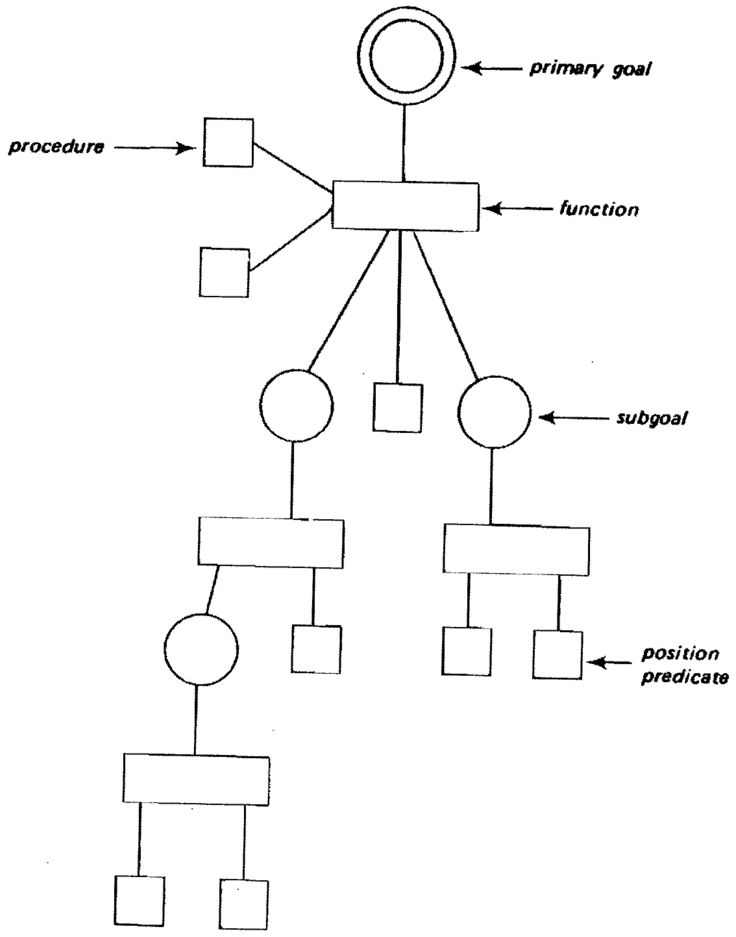

FIG. 1. A schematic representation of the knowledge in a hypothetical chess end game

Figure 1 is a schematic representation of the structure proposed but it should be noticed that the function blocks hide the fact that when a goal can be achieved through different combinations of its subgoals, some combinations will be more efficient than others, which implies a priority structure within those blocks.

LEARNING IN THE HIERARCHICAL STRUCTURE

Description of the model: single-pawn game case

In order to implement the proposed structure for chess end games in a computer we must be able to specify the subgoals, the position predicates and the functions that express a subgoal in terms of its subgoals and position predicates.

The latter could be a very difficult task unless some systematic way of doing it can be devised. Inductive learning through a variable-valued logic system like VL1 offers a solution for the problem. In order to test this, a partial model for the single-pawn end game (white king and pawn against black king) was constructed and VL1 formulas were obtained for two function blocks of the model.

To express a goal in terms of its subgoals and position predicates the following approach was used:

(a) Assign to each offensive subgoal a variable that can assume 3 values:

0-the subgoal cannot be achieved
1-the subgoal can be achieved
2-the subgoal was already achieved

(b) Assign to each defensive subgoal a variable that can assume 3 values:

0-the subgoal cannot be achieved (although it should)
1-the subgoal must be achieved (the opponent makes a threat).
2-the subgoal does not need to be achieved (the opponent is not making any threat)

(c) Assign to each position predicate a variable that assume as many values as the number of states of the predicate.

(d) Generate a learning sample of positions for the classes defined by each value of the variable that represents the goal for which the formulas are to be generated.

(e) Run the programs of the VL1 system (Larson, Michalski, 1975).

(f) Generate a testing sample of positions for the classes defined by each value of the variable for which the formulas were generated. Test the formulas obtained in the item (e) with the testing positions. Add the positions misclassified by the formulas to the learning sample and rerun the VL1 system programs. Keep doing this till all the testing positions are classified correctly by the formulas.

The partial model of the single-pawn end game consists of 4 submodels:

-Rook Pawn, Black's turn
-Rook Pawn, White's turn
-Non Rook Pawn, Black's turn
-Non Rook Pawn, White's turn

- *Rook pawn, Black's turn*

Primary Goal: To draw the game. It's a function of three subgoals and one position predicate (the pawn can run through the 8th row and get crowned).

1-to take the pawn
2-to enter the critical squares (figure 2). Once inside the critical squares the black king can keep inside them and avoid the pawn getting crowned.
3-to reach the critical position (figure 3). From the critical position the black king can either take the pawn or enter the critical squares.

The subgoal "to take the pawn" can be expressed in terms of simpler subgoals or directly in terms of position predicates. The other two can be expressed directly in terms of the position predicates.

- *Rook pawn, White's turn*

Primary Goal: To crown the pawn. It can be expressed in terms of a position predicate (the pawn can run) and two subgoals.

Subgoals of the Primary Goal:

1-to defend the critical squares.
2-to defend the pawn

These subgoals can be expressed directly in terms of position predicates.

- *Non-rook pawn, White's turn*

Primary Goal: To crown the pawn. It can be expressed as a function of one position predicate (the pawn can run) and four subgoals.

Subgoals of the Primary Goal:

1-to enter the critical squares of type 1 (figure 4).
2-to enter the critical squares of type 2 (figure 5) when the rank of the pawn is greater than 5.
3-to defend the pawn.
4-to enter the critical position (figure 6) when the rank of the pawn is equal to 7.

There is one exception to the subgoals 1 and 2 when the pawn is the knight pawn, it has rank equal to 6 and the black king is on the corner of the board. All these subgoals can be expressed directly in terms of position predicates.

- *Non-rook pawn, Black's turn*

Primary Goal: To draw the game. It can be expressed as a function of two position predicates (pawn can run, stalemate position) and five subgoals.

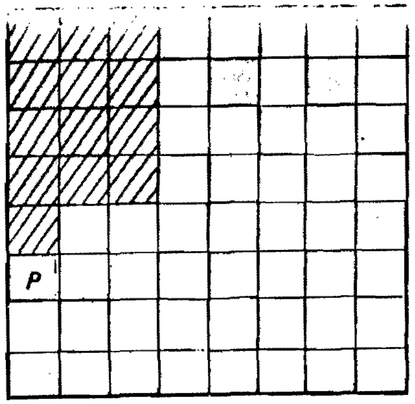

FIG. 2. Critical squares in the rook-pawn case.

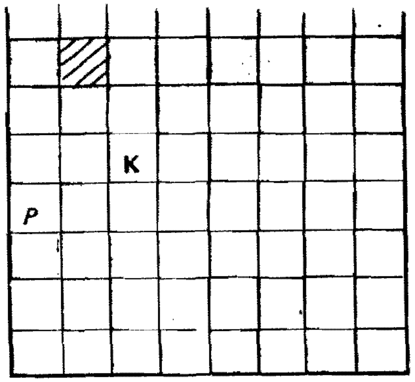

FIG. 3. Critical position in the rook-pawn case; the white king cannot be in the shadowed square

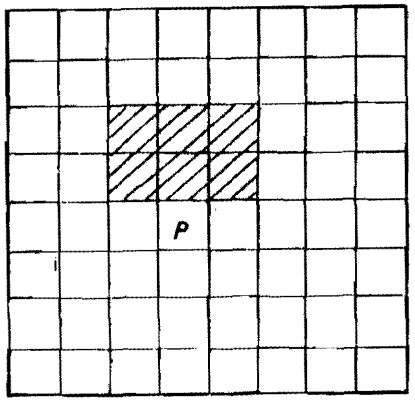

FIG. 4. Critical squares of type 1 for white king in the non-rook-pawn case

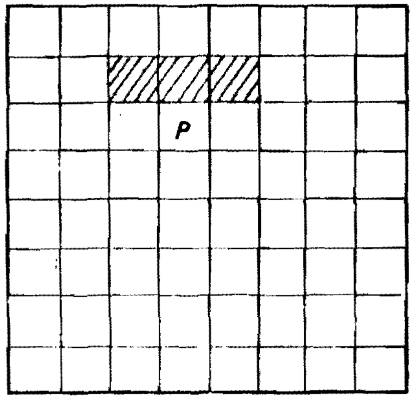

FIG. 5. Critical squares of type 2 for white king in the non-rook-pawn case

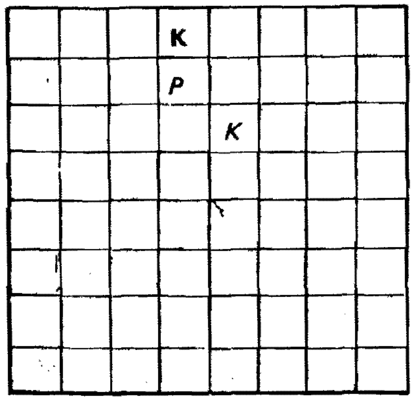

FIG. 6. Critical position for white in the non-rook-pawn case

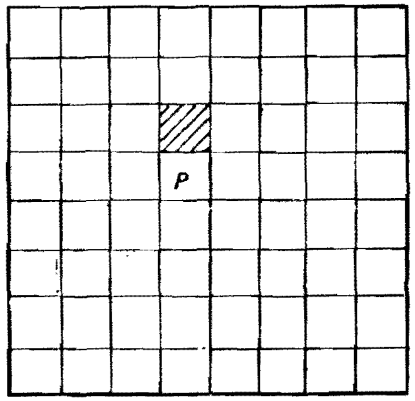

FIG. 7. Square 1 for the black king in the non-rook-pawn case

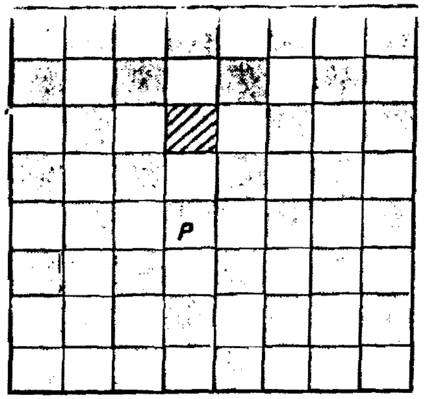

FIG. 8. Square 2 for the black king in the non-rook-pawn case

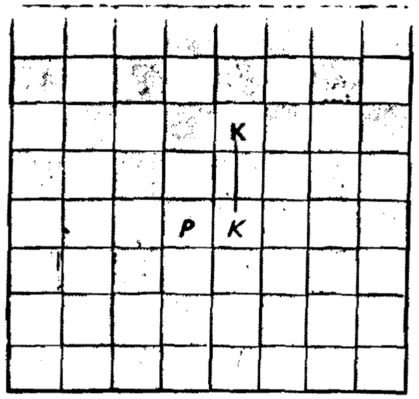

FIG. 9. Opposition of type 1

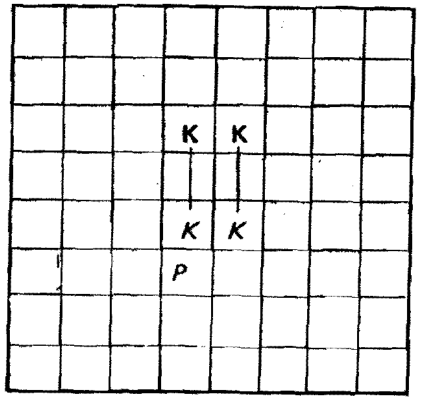

FIG. 10. Opposition of type 2

Subgoals of the Primary Goal:

1-to take the pawn
2-to reach square 1 (figure 7).
3-to reach square 2 (figure 8) when the rank of the pawn is less than 6.
4-to enter in opposition of type 1 (figure 9).
5-to enter in opposition of type 2 (figure 10) when the rank of the pawn is less than 5.

These subgoals can be expressed in terms of position predicates.

Learning experiments

First experiment

The first experiment was to generate the formulas that define the primary

goal of the black (rook pawn case) as a function of its subgoals and position predicate.

Variables used:

X1: 0-the pawn cannot run
1-the pawn can run

X2: 0-black king cannot enter the critical squares.
1-black king can enter the critical squares.
2-black king already in the critical squares.

X3: 0-black king cannot take the pawn.
1-black king can take the pawn.

X4: 0-black king cannot reach the critical position.
1-black king can enter the critical position.
2-black king already in the critical position.

The formulas obtained have the following terms:

Class 1: Black can draw the game

$(X1=0)\,(X4=1,2)\vee$
$(X3=1)\vee$
$(X1=0)\,(X2=1,2)$

Class 2: Black cannot draw the game

$(X1=1)\vee$
$(X2=0)\,(X3=0)\,(X4=0)$

This experiment was very simple and it was done only to illustrate the learning process for a high level of the structure. The learning positions were chosen in order to cover the whole event space. This could be done only because the number of variables is very small as well as their cardinalities. The formulas were obtained in one step of the learning process and as they are self evident they were not tested.

Second experiment

The second experiment was done to illustrate the learning process in a low level of the structure in which a subgoal is expressed exclusively as a function of position predicates. The formulas were obtained for the subgoal "black king can take pawn" in the rook pawn case.

Variables used:

X1: 0-black king in the region A of the board. (figure 11).
1- " " " " " B " " "
2- " " " " " C " " "
3- " " " " " D " " "
4- " " " " " E " " "

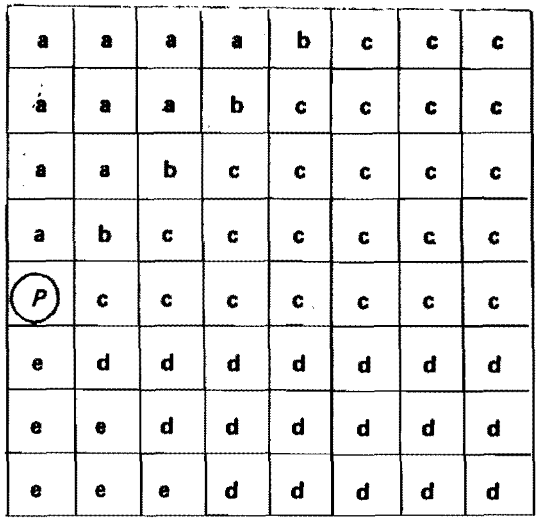

FIG. 11. Division of the board into regions according to the pawn position

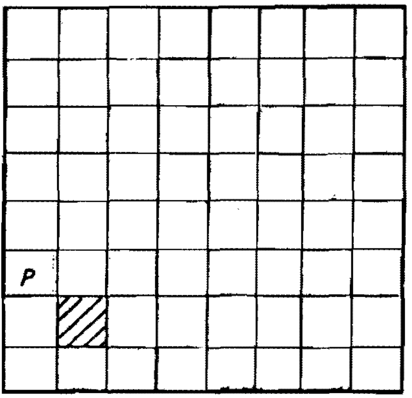

FIG. 12. Square *a*

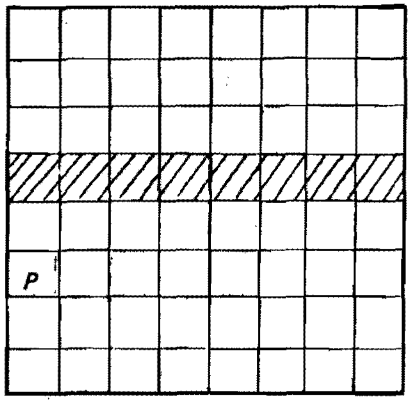

FIG. 13. Row *a*

X2: 0-white king in the region A of the board.
1- " " " " " B " " "
2- " " " " " C " " "
3- " " " " " D " " "
4- " " " " " E " " "

X3: 0-distance black king-pawn - distance white king-pawn < -2
1- " " " " " " " " = -1
2- " " " " " " " " = -1
3- " " " " " " " " = 0
4- " " " " " " " " = 1
5- " " " " " " " " > 1

X4. 0-distance black king pawn - distance white king pawn < -1
1- " " " " " " " " " = -1
2- " " " " " " " " " = 0
3- " " " " " " " " " = 1
4- " " " " " " " " " > 1
See figure 12 for the square *a*.

X5: 0-distance black king-pawn's file - distance white king-pawn's file < -1

X5: 0-distance black king-pawn's file - distance white king-pawn's file < -1
1- " " " " " " " " " " = -1
2- " " " " " " " " " " = 0
3- " " " " " " " " " " = 1
4- " " " " " " " " " " > 1

X6: 0-distance black king-pawn's rank - distance white king-pawn's rank < -1
1- " " " " " " " " " " = -1
2- " " " " " " " " " " = 0
3- " " " " " " " " " " = 1
4- " " " " " " " " " " = 1

X7: 0-distance black king-row *a* - distance white king-row *a* < -1
1- " " " " " " " " " " = -1
2- " " " " " " " " " " = 0
3- " " " " " " " " " " = 1
4- " " " " " " " " " " > 1
See figure 13 for row *a*.

X8: 0-pawn cannot run
1-pawn can run

X9: 0-pawn's rank = 2
1-pawn's rank = 2

X10: 0-black king cannot take the pawn in the present move
1-black king can take the pawn in the present move

The formulas obtained have the following terms:

Class 1: The black king can take the pawn

(X1=0) (X3=0:3) (X4=0) (X8=0) (X9=1) v
(X4=0:2) (X5=0) (X8=0) v
(X3=0:3) (X6=0,1) (X7=0) (X8=0) (X9=1) v
(X1=2:4) (X3=0:2) (X6=0:3) (X8=0) (X9=1) v

(X1=2:4) (X3=3,4) (X5=3) (X6=0,1) (X7=0:3) (X8=0) v
(X1=2,3) (X2=1:4) (X3=0:3) (X4=0:2) (X8=0) (X9=1) v
(X3=0,1) (X5=2:4) (X6=0) (X8=0) v
(X1=2) (X4=0,1) (X5=0:3) (X7=0,1) v
(X3=1,2) (X6=0) (X7=3,4) (X8=0) v
(X3=3) (X4=2) (X7=0,1) v
(X1=0) (X3=2)

Class 2: The black king cannot take the pawn

(X3=3:5) (X5=1:4) (X6=2:4) (X9=1) (X10=0) v
(X1=0:2) (X2=0:2) (X4=3,4) (X5=0:2) (X9=1) v
(X1=1:4) (X2=0,1) (X3=2:5) (X6=1:4) (X10=0) v
(X1=2) (X4=1:4) (X5=3,4) (X6=0) (X7=3,4) (X9=1) v
(X1:4) (X4=1:4) (X8=1) v
(X1=0,1) (X4=1:4) (X5=1:4) (X6=2:4) v
(X1=1:4) (X3=4,5) (X5=4) (X7=0:3) v
(X1=3,4) (X2=0) (X5=3) (X7=1:4) (X10=0) v
(X1=0:3) (X6=0) (X7=2) (X9=0) v
(X1=2:4) (X3=2) (X5=4) (X7=0) v
(X1=1:4) (X3=1:5) (X6=4) (X9=0) v
(X4=3,4) (X5=0,1).
(X8=1)

Initial samples of 120 learning positions and 120 testing positions were used. After three steps of adjusting the formulas by adding to the learning sample the testing positions that were misclassified we got all the testing positions correctly classified by the last formulas. The learning positions were generated by dividing the positions in 25 classes defined by the 25 possible combinations of the values of the variables X1 and X2 and picking from each class some positions according to a 'near miss' criterion (Michalski, 1975) in which positions which are near to a 'critical line' that separates the class 1 from class 2 are taken. The testing positions were generated by taking some general positions from the 25 classes defined before.

CONCLUSIONS AND SUGGESTIONS FOR FURTHER RESEARCH

In a previous work (Michalski and Negri, this volume) we tried to introduce inductive learning in chess end games by generating formulas for two classes of positions of the single-pawn end game: winning positions (for White) and drawing positions (for Black). The variables used were position predicates used by Tan (Tan, 1972) in his model for this end game and other variables that represented the positions of the men on the board. The results although not totally unsatisfactory showed the deficiencies of the model used: the formulas obtained were heavily dependent on the second type of variable. This indicates that the variables of the first type were too specific to some groups of positions so the

"preference" for the second type of variables that are more general. What happened is that we clustered in two classes positions that had quite different characteristics. To get simple formulas for these classes we would have to use a large number of variables to cover all those characteristics, which is not practical.

In the model that we propose in this work, we use the learning process to generate formulas for some reduced classes of positions, namely those associated with some subgoals of the structure. This permits us to use exclusively variables that are relevant for those classes of positions in particular, which makes the learning process simpler and more efficient.

We think that the hierarchical structure defined by our model is very close to the way people play chess end games. The procedures attached to the structure to generate moves can be very simple because they generate moves that have to reach some determined goal. A simple search process can be efficiently used because we can eliminate *a priori* from the search moves that will not lead to the desired goal. This seems to be the kind of process involved in human thinking that automatically rejects irrelevant moves. Besides that, information obtained in the lower levels of the structure can be used by those procedures.

We did not try to develop a playing model for the case in which the desired goals cannot be achieved, i.e., the position is a losing one. But we think it could be done because, at least we can know from the formulas the "weaknesses" of the opponent and try to set traps for him.

The next research step should be the development of automatic ways of generating the position predicates that are used in the bottom of the structure for defining the simple subgoals. This is boring work and it would be nice if it could be automated. A possibility could be to extend to chess the ideas developed by Newman and Uhr (Newman and Uhr, 1965) for board games like Go Moku and Tic Tac Toe. Their method consists in defining a utility index for the patterns that appear on the board during the game. This can be defined simply as the number of times the pattern appeared in winning positions over the number of times the pattern appeared at all. Patterns that have roughly the same indexes are put in the same class. Now to the patterns that belong to some class with a high index of utility, geometric transformations like rotation, translation and reflexion are applied in order to transform several of these patterns into a stereotype pattern. The patterns that are considered equivalent under those transformations define a "predicate" of the positions. The problem in extending these ideas to chess is that as Pitrat observed (Pitrat, 1974), the logical relationships among the men on the board are not easily expressed by geometrical transformations. But if we use VL1 formulas to cover the positions with the same utility index against the other positions, the formulas so generated could define the desired predicates. In this case these formulas would use exclusively primitive variables like position descriptors for the men on the board. Finally it should be noticed that these formulas should use the full power of the VL1 language in order to include selectors of the kind $[X_i\text{-}X_j<K]$, where K is some constant. The generation of formulas for the variables used in the second

experiment described in this paper [illegible] a clear example for the necessity [illegible] this
kind of selector. Programs like Jensen's (1975) for synthesizing VL_1 formulas with symmetric selectors could be used for that purpose.

REFERENCES

Jensen, G.M. (1975) SYN-1: A Program that Detects Symmetry of Variable-Valued Logic Functions. Department of Computer Science, University of Illinois.

Larson, J. and Michalski, R.S. (1975) AQVAL/1 (AQ7): User's Guide and Program Description. Department of Computer Science, University of Illinois.

Michalski, R.S. (1973) AQVAL/1–Computer Implementation of a Variable-Valued Logic System and the Application to Pattern Recognition, *Proceedings of the First International Joint Conference on Pattern Recognition*, Washington, D.C.

Michalski, R.S. (1975) On the Selection of Representative Samples from Large Relational Tables for Inductive Inference. M.D.C. 1.1.9 Department of Information Engineering, University of Illinois at Chicago Circle.

Newman, C. and Uhr, L. (1965) BOGART: A Discovery and Induction Program for Games. *ACM 20th National Conference.*

Pitrat, J. (1974) Realization of a Program Learning to Find Combinations in Chess. *Computer Oriented Learning Processes.* NATO Advanced Study Institute, Bonas (gers) 1974.

Tan, S.T. (1972) Representation of knowledge for very simple endings in chess, *Memorandum MIP-R-98*, School of Artificial Intelligence, University of Edinburgh.