SP-2150

Report on a Mating Combinations Program

G. W. Baylor

4 June 1965

SP *a professional paper*

Report on a Mating Combinations Program

by

G. W. Baylor

4 June 1965

SYSTEM
DEVELOPMENT
CORPORATION
2500 COLORADO AVE.
SANTA MONICA
CALIFORNIA

ABSTRACT

This paper describes a computer program that analyzes chess positions that contain checkmating combinations. Over the range of positions the program can handle, the program is viewed as a psychological model of human problem-solving behavior. The model has a set of mechanisms for generating a small, highly selective set of moves for analysis and a search strategy for conducting the chess analysis. These parallel the human chess player's search behavior on several points: particularly on the quality of the moves considered and on the heuristics and stop-rules for keeping the ever-branching tree of move possibilities within manageable limits. Some specific hypotheses about the chessmaster's perceptual abilities are offered to account for the quality of the moves that come under consideration while some hypotheses about uncertainty reduction and the nature of the constraints imposed by immediate memory are suggested to account for some of the structural facets of the thought process. These derive from the detailed process comparison of a human's behavior with the model's. On a performance measure the program has discovered some of the most sparkling combinations in the chess literature, in positions requiring analyses ranging from two to eight moves in depth.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

PREFACE

This paper is directed to several audiences. Primarily, it is intended as a guide to a computer program of some 5000 IPL-V instructions. The hope is that together with the program (a copy of which can be obtained upon request) this guide will be sufficient to enable other investigators of the Royal Game to dig in and pursue the ideas embodied in the program.

I do not suppose, however, that many readers will be interested in such a fine level of detail. Psychologists of the computer simulation ilk may want to skip the sections on representation and organization altogether and head straight for the last two sections on heuristics and psychological interpretation. Investigators concerned solely with artificial intelligence may prefer to read selectively from all four of the sections on representation, organization, heuristics, and results. Chess players, especially those like U. S. Champion Bobby Fischer who thinks the whole conception a "pipe dream," should take heed.

ACKNOWLEDGMENTS

I am indebted to a host of friends and advisors in this undertaking: especially to my thesis committee in the Psychology Department at Carnegie Tech, consisting of Professors G. A. Forehand, B. F. Green, and H. A. Simon, chairman, and to H. Borko and R. F. Simmons, during a valuable summer at the System Development Corporation (SDC) in Santa Monica. All five advised well.

The ideas and insights of A. Newell, H. A. Simon and M. R. Quillian at Carnegie Tech and of A. D. de Groot at the University of Amsterdam so permeate this paper that it would surely never have been written without them.

The aid and cooperation of the systems men were indispensable. T. van Wormer at the University of Pittsburgh Computation Center and, during the summer of 1964, D. Londe and C. Weissman at SDC were always quick to give of their time and know-how.

E. Book of SDC and W. Teitelman of M.I.T. genially consented to participate in the chess experiment reported in the paper.

For suggesting many needed corrections in an earlier version of this paper I should like to thank, in addition to my thesis committee and mentors mentioned above, E. J. O'Connell, P. M. Wortman, and T. A. Standish; and Miss Patricia Flower at SDC for typing the final manuscript.

None of the above is to be held responsible for the errors and infelicities which remain, all of which, I am sure, are peculiarly my own.

I. INTRODUCTION[1]

The most successful chess machine to date (Kotok, 1962) is a coward. The program has only once administered checkmate and that was sheerly fortuitous: snatching a Pawn also happened to mate the opponent. The program reported here, while bloodthirsty, is not a complete chess player; it does not play games. Rather it is a chess analyst limited to searching for checkmating combinations in positions rife with tactical possibilities.

A combination in chess is a series of forcing moves with sacrifice that ends with an objective advantage for the active side (Botvinnik, 1960, pp. 266-67). A checkmating combination, then, is a combination in which that objective advantage is checkmate.[2] Thus the program described here--dubbed MATER--given a position, proceeds by generating that class of forcing moves that put the enemy King in check or threaten mate in one move, and then analyzing first those moves that appear most promising. The course of MATER's analysis--the moves it discovers and chooses to explore and the reasons why--is what makes it psychologically interesting. After all, in agreement with the former world champion: "the essence of a chess master's art ... fundamentally ... consists of the ability to analyse chess positions" (Ibid., p. 11).

The organization of this paper centers around MATER's "ability to analyse chess positions." After a brief look at the program's history in Section II, the programming language representation of the chessboard and chess pieces and the basic chess capabilities this affords are examined in some detail in Section III. Section IV treats the overall

---

[1]This paper was submitted to the Department of Psychology, Carnegie Institute of Technology, in partial fulfillment for the degree of Master of Science. This investigation was supported in part by the Public Health Service Research Grant MH 07722, from the National Institute of Mental Health.

[2]Sometimes the defender is able to avert the checkmate by incurring a heavy loss in material (pieces and/or Pawns). If the attacker's gain in material is indeed an "objective advantage"---the defender being left with no compensatory attacking chances--then such combinations would generally be called mating combinations, even though not ending in mate. The current version of the program confines itself to mating combinations in the narrow sense--those from which there is no escape. Inclusion of the broader class is an obvious extension.

organization of the program, which is designed to allow flexible movements in an analysis tree of possibilities. Then in Section V the "top level" comes under consideration: MATER's heuristics of analysis--the search rules and priorities, the search evaluators--that enable it to find a mate in the maze of possibilities. In Section VI the evidence for the program as a model of human problem-solving behavior is presented.

II. HISTORY OF THE PROGRAM

MATER has led a checkered life. The original mating combinations program, as conceived by Simon and Simon (1962), was a hand simulation setting forth a strategy of search. Briefly, the program (1) generated all checking moves for the attacker (listing them in a prescribed priority order, as will be discussed in Section V); (2) selected and tried the move with highest priority; (3) enumerated all the opponent's legal replies to it; and (4) selected a reply and played it. If no checkmate had yet been found, the cycle continued by (1) again generating all checking moves; etc. Once a mate was found, of course, alternative replies by the opponent had to be investigated: a checkmate must be demonstrable no matter what the opponent does. The program escaped some proliferations in the analysis tree by eliminating from consideration those checking moves to which the opponent had more than four legal replies.

According to hand simulations the program discovered mating combinations in 52 of the 129 positions collected in Fine's (1952) chapter on the mating attack. The mates in these 52 positions were all forced sequences of checking moves. But a hand simulation is not a rigorous model and, as such, is itself sometimes prone to the imprecisions and ambiguities of many verbal theories. Indeed, Myron and May (1963) pointed out two such ambiguities in the specifications laid down by Simon and Simon.

Newell and Prasad (1963) set out to remedy the situation. They coded an IPL-V program which set up a chessboard, recorded positions, made moves, tested their legality, and performed a few other functions (see Section III). This they overlaid with the beginnings of a mating program.

The principal organization for the first version of a mating program--MATER I--began in the fall of 1963 when H. A. and P. A. Simon set out to implement on the computer their earlier specifications for a hand simulation. The author of this paper worked on that implementation with them and, during the summer of 1964, wrote a different executive structure into a second version of the program--MATER II. The designs and results of both these efforts are reported here.

III. THE BASIC REPRESENTATION

Two interrelated questions guided the choice of representation:

(1) What are the necessary components of a chess representation?

(2) How should this information be organized?

The first question is largely task specific, the second a more general psychological and programming consideration.

In response to (1): The program should be able to "see" the same things a human sees when he looks at a chessboard. Thus the program requires an internal representation of the squares and pieces on a chessboard and the relations among them, and a set of processes that can pick off and make use of these relations as needed. The former requirement is called "setting up the chessboard," the latter "move-making and board processing capabilities."

In response to (2): The game of chess provides an inhomogeneous collection of information out of which moves must be forged. Thus there must be enough variety in the representation to discriminate all the different kinds of moves; given that, the information should be stored in such a way that little space is allotted to moves that seldom occur (such as Pawn promotions, castling, etc.), and the dependence and division of information between routines and data should remain flexible and open to change and never solidify into a resistant collection of conventions (Newell, 1962). List processing languages such as IPL-V (Newell, 1961) are specifically designed to cope with such problems. Lists permit an efficient allocation of space, and description lists provide an associative memory with easy accessibility to the descriptive information stored there.

A chessboard is made up of squares, which lodge pieces, which make moves from one square to another. Objects in chess, like the 64 squares and 32 men, can be represented as symbols on lists, and moves can be represented as names of description lists with certain prescribed associations (such as the square <u>from</u> which a piece comes and the square <u>to</u> which it moves, and the kind of <u>move</u> in question). A chess position, <u>mo</u>reover, can be fully described as a list of pieces and squares and a chess game as a list of moves that originate from a standardized initial position and terminate in a well-defined checkmate position.

With this introduction the Newell and Prasad representation, which has been slightly revised to fit it to a mating program, can be considered in detail.

SETTING UP THE CHESSBOARD

A chessboard is made up of eight ranks and eight files which rule off 64 squares. The sequence of symbols S1...S64 is used to denote these 64 squares, as depicted in Figure 1.

Ranks ↑								
R8	M25/S57	M26/S58	M27/S59	M28/S60	M29/S61	M30/S62	M31/S63	M32/S64
R7	M17/S49	M18/S50	M19/S51	M20/S52	M21/S53	M22/S54	M23/S55	M24/S56
R6	S41	S42	S43	S44	S45	S46	S47	S48
R5	S33	S34	S35	S36	S37	S38	S39	S40
R4	S25	S26	S27	S28	S29	S30	S31	S32
R3	S17	S18	S19	S20	S21	S22	S23	S24
R2	M9/S9	M10/S10	M11/S11	M12/S12	M13/S13	M14/S14	M15/S15	M16/S16
R1	M1/S1	M2/S2	M3/S3	M4/S4	M5/S5	M6/S6	M7/S7	M8/S8
	F1	F2	F3	F4	F5	F6	F7	F8

Files →

Figure 1. The Naming Scheme for the Chessboard (reprinted with permission from Newell and Prasad, 1963, p. 16)

In the data section of the program there are a list of ranks, L1, containing members R1 through R8, and a list of files, L2, containing members F1 through F8. Each rank is itself the name of a list containing eight member squares; e.g., R1 contains S1, S2,....., S8.

In the routines section of the program there is a superroutine, E1, which sets up a chessboard; it calls nine routines, E2-E7, E9, E10, and E12, which do the work. Routine E2 creates the relations in the file (vertical) direction, the same relations that are given in the data section to the ranks. That is, E2 builds the eight file lists, F1 through F8, out of the rank lists, R1 through R8. For example, the list F1 is erected out of member squares S1, S9, S17, S25, S33, S41, S49, S57, as in Figure 1. Then routine E12 takes each of the 64 squares and assigns it rank and file (x,y) coordinates, which are later used to compute another set of relations among squares.

Squares

For each square on a chessboard it is essential to know:

(1) the name of its occupant, if there is one, and

(2) the name of all its neighboring squares in the chess-legal directions.

The first desideratum is effected by defining an attribute MO, "Man on Square?," on the description list of every square and assigning as its value the name of the piece occupying it--if there is one.

The extensive network of relations among squares, constituting all legal move directions in chess, is captured by defining 16 directions on the chessboard, as in Figure 2.

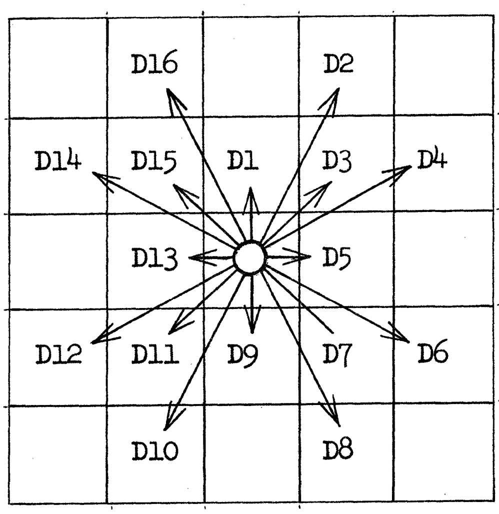


Figure 2. Sixteen Chess Directions. The reference square is the one with the circle in the middle. (reprinted with permission from Newell and Prasad, 1963, p. 16)

The even numbers (D2, D4,....., D16) define the eight possible Knight move directions; half the odd numbers (D1, D5, D9, D13) define rank (horizontal) and file (vertical) directions, the other half (D3, D7, D11, D15), diagonal directions.

All of these directions can be broken down into horizontal and vertical component directions, i.e., redefined in terms of D1, D5, D9, and D13. In the data structure, lists D2, D3, D4, D6, D7, and D8 are so defined; list D2, for example, is composed of members D5, D1, D1--the component directions for that particular Knight jump in the NNE direction. L6 names the list of these six "decomposable" directions, and L5 makes allowance for the other half by listing the inverse or opposite directions of D1 through D8; thus, L5 contains, in pairs, D1, D9, D2, D10,..., D8, D16.

Routines E3, E4, E5, and E8 use lists L1, L2, L5, and L6 to build the network of relations among squares by assigning to each of the 64 squares all surrounding squares as values of each of the 16 directions that obtain. In the initial position, for example, the list structure of the square S8--White's KR1 in standard American chess notation--would look like this:

Name	Symbol	Comments
S8	9-0	
9-0	0	
	M0	(Attribute: "Man on Square?")
	M8	(Value: M8, the White King Rook)
	D1	(Attribute: "Square to the North?")
	S16	(Value: S16, the square KR2)
	D13	(Attribute: "Square to the West?")
	S7	(Value: S7, the square KN1)
	D14	(etc.)
	S14	
	D15	
	S15	
	D16	
	S23	0

Note that in such a list structure representation only those five of the 16 legal directions that are needed are defined; space in memory is not consumed by providing the "information" that the other 11 directions have *no* values, as it would seem to be in a matrix representation of this kind of data. (Cf. the argument in Newell, 1962, pp. 411-12 for a fuller statement of this view.)

Men

The 32 men take and retain their designations from their placement in the initial position; they are denoted by the sequence of symbols M1...M32, as in Figure 1.

For each piece, it is essential to know:

(1) the square he occupies,

(2) his type (attribute A1),

(3) his color or side (attribute A2),

(4) his legally permissible move directions (attribute A20), and

(5) his legally permissible capture directions (attribute A21).

The first of these is effected by defining an attribute SO, "Square on?," on the description list of every piece and assigning as its value the name of the square the piece currently occupies.

The complete range of values which the other four attributes may assume is laid out in Table 1; note the contingency of the values of attributes A20 and A21 on A1 and A2.

Table 1

Table of Values for the Four Invariant Attributes of Pieces

Attributes ⟶		A2 (Color)	A20 (Move directions)	A21 (Capture directions)
↓	K1 (Pawn)	K10	D1	L13
		K11	D9	L14
	K2 (Knight)	K10, K11	K22	K22
	K3 (Bishop)	K10, K11	K21	K21
A1 (Type)	K4 (Rook)	K10, K11	K20	K20
	K5 (Queen)	K10, K11	K23	K23
	K6 (King)	K10, K11	K23	K23

K20 names the set of rank and file directions (D1, D5, D9, D13); K21 the set of diagonal directions (D3, D7, D11, D15); K22 the set of Knight directions (D2, D4, D6, D8, D10, D12, D14, D16); and K23 the set of rank, file, and diagonal directions (D1, D3, D5, D7, D9, D11, D13, D15). L13 names a two-member list composed of directions D3 and D15; L14 a two-member list composed of D7 and D11.

In the data structure there are eleven lists that group the chess men by types or otherwise useful categories. Table 2 lays these out:

Table 2

Lists of Functional Groupings of Pieces

Grouping	List Name	List Members
White Pawns	L11	M9, M10,....., M16
Black Pawns	L12	M17, M18,....., M24
Bishops	L15	M3, M6, M27, M30
Rooks	L16	M1, M8, M25, M32
Knights	L17	M2, M7, M26, M31
Queens	L18	M4, M28
Kings	L19	M5, M29
White Rooks, Bishops, and Queen	L20	M3, M6, M1, M8, M4
Black Rooks, Bishops, and Queen	L21	M27, M30, M25, M32, M28
White Knights	L23	M2, M7
Black Knights	L24	M26, M31

For each side, moreover, there is a <u>man with type</u> list: L3 for White, L4 for Black. The first symbol on list L3 is K10 (the symbol for the White side) followed by M1, K4, M2, K2,..., M16, K1; that is, M1 (the White Queen Rook) is of type K4 (a Rook, see Table 2), M2 of type K2 (a Knight), etc.

For each side, there is also a <u>type of man with move directions</u> list: L7 for White, L8 for Black. List L7 contains K1, L13, K2, K22, K3, K21, K4, K20, K5, K23, K6, K23, and differs from List L8 only in that the latter's K1 value is L14--different Pawn capturing directions. The move and capture directions, which are themselves the names of lists (K20, K21, etc.), have already been catalogued in Table 1 above.

Routines E6 and E7 make use of lists L3, L4, and L11-L19 (see Table 2) during the initial board set-up in order to assign to each man his type, color, move directions, and capture directions. In the initial position, for instance, the list structure of M8 would look like this:

Name	Symbol	Comments
M8	9-0	
9-0	0	
	S0	(Attribute: "Man on what square?")
	S8	(Value: S8, the square KR1)
	A1	(Attribute: "Type of man?")
	K4	(Value: K4, a Rook)
	A2	(Attribute: "Color?")
	K10	(Value: K10, White)
	A20	(Attribute: "Move directions?")
	K20	(Value: K20, a list of directions D1, D5, D9, D13)
	A21	(Attribute: "Capture directions?")
	K20	(Value: K20, a list of directions D1, D5, D9, D13).

Positions

A chess position can be described fully in terms of squares and pieces. Since MATER is supposed to find a checkmate in any given position, obviously some representation is necessary for encoding that particular position. This representation is a describable list called the position list L10. Its main list consists simply of the name of each man present on the board and the name of the square each man occupies, arranged in attribute-value pairs. Its description list contains a set of special attributes pertinent to the characterization of the position; in particular, S65, the "Whose move is it?" attribute that flip-flops between K10 (White on move) and K11 (Black on move); S66, the name of the castle list that contains the Kings and Rooks still "eligible" for castling; S67, the signal cell that gets set when an en passant capture is in the offing; and S69, the name of the most recent move made on the board.

Routine E10 takes as input any position list--either the initial position or the mating position, which is read in from cards by routine E90--and converts it into a set of associations between pieces and squares such that every piece has an attribute S0 ("Square on?") with the square that piece occupies as value, and every square has an attribute M0 ("Man on?") with the name of the chess piece--if there is one--occupying that square as its value. This is how the M0 and S0 values are assigned for all pieces and squares in any particular position. (For the initial position see the example M0 for S8 and S0 for M8 on page 16 and page 19, respectively.)

With this, discussion of what might be called the "static" perceptual relations on the chessboard comes to a close. What follows is the bundle of basic routines that attempt to provide "dynamic" perceptual relations to the program.

MOVE-MAKING CAPABILITIES

Moves are the operators that transform one chess position into another. What are the common properties of chess moves? Each involves a piece, or sometimes two, going from one square to another. If the "to square" is already occupied, the move is called a capture. If the "to square" is on the eighth rank and the piece a Pawn, it is called a promotion. But in all cases the common "from-to" property holds.

This permits a move to be represented as the name of a description list containing a "from square" (as the value of an attribute A40) and a "to square" (as the value of an attribute A41). This information is sufficient to specify most moves. There is a special class of moves, however, which, while adhering to this "from-to" pattern, introduces some idiosyncratic properties of pieces. Each of the five members in this class is assigned a different value to the special move attribute A42: for King's side castling and Queen's side castling, the value of attribute A42 equals K12 and K13, respectively; for a double Pawn move V(A42) = K15; and for an en passant response thereto, V(A42) = K14. Finally, for a Pawn promotion, V(A42) is the type of man called for, i.e., either K2, K3, K4, or K5.

Five steps are required to make a move: first, the move must be constructed; second, it must be tested for legality; third, for repetition of position; fourth, it must actually be made on the board; and, fifth, it should be printed.

Routines E51 and E52 construct regular moves and special moves, respectively. Both create a new cell or symbol, which becomes the name of the move. Both take as input the square from which the piece is to move and the square to which it is to move and assign these as values of attributes A40 and A41, respectively. For the special move routine, E52, the type of move must also be specified as input; it is assigned as the value of attribute A42. The name of the man moved is also recorded as the value of another attribute A51, for reasons that will appear under step 3 below. The move 1.P-K4, for example, would be represented as follows (where a_1 and a_2 are internal cell names):

Name	Symbol	Comments
a_1	9-0	
9-0	0	
	A40	(Attribute: "From square?")
	S13	(Value: S13, the square K2)
	A41	(Attribute: "To square?")
	S29	(Value: S29, the square K4)
	A51	(Attribute: "Man moves?")
	M13	(Value: M13, the White King Pawn)

Similarly, the special move P-K8=Q would be represented in the following format:

Name	Symbol	
a_2	9-0	
9-0	0	
	A40	(Attribute: "From square?")
	S53	(Value: S53, the square K7)
	A41	(Attribute: "To square?")
	S61	(Value: S61, the square K8)
	A42	(Attribute: "Special move?")
	K5	(Value: Promotion to a Queen)
	A51	(Attribute: "Man moved?")
	M13	(Value: M13, the White King Pawn)

Second, a routine E18 checks to insure that a newly constructed move is legal. The routine tests, for example, whether a Bishop is moving through a Pawn, whether a Rook is making Bishop moves, whether a player is castling through check, and the like. The output is a simple "+" ("yes, the move is legal") or "-" ("no, the move is illegal").

Third, according to the laws of chess a threefold repetition of position constitutes a draw and, according to the laws of computers, a loop. Before a move is executed, therefore, a routine E55 tests if the move under consideration has been played before in this same position. The position could not have occurred before if the move is irreversible, that is, if once the move is made on the board no subsequent set of legal moves can ever regain the exact same position. Captures, Pawn moves, and castling are all irreversible. Thus when a capturing move is constructed (step 1), it is given an attribute A44 with the man captured as its value. When a castling move is constructed, its status as a special move is recorded as the value of attribute A42. And for a Pawn move, a record is kept via the man moved attribute, A51. Routine E56, called by E55, tests for any of these three conditions to declare a move irreversible. If none of them obtains, E55 must take some further comparisons between the A40 and A41 values of the proposed move and earlier moves.

Fourth, a routine E65 makes a regular move on the board; routines E71-E75 and E81-E85 execute the special moves. A move is made by updating the position list, which is done in two steps: first, with the assistance of routine E11, the description lists of the pieces and squares affected by the move are updated. For the "from square" the value of attribute M0 is erased and for the "to square" a new value of M0 is added and the old deleted if necessary. Similarly, for the piece moved the value of S0 is revised. Second, the signal cells--S65, S66, S67 and S69--affected by the move are reset.

Since different routines are needed to make each of the five special moves, these routines are simply associated with their respective special move values in the data section of the program. Data list L32 contains K2, E71, K3, E71, ..., K15, E75, where the K-values of attribute A42 are as described on page 20 and the E-values are the actual names of the routines that make the move in question. Data list L33 is an analogous list for special move captures, consisting of the same K-values but E-valued routines numbered in the E80's.

Fifth, there is a print routine, E16, which prints out the name of the move, the "from square," the "to square," and the man captured, if any.

ADDITIONAL BOARD PROCESSING CAPABILITIES

In addition to the move-making capabilites just described, there is a second group of routines intended to provide the machine with some more of the perceptual capabilities a human possesses; these are the board processing routines which provide answers to questions asked of the board. Routine E13 finds the direction, if one exists, between two given squares. Routine E14 tests to see if there is a piece between two squares in a given direction, and E24, if there is one and only one piece between two squares in a given direction. E15 tests if a piece is under attack, and routine E26 asks specifically if that piece is the enemy King. Routine E33 tests whether a given square is under attack, while E34 builds the list of men of a particular color attacking a given square. Routine E36 tests if a given square is defended. These are some of the more important "building block" routines and provide a substrate for the move tree, the subject of the next section.

IV. ORGANIZATION OF THE ANALYSIS TREE

THE PROBLEM

As stated in Section I the mating program analyzes chess positions. An analysis of a position--as the term is used here[3]--consists of the set of moves and evaluations made in the course of resolving the choice-of-move problem. Taken together, moves and positions make up a tree of possibilities in which moves operate on positions to produce new positions (see Figure 3) and on which evaluations of positions and of moves in achieving desired positions can be hung as desired.

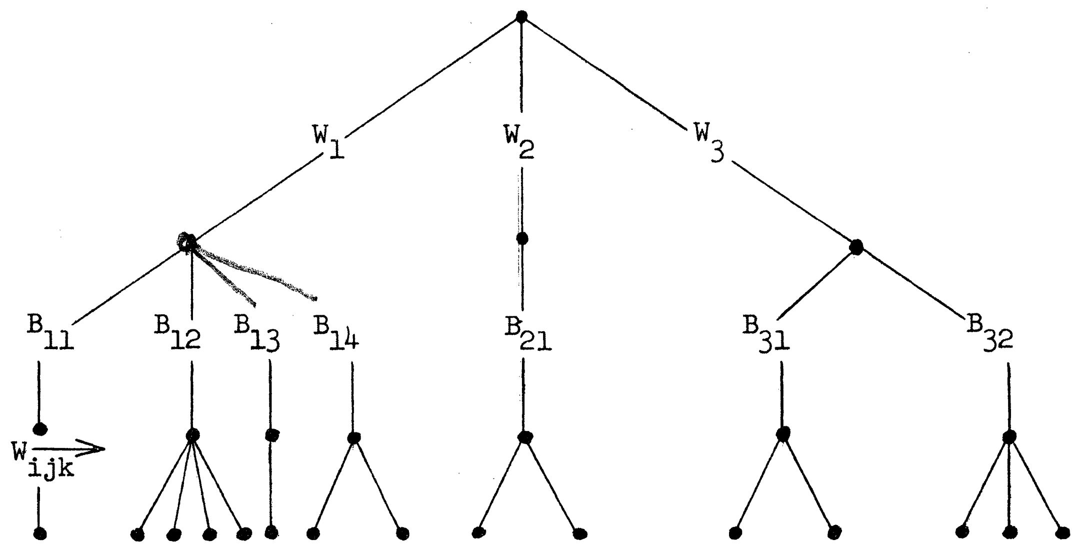


Figure 3. The Analysis Tree
(The W's are White moves, the B's, replies)

The dots or nodes in Figure 3 denote positions (static states) and the lines between dots denote moves (operators) that transform one position into another.[4]

---

[3] Chess players would probably prefer to define "analysis" as the finished product rather than the process of search, laying stress on the "right" moves and continuations rather than emphasizing how these were arrived at.

[4] Simon and Newell (1956) have often drawn this difference equation analogy to the problem-solving process: given an initial state description and a desired state, the problem is to find a process description that operates on the initial state to produce the desired one. In discussion of the Logic Theorist (Newell, Shaw & Simon, 1962), for example, the logic expressions correspond to the static states, and the rules of inference, to the operators.

How should the analysis of a position be conducted? Figure 4 presents one simple scheme:

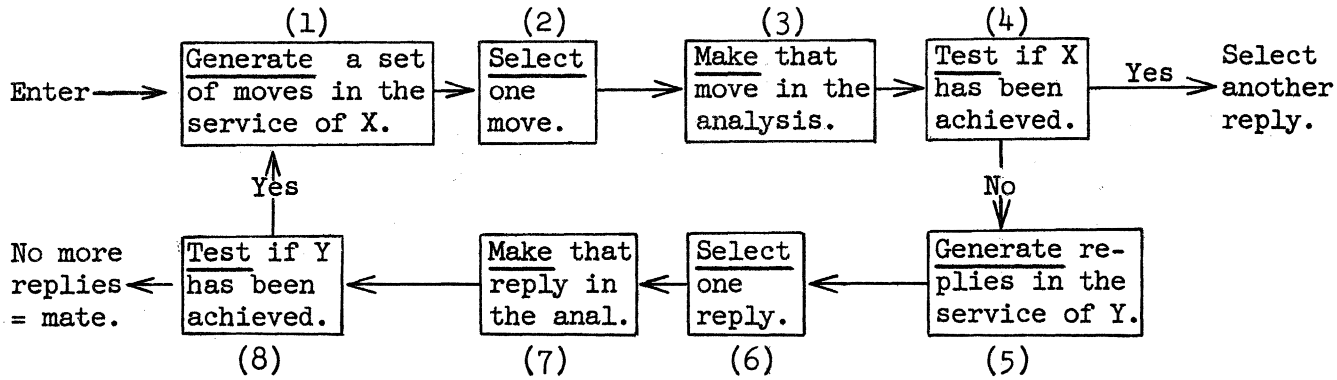


Figure 4. A Simple Recursive Mating Scheme

Would this scheme be workable if it were made operational? For example, let:

(1) "X" be defined as checkmate and the program be given the capability of generating moves in its service;

(2) the criteria for deciding among moves be specified by certain rules of selection;

(3) the program be given the capability of making moves and updating board positions;[5]

(4) a test be provided so that the program "knew" if it had achieved "X"; and

(5-8) the corresponding provisions be made for "Y," defined as escaping check, and for choosing among replies.

The answer is no, not quite. The scheme lacks a means for recovering from false starts, for retracing its steps when it runs into blind alleys. Indeed, what is lacking can be seen by considering the difference between actually playing a game of chess and analyzing a chess position: the course of analysis is fickle and reversible, whereas in an actual game a move once made cannot be unmade. In other words, the scheme outlined above needs provisions for unmaking moves and for abandoning seemingly unpromising positions as much as it needs the capability of making moves and pursuing promising positions. Ideally, one should like to be able to enter and reenter the move tree at any node (position) at any time and from there to proceed down any branch, old or new. Indeed, providing

---

[5]This capability has already been provided, as explained in Section III.

capabilities for reinstating the right position at the right time is probably the central problem of organization at this level of the program, while making operational and making sense out of steps 1-8 above is probably the central problem at the next higher conceptual level. This section reports on the former problem: implementation of a flexible move tree. Section V is devoted to the latter: defining the problem and the heuristics of search.

BUILDING THE TREE

The notion of analysis as a tree search is misleading to the extent that it implies that each step consists solely of selecting a move from the many available alternatives. Actually, the process is more one of generating moves as one goes along, of building one's own tree. This is the very distinction Maier (1960) has drawn between decision making under conditions of uncertainty[6] and problem solving: "Decision making implies a given number of alternatives, whereas in problem solving the alternatives must be created. Thus, problem solving involves both choice behavior and the finding or creating of alternatives" (Maier, 1960, p. 218).

In every chess position, of course, the rules of the game place an upper limit on the number of possibilities that can be created; de Groot (1965) found that, averaged across the course of a game, the mean number of move possibilities lies somewhere between 30 and 35. This is the full-grown tree; the one the searcher actually builds is much smaller: on the average of 4 or 5 branches at the top node and smaller thereafter.

The question addressed in this section is the technical one: How does one build a tree? Two of the necessary raw materials, limbs and nodes--moves and positions, respectively--were made ready in the last section. In this section, the means by which these materials are structured into a tree of possibilities is the first concern.

To see the second concern of this section clearly, it helps to think of the chess player climbing the tree as he builds it. Crawling along a branch in one direction corresponds to making a move in the current position (node) while traversing it in the opposite direction corresponds to unmaking a move and restoring the previous position (node). This ability to back up the tree is what enables a player to abandon unpromising lines of investigation and start afresh. In starting afresh, moreover, the player may either re-investigate branches he has previously built or build new ones.

---

[6]According to Luce and Raiffa (1957) decision making under uncertainty is the condition in which the outcomes of the various known alternatives are unknown.

Third, as the player builds and climbs he also accrues and retains information. The information garnered en route and the use to which it is put are in large part what Denkpsychologists have called the development of the problem. That is, the searcher's conception of the problem at any one time consists of the information he has about the problem, how he has evaluated this information, and even how it has shaped his definition of what the problem is. (Cf. Duncker, 1945; de Groot, 1965.) Provisions for gathering information are considered in both this section and the next; the use to which information is put and the matter of problem development are more properly treated in the subsequent sections.

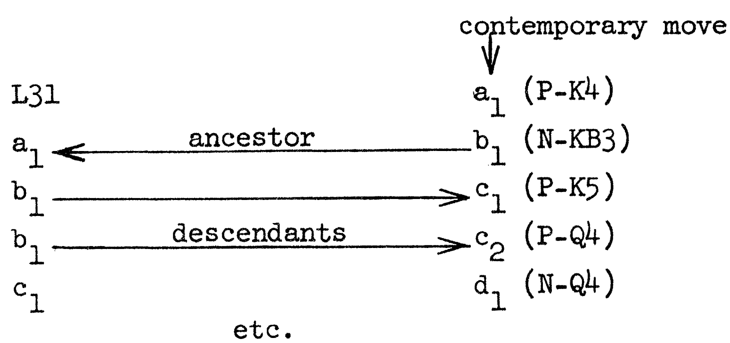
Most of the organizational problems are solved via the description lists of moves. For convenience of reference the entire list of possible attributes a move can take on is set forth in Table 3:

Table 3

Move Attributes

A40	From square
A41	To square
A42	Special move
A43	Man removed from castle list
A44	Man captured
A45	Square of captured *en passant* Pawn
A46	Ancestor
A47	List of descendants
A48	Irreversibility of move (J3 if reversible; J4 if irreversible)
A49	Value of move (J4 if win for first side)
A50	Number of descendants
A51	Man moved
A52	Double check (J3, no; J4, yes)
A53	Discovered check (J3, no; J4, yes)
A54	Checking move (J3, no; J4, yes)
A55	Descendant's list of mate threats
A56	Threatened mating square
A58	Mating piece on V(A56) square
A60	Move value = MATE
A61	Move value = NO MATE
A62	Move value = NO VALUE
A63	Reply NOMV (no move)
A65	Number of checking moves generated to date
A66	Number of replies generated to date
A70	List of King replies
A71	List of capturing replies
A72	List of interposing replies

With respect to the first concern--provisions for holding the tree together--a signal cell S69 and attributes A46 and A47 do the job. S69 contains the name of the most recent move made on the board--the contemporary--while attributes A46 and A47 are its ancestor and its list of descendants, respectively. The course of the analysis is preserved on a list L31, which is a log of the analysis tree and consists of a running record--a chain of moves--linked together as ancestors, contemporaries, and descendants, as follows:



Second, because of the strong family ties just described, one can eventually crawl one's way down any branch from any node and then back up. That is to say, one can make or unmake any move. Two routines, E65 and E66, make and unmake moves, respectively. The procedure for making a move was described in Section III; the procedure for unmaking a move is exactly the reverse. With the help of routine E11 the position list is restored, that is, the description lists of the pieces and squares affected are restored and the signal cells--S65, S66, S67, and S69--are reset.

At any node a new limb may also be constructed (by routines E51 and E52; see page 20) simply by specifying the "from square" and the "to square" (and special move status, if any) whereupon the move is added to the list of descendants, V(A47), and assigned an ancestor, V(A46).

Third, information gathered in the search for mate is stored on the description list of the move that gathers it. (See Table 3.) When a move is constructed, its ancestor is always assigned as a value of A40 and the man moved is always assigned as a value of A51. Conditionally, a move is assigned a value of A43 if a Rook or King is removed from the castle list, a value of A44 if a man is captured, a value of A45 if a Pawn is captured <u>en passant</u>, a value of A52 if it is a double check, a value of A53 if it is a discovered check, and a value of A54 if it is a checking move at all.

Evaluative information is also gathered: attribute A49 records the win-lose value of a move, while A60, A61, and A62 represent mate, no mate, and no value, respectively. If a checking move has no descendants, that

move mates; consequently, attributes A70, A71, and A72 record the kinds of replies to check. Attribute A47 lists the descendants <u>in toto</u> and the value of A50 is a count of them.

The point here is to illustrate how information is hung on the move tree as it is gathered. How the information is retrieved and utilized is a topic for the next section.

V. THE EXECUTIVE AND HEURISTICS OF SEARCH

In the social sciences recent research on small groups has led to a sharp distinction between the structure of communication networks, on the one hand, and the content of the communications, on the other (Leavitt, 1964). A thought process, similarly, can be studied from both these points of view. Structurally, trees derived from protocols of thought processes or from computer programs can be compared on many formal criteria; e.g., branchiness at each node, maximal depth, total number of nodes or branches, the path charted through the tree, phase structure, etc. These measures relate to de Groot's (1965) important general thought methods and because of his remarkable finding that "it is not generally possible to distinguish the protocol of a grandmaster from the protocol of an expert player solely on structural and/or formal grounds" (*Ibid.*, p. 319), these measures raise many questions about invariances of immediate memory, stop rules under varying conditions of uncertainty, and other perhaps task-independent aspects of the thought process.

The content of the thought process, that is, which branches are investigated and which nodes are reached, are measures of selectivity and relate to the chess player's system of playing methods. Here is where de Groot (1965) found the contrasting levels of skill among chess players showing up: in the quality of the chosen move, in the quality of the analysis in its support, and in certain striking perceptual differences that arose when positions were exposed for but a few seconds.

The research reported in this paper is primarily an effort in the latter direction; that is, it is addressed to the question of selectivity, though a human's behavior will be compared with the program's from both points of view in the next section. Specifically, the question here is: In a given position, what moves should be considered and in what order? Or, equivalently, in a given position, what moves should be generated, when, and, subordinately, how? As has already been argued, chess players are highly selective in the moves they look at, a selectivity based on their heuristics of search or on what de Groot (1965) called their system of playing methods or of experiential linkings. What follows then is a discussion of the search heuristics incorporated into the early version of the mating program, some measures on its search behavior, a brief description of the routines that effect the move generation, and, finally, later developments incorporated into the second version of the program, MATER II.

MATER I

Restricting the mobility of the opponent's pieces is a recognized principle of chess strategy. It is particularly important in checkmating combinations since checkmate is defined as an unopposed attack on an enemy King whose mobility has been reduced to zero. Strategically this means the attacker strives to gain control over (1) the square the enemy King occupies, as well as (2) all the squares contiguous to it that do not contain an enemy piece. If just condition (1) obtains, the enemy King is simply in check; if just condition (2) holds, the enemy King is stalemated; while if both (1) and (2) hold, he is checkmated. Viewed in this light, checkmate is a process of acquiring controls, of more and more restricting the enemy King's mobility. This principle is the cornerstone of the mating program.

The restriction of mobility principle applies to the generation and selection of moves as well as to decisions about when to abandon search in certain directions. Thus in the mating program: Only checking moves (in MATER I) and moves that threaten mate in one move (in MATER II) are generated for the attacker; the move selected for investigation is the one that most restricts the opponent's mobility; and search is continued down a chosen path only so long as the opponent's mobility is on the decline. This--the rate of growth of the search tree--is an alternative formulation to an evaluation function for terminating search in a particular direction.

Before illustrating the flow of control and the program's executive structure, it is necessary to introduce the notion of a try-list, a notion similar to the "pool of subgoals" in the Logic Theory Machine (Newell, Shaw & Simon, 1963b). Since only one move can be tried out at a time in any particular position, other "eligible" checking moves must wait their turn on a list, L35--the try-list. This list has two noteworthy properties: first of all, it is an ordered list, and second it is independent of a move's level.[7] Such independence proved powerful in directing search.

The list is ordered by a fewest-replies heuristic: highest priority goes to moves with the fewest number of legal replies, while checking moves with more than four legal replies are discarded entirely. Ties are broken by giving priority to double checks, then to checks that have no capturing responses, then to the order in which the checks were generated. The second property--that checks from all levels are mixed--effects the evaluative principle that search is continued down a particular path only so long as the opponent's mobility is on the decline: when the number of replies at some node in the current line of investigation is equal to or greater than the number of replies at some prior node, the current line is abandoned, the prior node restored, and the

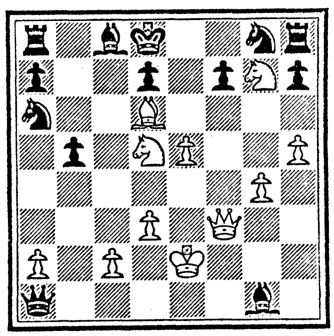
[7] The level of a move refers to its depth in the move tree, i.e., how many moves out it is from the particular starting position.

alternative that had once been passed over is tried. This nips in the bud unpromising proliferations in the move tree.

In addition to the notions just described--rate of growth serving to terminate search in a particular direction, the set of considerable moves serving to restrict the set of applicable operators in the given position, the try-list ordered by the fewest-replies heuristic serving to stipulate the application of offensive operators--some heuristics of chess strategy serving to stipulate the order of application of defensive operators can also be seen more clearly from the following illustrative position taken from Fine (1952) and MATER I's performance on it.[8]

The following layout is adapted to the simple recursive scheme of Figure 4.

Diagram 36



(1) Generate checking moves: 1.N-K6ch; B-K7ch; B-B7ch; Q-B6ch.

(2) Select one move for further analysis:

(a) For each check generate and count replies:

1.N-K6ch	1.B-K7ch	1.B-B7ch	1.Q-B6ch
1...K-K1	1...NxB	1...NxB	1...NxQ
1...QPxN	1.	1.	1...N-K2
1...BPxN			2.
3.			

[8] They can also be seen more clearly within the picture of heuristic search in general in Newell and Ernst (1965).

(b) Transfer checks to try-list; order them by their "u" values (number of replies):

L35	u
1.B-K7ch	1
1.B-B7ch	1
1.Q-B6ch	2
1.N-K6ch	3.

(c) Select and delete top move from try-list L35: 1.B-K7ch.

(3) Make that move (1.B-K7ch) in the analysis.

(4) Test if checkmate has been achieved: No.

(5) Generate replies to relieve check: 1...NxB.

(6) Select "best" reply[9] : 1...NxB.

(7) Make that reply (1...NxB) in the analysis.

(8) Test if check has been relieved: Yes.

(1) Generate checking moves: 2.N-K6ch.

(2) Select one move for further analysis.

(a) For each check generate and count replies:

2.N-K6ch
2...K-K1
2...QPxN
2...BPxN
3.

(b) Transfer check to try-list: order them by their "u" values:

L35	u
1.B-B7ch	1
1.Q-B6ch	2
1.N-K6ch	3
2.N-K6ch	3.

(c) Select and delete top move from try-list L35: 1.B-B7ch.

---

[9]"Best" is defined in the text immediately following this example.

(3) Make that move (1.B-B7ch) in the analysis, restoring the board to the initial position.

(4) Test if checkmate has been achieved: No.

(5) Generate replies that relieve check: 1...NxB.

(6) Select best reply: 1...NxB.

(7) Make that reply (1...NxB) in the analysis.

(8) Test if check has been relieved: Yes.

(1) Generate checking moves: 2.N-K6ch; Q-B6ch.

(2) Select one move for further analysis.

(a) For each check generate and count replies:

2.N-K6ch	2.Q-B6ch
2...K-K1	2...NxQ
2...QPxN	2...N-K2
2...BPxN	2.
2...NxN	
4.	

(b) Transfer checks to try-list; order them by their "u" values:

L35	u
1.Q-B6ch	2
2.Q-B6ch	2
1.N-K6ch	3
2.N-K6ch	3
2.N-K6ch'	4.

(c) Select and delete top move from try-list: 1.Q-B6ch.

(3) Make that move (1.Q-B6ch) in the analysis, restoring the board to the initial position.

(4) Test if checkmate has been achieved: No.

(5) Generate replies to relieve check: 1...NxQ; N-K2.

(6) Select best reply: 1...NxQ.

(7) Make that reply (1...NxQ) in the analysis.

(8) Test if check has been relieved: Yes.

(1) Generate checking moves: 2.N-K6ch; B-K7ch; B-B7ch.

(2) Select one move for further analysis.

(a) For each check generate and count replies:

2.N-K6ch	2.B-K7ch	2.B-B7ch
2...K-K1	0.	2...NxB
2...QPxN		1.
2...BPxN		
3.		

(b) Transfer checks to try-list; order them by their "u" values:

L35	u
2.B-K7ch	0
2.B-B7ch	1
2.Q-B6ch	2
2.N-K6ch"	3
1.N-K6ch	3
2.N-K6ch	3
2.N-K6ch'	4.

(c) Select and delete top move from try-list L35: 2.B-K7ch.

(3) Make that move (2.B-K7ch) in the analysis.

(4) Test if checkmate has been achieved: Yes.

(6') Select next best reply: 1...N-K2.

(7') Make that reply (1...N-K2) in the analysis, restoring the board to the appropriate position.

(8') Test if check has been relieved: Yes.

(1) Generate checking moves: 2.N-K6ch; BxNch; B-B7ch; QxNch.

(2) Select one move for further analysis.

(a) For each check generate and count replies:

2.N-K6ch	2.BxNch	2.B-B7ch	2.QxNch
2...K-K1	0.	2...NxB	0.
2...QPxN		1.	
2...BPxN			
3.			

(b) Transfer checks to try-list; order them by their "u" values:

L35	u
2.BxNch	0
2.QxNch	0
2.B-B7ch'	1
2.B-B7ch	1
2.Q-B6ch	2
2.N-K6ch"	3
2.N-K6ch"'	3
1.N-K6ch	3
2.N-K6ch	3
2.N-K6ch'	4.

(c) Select and delete move from try-list L35: 2.BxNch.

(3) Make that move (2.BxNch) in the analysis.

(4) Test if checkmate has been achieved: Yes.

(6') Select next best reply: None.

(7') Make that reply (None) in the analysis.

(8') Test if check has been relieved: No.

Mark MATE and print move-tree (Figure 5).

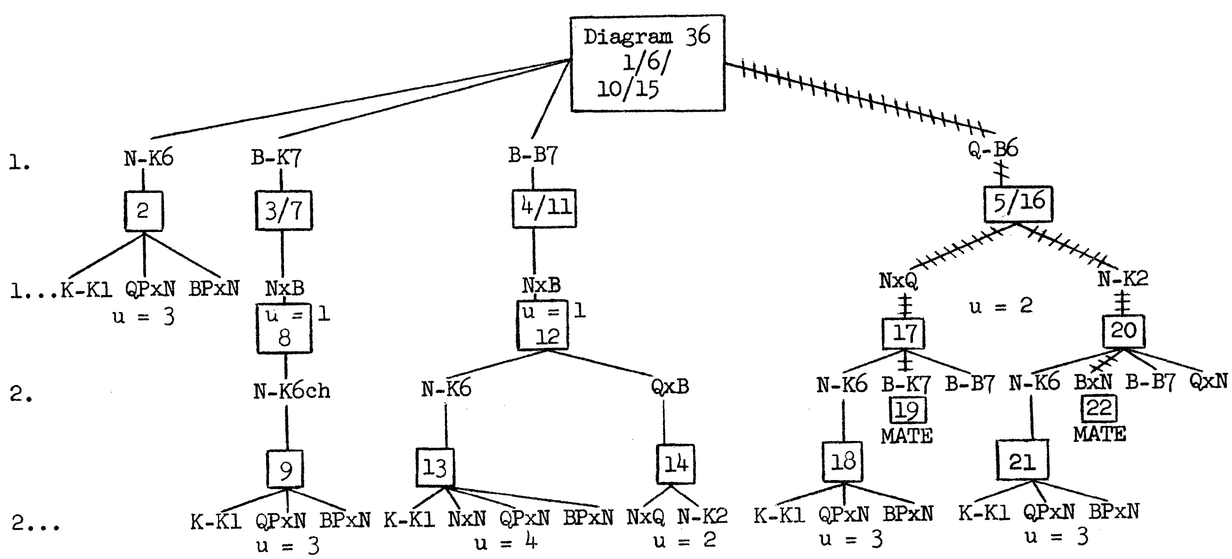


Figure 5. MATER I's Analysis Tree of Diagram 36, Fine (1952) (The u's are a tally of the number of replies by which the priority of moves on the try-list is established. The square boxes represent positions and the numbers in them trace the course of the investigation--the order in which the positions were taken up. The crosshatched branches trace the mating path.)

This example also illustrates the criteria by which the order of application of defensive operators (moves) is accomplished: by "best" reply is meant that reply that seems most likely to give the attacker trouble. Thus the priority of defensive moves Black tries is, first, the capture of the most valuable White pieces by the least valuable Black pieces followed by King moves, interpositions, then order of generation. Again this is an attempt to clip unnecessary proliferations in the move tree: if there is a "killing" reply to a checking move, further analysis of that checking move would seem futile.[10]

Measures of Search Behavior

Many measures of search behavior can be picked off an analysis tree like MATER I's of Diagram 36, Figure 5. For example, the tree can be characterized by counting the number of positions or the number of moves. Simon and Simon (1962) call the total number of positions examined the "exploration tree"; in Diagram 36, above, the position count yields an exploration tree of size 16. In general, however, more moves are seen than positions are investigated, which is to say that some moves remain unexplored, such as the replies to 1.N-K6ch in the example above. The count of moves seen--the uninvestigated as well as the investigated ones--will be called the discovery tree; in Diagram 36, above, this move tally yields a discovery tree of size 36 (14 checks and 22 responses). One further refinement can be carved out of the exploration and discovery trees; namely, the "verification tree," which Simon and Simon (1962) define as the total number of positions required to prove the combination--the positions resulting from the single best move at each node for the attacker and from every legal move at each node for the defender; respectively, the positive and negative parts of the proof schema (de Groot, 1965, Section 9). The verification tree "is precisely analogous to the correct path in a maze. It is a tree instead of a single path because all alternatives allowed to the defender must be tested" (Simon and Simon, 1962, p. 427). In Diagram 36, above, the branches of the verification tree are crosshatched, yielding a position count of size 6 or, alternatively, a move count of size 5.

These measures do not reveal the time order in which the tree was generated. Human chess players are fickle tree climbers, "progressive deepeners," to use de Groot's (1965) term for the phenomenon: "The investigation not only broadens itself progressively by growing new branches, countermoves,

---

[10]This is the minimax assumption; namely, that the opponent will make his strongest reply at every opportunity. McCarthy's <u>killer heuristic</u> (see Kotok, 1962) assumes that a killing reply to one checking move may be a killing reply to other checking moves and thus should be looked at first.

or considerable own-moves, but also literally deepens itself: the same variant is taken up anew and is calculated further than before" (de Groot, 1965, p. 266). In other words, the search strategy is an important structural characteristic of the thought process. In Figure 5 the order in which positions are taken up is captured by numbering the nodes (positions) in the analysis. These measures will be used for comparative purposes in Section VI.

Routines

How are checking moves and replies actually generated in any given position? There would seem to be two tacks, corresponding to a one-many approach and a many-one approach. In trying to find all the checks in a given position, for example, one could either radiate out from the enemy King and from each square, search for a piece that can get there and give check (the one-many approach), or converge from the squares along the move directions of each attacking piece onto the enemy King's square (the many-one approach). If there are many pieces on the board, the former is the more efficient; if few, the latter.

G1 is a master routine that procures all checks in a given position. It employs the many-one approach, calling subroutine G11 for Queen, Bishop, and Rook checking moves; G12 for Knight and regular Pawn checks; and G13 for double Pawn moves that administer check. Similarly, R21 procures all replies to a given checking move: R11 generates all the King moves that get out of check, R12, all the captures of the checking piece, and R13, all the interpositions. In this way the mating program is able to enumerate all checks and all replies in a particular position.

MATER II

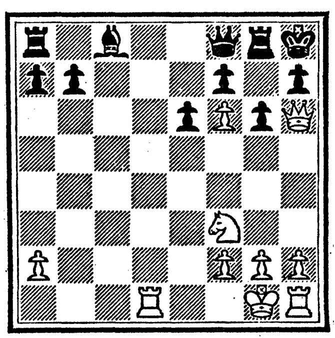
> "In designing search programs it is useful to distinguish the strategy of search from the information that is gathered during the search. The search strategy tells where to go next, and what information must be kept so that the search can be carried out. It does not tell what other information to obtain while at the various positions, nor what to do with the information after it is obtained. There may be strong interaction between the search itself and the information found, as in the decision to stop searching, but we can often view this as occurring within the confines of a fixed search strategy" (Newell and Simon, 1964, pp. 24-5).

MATER II adds a modification to MATER I's search strategy by bypassing the fixedness in the order of application of operators inherent in the try-list. The new search rules states: in the given position pursue immediately and in depth all checking moves that keep the enemy King stalemated (or nearly so), i.e., moves that can only be answered by captures and/or interpositions or, in the absence of both, by one and only one King move (the "nearly so" condition). In addition to altering the program's search strategy by telling it "where to go next," this procedure also gathers information about the position. In this respect it resembles what de Groot (1965) has called a "sample variation," a kind of trial balloon sent up for the express purpose of gathering information to direct subsequent investigation; in this sense it is orientative. Before turning to what information is gathered and how it is used, it should be mentioned that sometimes a sample variation pays off directly--the "sample moves" may be a path to a quick mate.

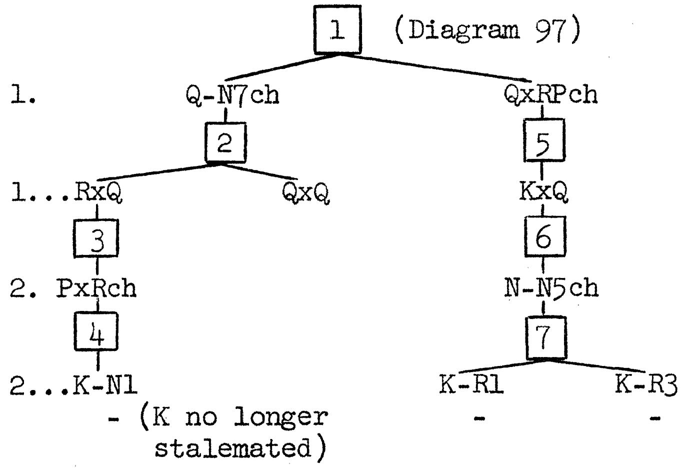
Specifically, a routine G10 conducts the preliminary search and, if no "easy mate" is found, records the sequence of moves investigated on a list B4. A routine G17 makes use of this information later in drawing up a plan of attack. Just how these routines operate can best be seen by considering in three parts MATER II's analysis for a particular position, Diagram 97 from Fine (1952).

The first part has to do with the preliminary search; the second with the use to which the recorded information is put in drawing up a plan; and the third with the exploration and verification of that plan.

Diagram 97



The first ten moves of the discovery tree are these. (Note that both the 1.Q-N7ch and 1.QxRPch sample variations are recorded on list B4.)



Recorded on list B4

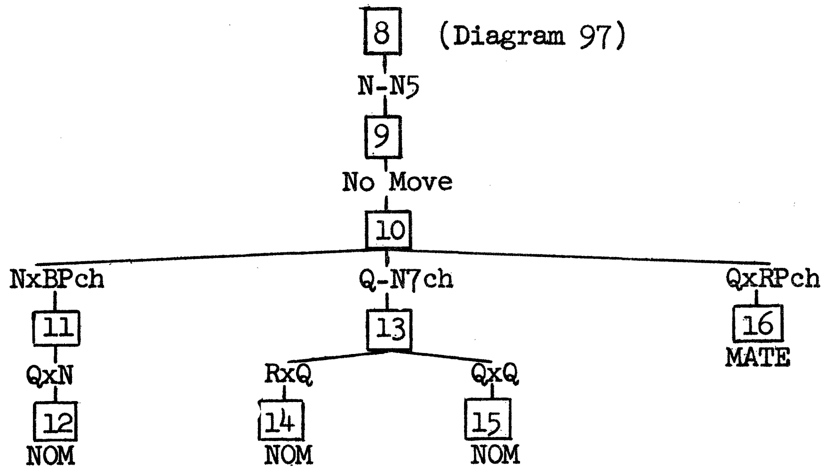
Sequence 1: 1.Q-N7ch, RxQ; 2.PxRch.

Sequence 2: 1.QxRPch, KxQ; 2.N-N5ch.

Clearly, the "wishful thinking" goal of the sample variations went unfulfilled; the preliminary excursions did not yield mate. They do yield two sequences of forcing moves, however, that may be useful in constructing a plan of attack. Indeed, the routine G17 searches list B4 for first move candidates and finds, in this example, 1.PxR and 1.N-N5. The former is rejected as illegal in the current position while the latter is deemed considerable. Note that the set of operators that may be applied in the initial position is expanded in MATER II to include moves other than just checkimg moves, yet the means by which these are generated continues to ensure a high degree of selectivity.

Routine G17 asks if a proposed move, in this case 1.N-N5, threatens mate in one move. It determines the answer by assuming that Black does nothing on his turn, that is, by playing a "No Move" and then seeing if White can enforce an immediate checkmate. And, indeed, 2.QxRP is mate. In other words, White leaves the actual problem space to seek a mate in a simplified planning space (see Newell, Shaw & Simon, 1962), and, in fact, the second part of the move tree is given over to solving the problem in the planning space:[11]

[11] A hybrid version of MATER I and II would first have reinvestigated 1.Q-N7ch and 1.QxRPch, invoking the fewest-replies heuristic, transferring these two checks to the try-list, and elaborating them, before even considering moves which threaten mate in one. Unfortunately the statistics gathered to date on the various versions of the program are too incomplete to say which search strategy is superior across positions, if in fact a correct strategy can be determined independent of position.



Finally, the third stage is devoted to testing the soundness of the plan; that is, suppose Black tries to avert 1.N-N5 and 2.QxRPmate. Can he? It happens that in this position he cannot so that the exploration tree and the verification tree are identical in this stage of the analysis. (The rather lengthy third stage is omitted here but is given in full in Section VI.)

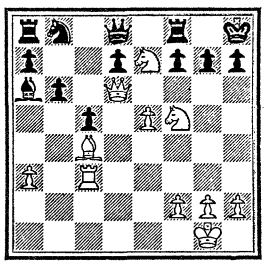
MATER II contains one other highly selective mechanism for finding moves that threaten mate in one. Controls exerted over the enemy King's square and the squares in this immediate vicinity are built into a list structure called the King's Sector. For a given square five kinds of control have been defined:

- no control--the enemy King can move to the given square;
- attacking control--the attacker can move to or capture on the given square;
- occupation control--one of the attacker's pieces occupies the given square;
- block control--one of the defender's pieces occupies the given square;
- X-ray control--the attacker can unmask an attacker control by removing one of his own pieces (corresponding to a "discovery" in chess jargon) or he could unmask an attacking control but for an enemy interposer (corresponding to a "pin").

The King's Sector, L40, is constructed by the four routines E91-E94. The complete structure of L40 and the information contained therein can best be seen in Figure 6, the King's Sector for Diagram 70 from Fine (1952). Attribute Y3 has data term X0 as its value, a tally of the number of uncontrolled squares in the Sector. (See Figure 6.)

```
L40 9-0
9-0 0
    S64 (KR8)
    9-1
    S63 (KN8)
    9-2
    S55 (KN7)
    9-3
    S56 (KR7)
    9-4
    Y3  (No control over how
         many squares?)
    X0 = 1
-----------

Knight          S54 (KB7)
directions      9-5
not used in     S47 (KN6)
tallying X0     9-6  0
```



```
(KR8) 9-1 0
          M0 (Man on KR8?)
          M29 (Black King) 0
(KN8) 9-2 0
          Y7 (X-ray control)
          9-10
          Y6 (Attacking control)
          9-11 0
(KN7) 9-3 0
          Y6 (Attacking control)
          9-12
          Y4 (Block control)
          M23 (Black Pawn blocks KN7) 0
(KR7) 9-4 0
          Y4 (Block control)
          M24 (Black Pawn blocks KR7) 0
(KB7) 9-5 0
          Y6 (Attacking control)
          9-13
          Y4 (Block control)
          M22 (Black Pawn blocks KB7) 0
(KN6) 9-6 0
          Y6 (Attacking control)
          9-14 0
```

```
9-10 0
     M6 (White Bishop X-rays KN8)
     M22 (Black Pawn is X-rayed) 0
9-11 0
     M2 (White Knight attacks KN8) 0
9-12 0
     M7 (White Knight attacks KN7) 0

9-13 0
     M6 (White Bishop attacks KB7) 0

9-14 0
     M4 (White Queen and White
     M2  Knight attack KN6) 0
```

Figure 6. IPL-V List Structure of King's Sector Controls (arranged in attribute-value pairs) of R. Fine's (1952) Diagram 70, from a game Alekhine-Supico, 1942

How is all this information retrieved and used by the program? First, a routine G14 tries to generate mate-threatening moves by converting an X-ray control into a second attacking control. For example, in Diagram 70, routine G14, seizing on the White Bishop's X-ray control of KN8, proposes 1.BxBP but then rejects the move because 2.B-N8 does not administer check, let alone mate. Second, a routine G19, given one attacking control, tries to add a second. Routine G19, seeing an attacking control over KN7 in position 70, proposes to add another with the moves 1.Q-KB6, 1.Q-N6, and 1.Q-R6. Since all three produce mate if Black does nothing, all three are accepted as considerable moves in the plan. (For the complete search tree of Diagram 70, see Figure 7 in Section VI.)

In summary, MATER II contains several mechanisms for generating a selective set of considerable moves. Incorporating MATER I's ability to generate all checking moves and all replies in a given position, MATER II goes on to generate mate-threatening moves based either on their earlier appearance in forced sequences of checking moves or on the function they serve in controlling key squares around the enemy King. Moreover, MATER II has a set of routines, R15, R16, R22, for generating defensive replies to a threatened mate.

MATER II also contains three principal mechanisms in its search strategy for specifying the order in which moves are to be considered. Defensively, in reply to checking moves, captures are preferred to King moves, which, in turn, are preferred to interpositions; while in reply to one-move mate threats, captures are preferred to moves that defend the mating square as well as to interpositions and King runs. Offensively, search is directed by pursuing particular moves in depth so long as the enemy King remains very highly constricted, and then later by pursuing the move that leaves the opponent with the fewest replies. Each of these search evaluators rests on a single criterion: sometimes a line of search is terminated because the defender is left with King moves in reply (nodes 4 and 7 in the move tree of position 97); sometimes a move is rejected because it does not produce immediate mate (nodes 11 and 12 in position 97); sometimes a move just never gets off the waiting list (node 2 in position 36); and checking moves with more than four legal replies are always rejected out of hand. Indeed, it is the thesis here that these kinds of criteria, criteria based on features of the task area, are what regulate chess players' choice-of-move decisions and form a superior representation to complicated weighting functions of the sort employed by Samuel (1963) in checkers and Bernstein and Roberts (1958) in chess. Even though mating combinations are the only facet of the game in which the final evaluators, MATE and NOMATE, are well defined[12]--a degree of certainty nowhere else attained in the rest of the game--the search-directing decisions intermediate to the final choice of move must all be made on far less than certain criteria, just like the rest of the game.

---

[12]"Well defined" is used in the sense that there exists a satisfactory test that enables the player to recognize the solution to his problem (see, e.g., Miller, Galanter & Pribram, 1960, p. 170).

Except for the sample variations recorded on the list B4, the information-gathering mechanisms rely on the description list of the move that gathered it, including the final evaluation, MATE or NOMATE, which are propagated back up the tree by the minimax inference procedure in an attempt to demonstrate the proof of a combination.

WITH ~~Will~~ all these structural and content characteristics of the search process in mind, the program's behavior can now be compared with the human's.

VI. PSYCHOLOGICAL INTERPRETATION AND RESULTS

Does MATER have a legitimate claim to be a model of human cognitive behavior? To a large extent the answer to this question rests on the theoretic rationale of information-processing models in psychology, especially on the problems of communicating and verifying these kinds of models.[13] Three levels of explanation attempt to communicate how the program works; in fact, each of the three preceding sections is directed toward a different facet or level of program detail and specificity: the basic representation, the organization of the move tree, and the strategy and heuristics of search.

With respect to the verification of simulation models in general, and problem-solving models in particular, two criteria for assessment seem to have emerged clearly: an achievement criterion and a process criterion. That is, can the model solve the class of problems it was designed to handle, and are its mechanisms for doing so equivalent to, or even comparable to, a human problem solver's? The answer to the first question is relatively straightforward, to the second, not, since the requirements for equivalence or comparability of process are themselves open to question. With respect to the process criterion, Newell and Simon (1961) have suggested that the adequacy of information-processing models can be evaluated at two levels: (1) at the level of general-thinking methods, ways of reasoning, and strategies, and (2) at a very specific level involving line-by-line comparison of program output with some equally detailed record of human behavior, such as a verbal transcript or protocol. Evidence on achievement as well as both kinds of evidence on process are presented below.

ACHIEVEMENT

MATER I solves combinations which consist of uninterrupted series of checking moves given that the defender at no node in the verification tree has more than four legal replies; MATER II solves combinations that begin either with checks or with one-move mate threats and checking moves thereafter. This limitation on the class of moves the program can see restricts severely the class of combinations on which the model can be tested. Nevertheless, the program has been tested on material taken from Fine's (1952) chapter on the mating attack. Solutions to one class of positions in the chapter call for an uninterrupted series of checking moves ending in mate

[13]For an excellent discussion of the theoretical and methodological issues involved, see Reitman (1964) or the second chapter of his forthcoming book (Reitman, 1965); and the Epilogue to de Groot's (1965) book.

(51/129 positions). Another class of positions is solved with one-move mate threats and checking moves thereafter (5/129 positions). In the residual class, mate can either be averted through a sacrifice of material or the mate is not "forced," as the term was defined in Section I (73/129 positions).

MATER I's Achievement

MATER I found solutions to 43 of the 51 mating positions. The machine missed one combination entirely by failing to move a Pawn that gave a discovered check and exhausted available space before finding the other seven.[14] Table 4 below breaks these 43 positions down according to certain structural measures of search behavior: the depth of search to mate (D), the mean size of the verification tree necessary to prove the combination ($\overline{VT}$ measured in moves), and the mean size of the discovery tree generated in searching for mate ($\overline{DT}$ measured in moves). These latter two measures were defined in Section V.

Table 4

MATER I's Performance on 43 Positions From Fine (1952)

N (positions)	D	$\overline{VT}$	$\overline{DT}$	$\overline{VT}$/D	$\overline{DT}$/D	$\overline{VT}/\overline{DT}$
15	2	3.5	15.5	1.8	7.8	4.4
11	3	5.4	24.6	1.8	8.2	4.6
14	4	9.9	61.5	2.5	15.4	6.2
2	5	11.0	56.0	2.2	11.2	5.1
1	8	17.0	108.0	2.1	13.5	6.4
43						

Simon and Simon (1962) suggest depth, number of positions in the exploration tree, and number of positions in the verification tree as measures of the difficulty of combinations. They remark that the four positions in their sample "are not ordered in the same way with respect to the different measures of difficulty" (*Ibid.*, p. 428). Using depth, number

[14] In particular, MATER I failed to see 2.P-B6ch in position 148. The seven positions that exhausted available space did so because the fewest replies heuristic failed to discriminate among alternative checking moves: among six alternative discovered checks in positions 41 and 100; among a large number of initial checks available in positions 109 and 140; and among a large number of checks in depth involved in a King hunt on the open board in positions 111, 130, and 157.

of moves in the discovery tree ($\overline{DT}$), and number of moves in the verification tree ($\overline{VT}$) as equivalent measures, the data of Table 4 do show, with but one exception, the same ordinal relationship across measures, at least when averaged over the 43 positions.

Between depth and the size of the verification tree ($\overline{VT}$/D) there is a fairly close correlation--around two moves in the verification tree per move in depth. This confirms the Simon and Simon (1962) observation: the tree varies linearly, not exponentially, with depth, and it probably is this property that makes deep analysis possible in combinations. Unfortunately, neither $\overline{DT}$/D nor $\overline{VT}/\overline{DT}$ shows any consistent relationship; a larger sample of combinations of depth 5, 6, 7, and 8 would be required before any speculations about step functions could even be offered.

The only roughly constant ratio, $\overline{VT}$/D, has more to do with the characteristics of mating positions than with characteristics of the mating program. The only measure on the program's search behavior is $\overline{DT}$ and there seems to be no consistent relationship between it and the two measures on the combinations ($\overline{DT}$/D and $\overline{VT}/\overline{DT}$).

MATER II's Achievement

MATER II has been tested on all five positions from Fine (1952) that necessitated an initial threat of mate in one and checks thereafter. It solved three directly, the other two, because of a change in computer facilities, by hand simulation. The search tree for position 97 has already been given in part and will be given in full under "Process" below. In position 107 the initial Queen sacrifice as well as an unexpected Bishop sacrifice were easily spotted in the service of mate. The analysis tree of position 70 is given in Figure 7 below. Note that the correct move and theme in position 70 derive from the celebrated game of Marshall's for which spectators showered the chessboard with gold coins![15] Of the two hand simulated runs position 95 required but the simple addition of a second control on the KN7 square via 1.Q-R6, while position 113 required the move 1.R-R5, which had been discovered in one of the exploratory sample variants.

[15]The program also finds the correct sequence of moves from the immortal Lewitzky-Marshall game, Breslau, 1912 (Diagram 69 in Fine, 1952); it is excluded from the count here since Lewitzky, had he not chosen to resign, could have averted mate at the cost of a piece.

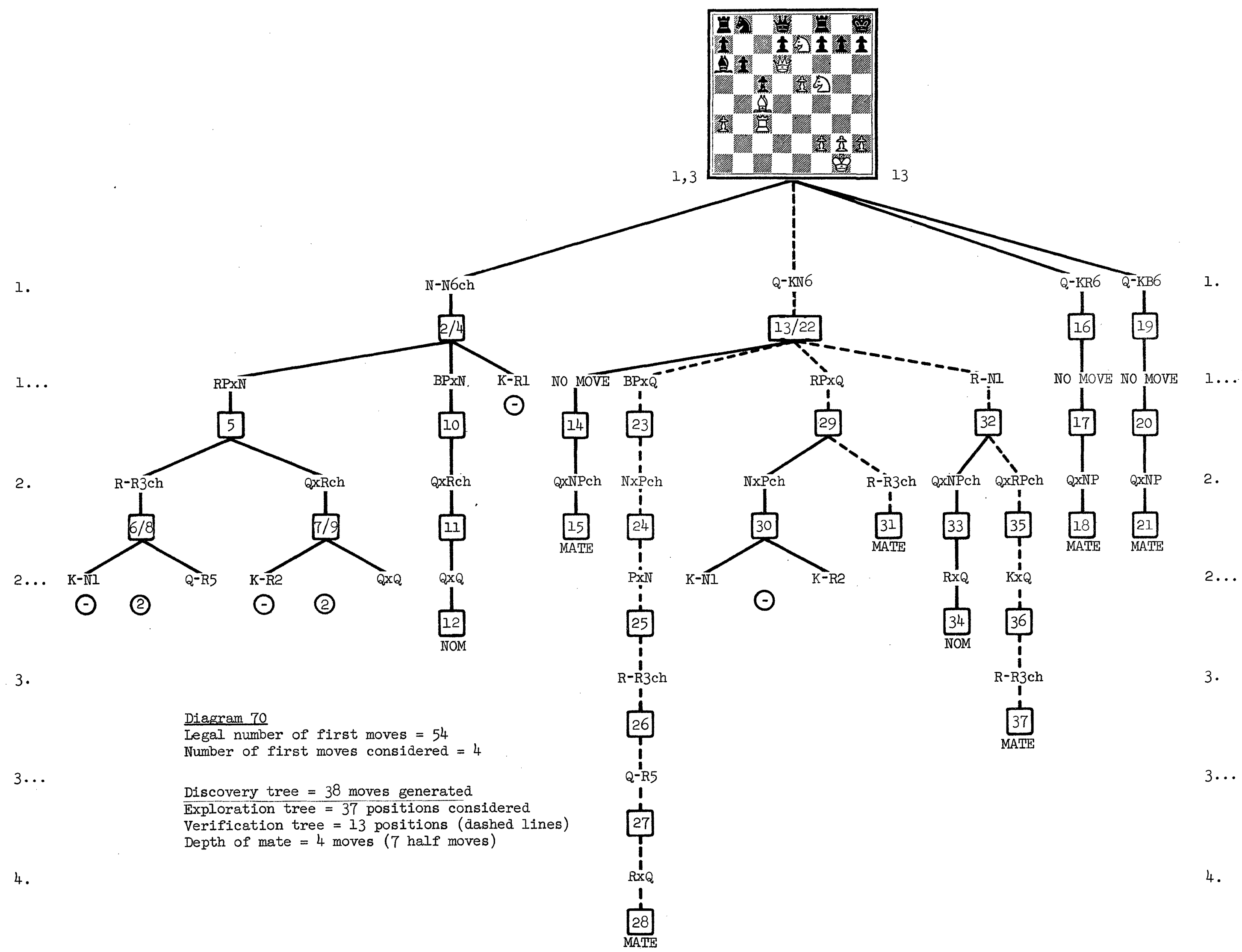


Figure 7. MATER II's Analysis Tree and Order of Search on Diagram 70, from Fine (1952)

PROCESS

General Thought Methods

At the level of general thought methods MATER shows some progressive deepening and broadening of its investigation, sample variations, and planning in a simplified problem space--all well documented structural characteristics of human thought processes (de Groot, 1965; Newell and Simon, 1964). In the exploration tree of position 70, for example (see Figure 7), the first move considered, 1.N-N6ch, is taken up two different times; each time it is elaborated to different depths and breadths. The move is rejected the first time because of a King move in reply (1...K-N1), but then is immediately reinvestigated since it is the only checking move available.

The best illustration of a sample variation comes from the analysis tree of position 97, as discussed in the last section. White samples 1.QxRPch and 1.Q-N7ch until King replies proliferate. The information gathered from the excursion is recorded and used later, as has also been discussed before.

Planning with the No-Move principle has been illustrated in position 97 too. In another position, position 70, White espies three moves in the same mating plan: 1.Q-KB6, 1.Q-N6, and 1.Q-R6, all of which threaten 2.QxNP mate (see Figure 7). The plan is a happy one, and, as can be seen by tracing through the program's course of analysis, the elaboration of 1.Q-N6 leads to a pretty mate.

Detailed Comparison

For evaluation of the model at the second and detailed level of comparison, protocols of two chess experts solving the same combination, position 97, were taken. The subjects were told there was a mate in the position. The instructions are presented in full in Appendix I and the two complete protocols in Appendix II.

The analysis trees of MATER II, Subject WT, and Subject EB follow in Figures 8, 9, and 10, respectively. Each figure contains the total tree structure of all three "players" taken together; then each player's search pattern is superimposed on this combined exploration tree. In this way the overlap and common structure of the three players can be read off directly.

What conclusions can be drawn from this kind of comparison? From the point of view of selectivity, again MATER II comes off well. There are 37 first moves in position 97. Subjects WT, EB, and MATER II taken together mention only seven of the 37. Individually WT mentions five, EB six, and MATER II three; but the three considered by MATER II (1.Q-N7ch, QxRPch, and N-N5)

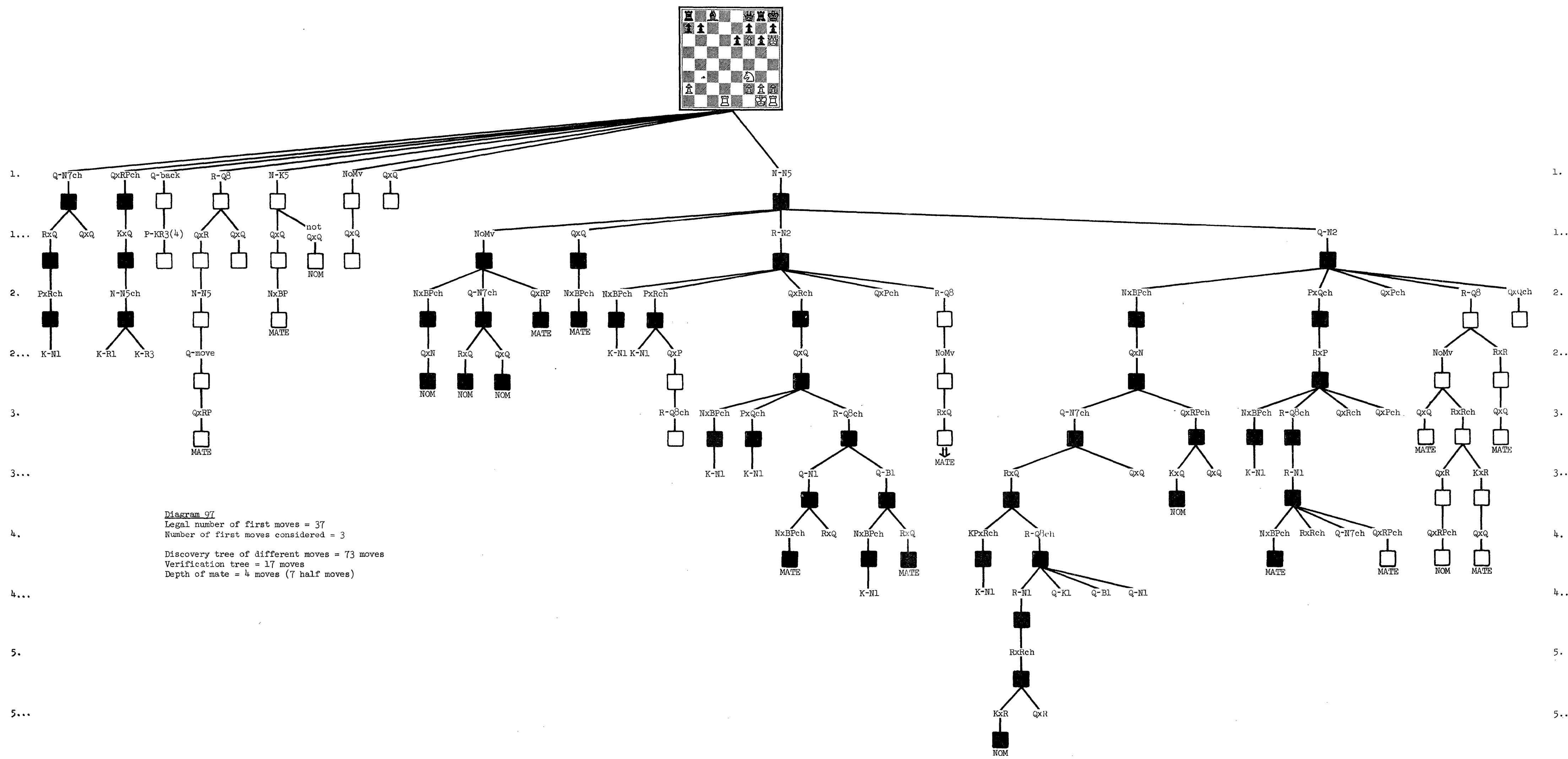


Figure 8. MATER II's Analysis Tree, blackened against the composite exploration tree of all three players, on Diagram 97, from Fine (1952)

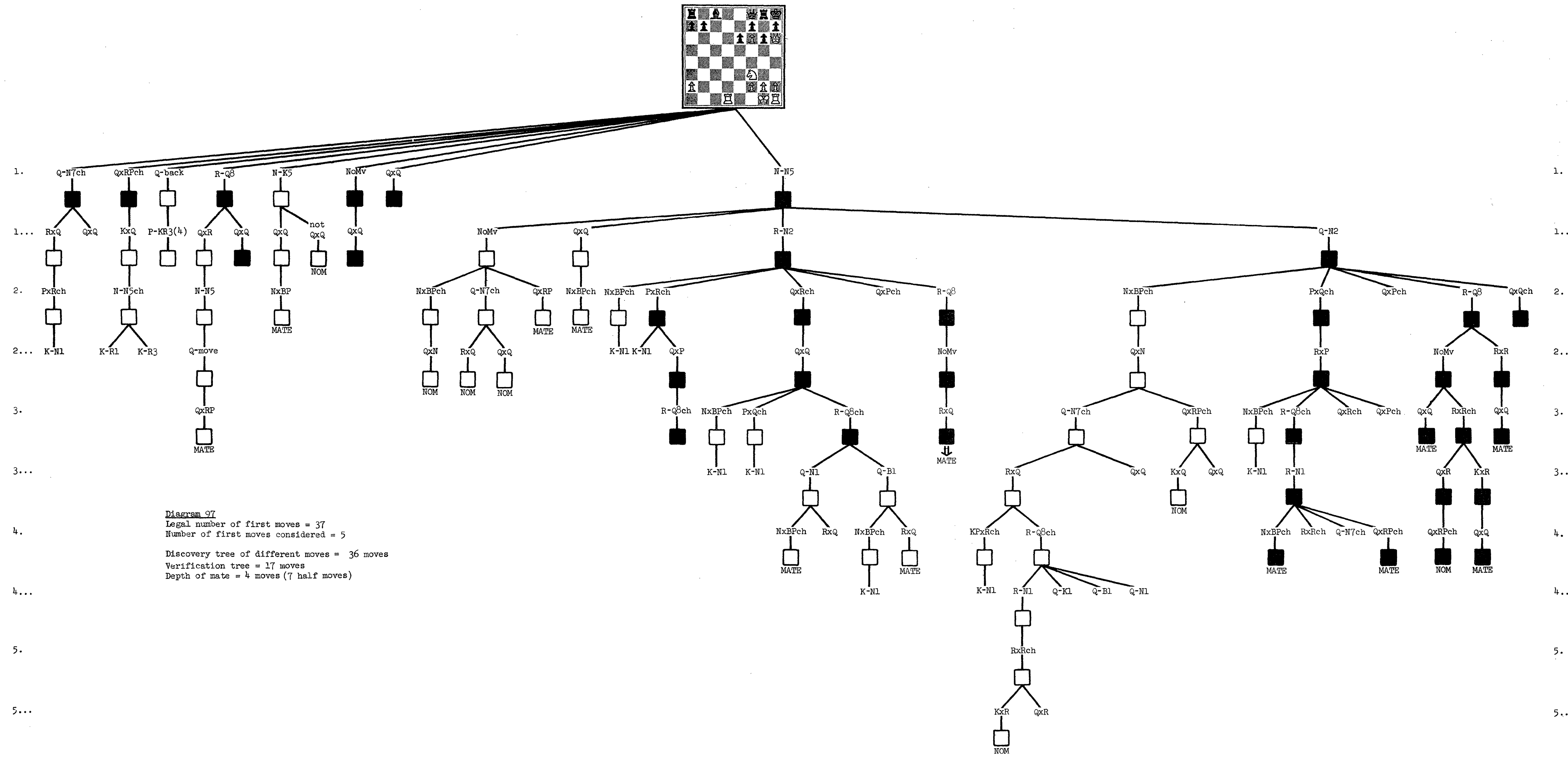


Figure 9. Subject WT's Analysis Tree, blackened against the composite exploration tree of all three players, on Diagram 97, from Fine (1952)

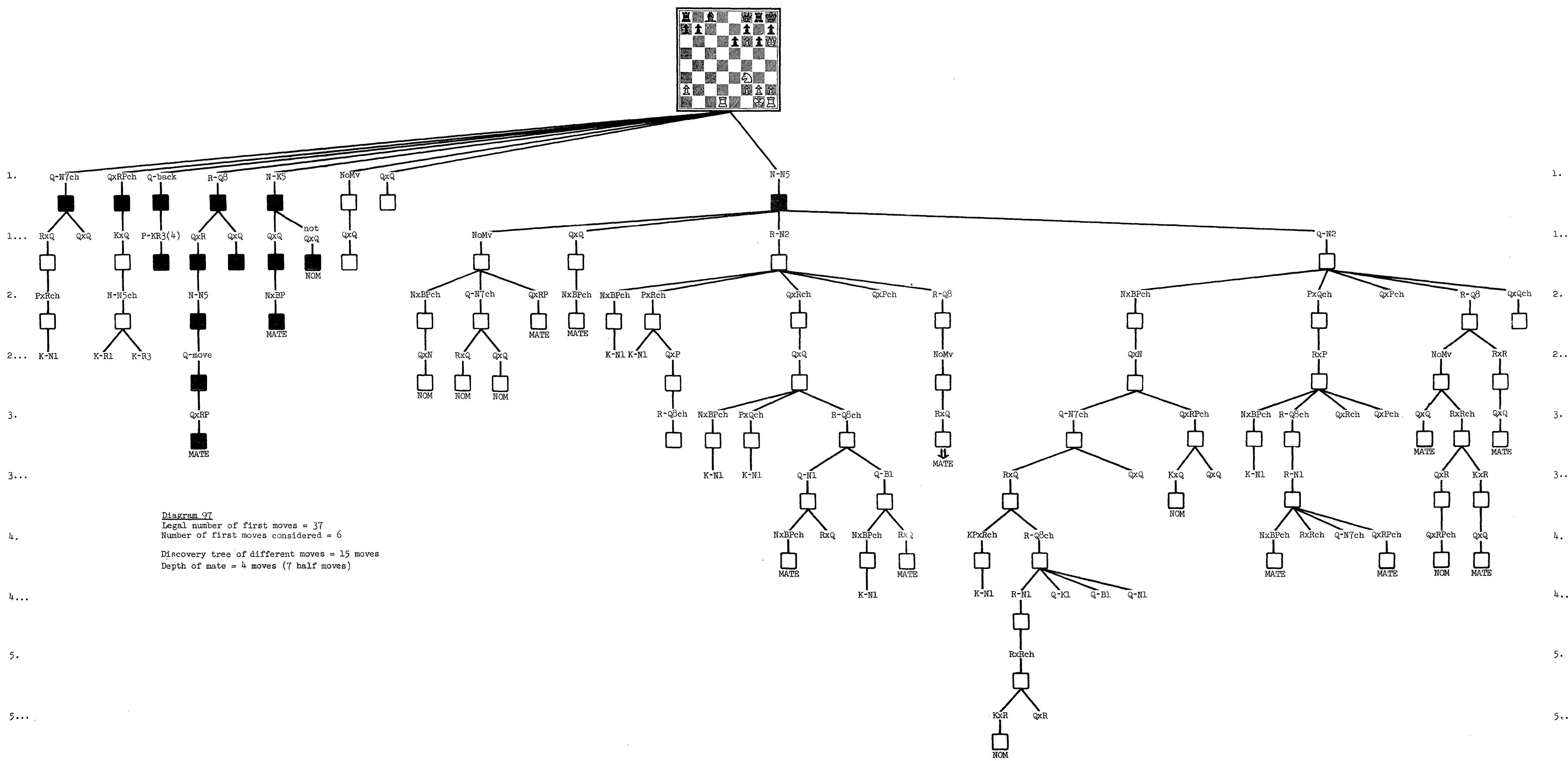


Figure 10. Subject EB's Analysis Tree, blackened against the composite exploration tree of all three players, on Diagram 97, from Fine (1952)

are the same three considered by both the other subjects, 1.R-Q8 being the humans' only other move in common. As usual the computer program is more complete in reporting what the subjects apparently reject out of hand: both subjects must have seen, for example, that neither 1.QxRPch nor 1.Q-N7ch led to checkmate though they failed to say so.

The quality of MATER's analysis in support of its move proposals also compares favorably with the human subjects': the three relevant defensive moves in reply to 1.N-N5, namely, 1...QxQ, R-N2, and Q-N2, are the only moves considered by WT or by MATER II. To each reply, both WT and MATER II deliver up a convincing proof of the mating combination.

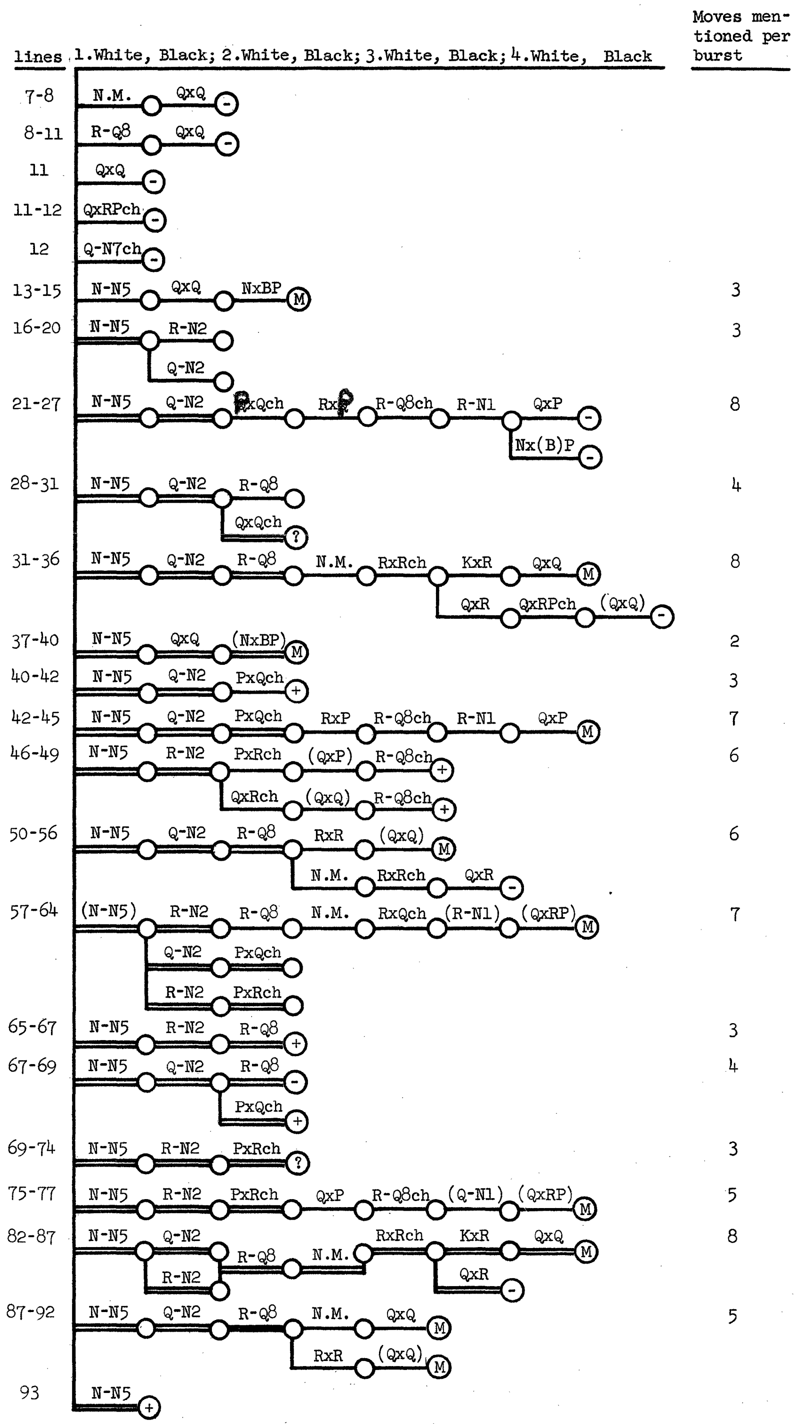
The allocation of time and effort to different parts of the analysis differs more between the human subjects than between one of the men and the machine. For example, Subject EB never mentions the bulk of the verification tree while WT devotes almost his entire time to producing a convincing and elegant proof. MATER II is like WT in that, once 1.N-N5 is discovered, the rest of the analysis is devoted to its proof.

Comparison of the search strategies of Subject WT and MATER II points to some strong dissimilarities between man and machine and, thus, brings some psychological issues to the fore. Figures 11 and 12 reveal quite different temporal patterns of search. The subject (Figure 11) searches relatively deep in each burst of activity but with little branching before returning to the initial position. This pattern agrees quite well both with the analyses of de Groot (1965) and of Newell and Simon (1964), where it has been called a "progressive deepening strategy." There are many reworkings of the same moves, some searches penetrating deeper and others broader. Variations that are retracings of previous moves are marked with double lines in the figures so that the new contributions of each burst can be read off clearly.

MATER II (Figure 12) searches both broadly and deeply with a great deal of branching in a few bursts. This pattern corresponds roughly to what has been called a Depth-First Strategy, the strategy used in the Newell, Shaw & Simon (1963a) chess playing program.

IMPLICATIONS

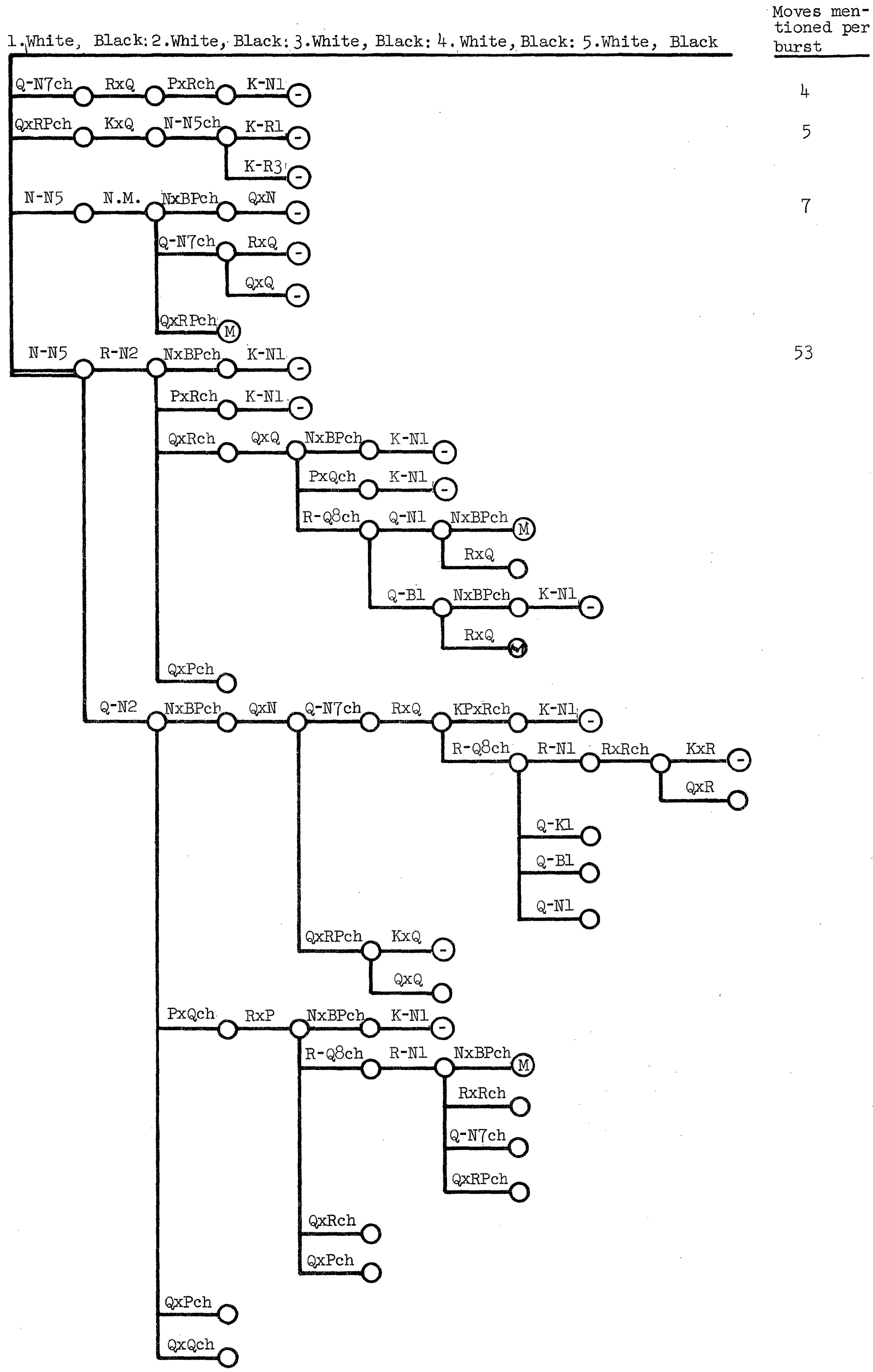
The implications of such detailed process comparison point up well the value of information-processing models in psychology. One is led to ask what sorts of mechanisms MATER would have to possess in order to produce behavior like Subject WT's. This continuous process of comparison leading to program modification, comparison on new data leading to further program modification, etc., contributes directly to theory building and has been well described by Quillian (1965, p. 10ff.)



The order of search is from left to right, then down.
The evaluations are (M) for mate, (+) for a favorable evaluation, (-) for an unfavorable one, (?) for uncertain, and ○ for unevaluated.
The numerals in the right hand column refer to the number of explicitly mentioned moves per burst of activity.
Double lines indicate the re-investigation of a sequence of moves.
Moves in parentheses are inferred since they are not mentioned explicitly in the protocol.

LINE NUMBERS REFER TO PROTOCOL 1, APPENDIX II

Figure 11. Subject WT's Search Behavior by Order of Move Generation



The order of search is from left to right, then down.
The evaluations are (M) for mate, (-) for an unfavorable evaluation, ○ for unevaluated.
The numerals in the right hand column refer to the number of explicitly mentioned moves per burst of activity.
Double lines indicate the re-investigation of a sequence of moves.

Figure 12. MATER II's Search Behavior by Order of Move Generation

Here some mechanisms are postulated which, it is hypothesized, if incorporated into the program would yield closer fits to the observed human behavior. First, there follow some hypotheses about the perceptual processes necessary to yield the requisite selectivity, and then, on the structural side, some hypotheses about uncertainty reduction and the constraints imposed by immediate memory.

Selectivity Implications

In trying to get at the issue of selectivity one is led to ask what the chess player has stored in long-term memory that enables him to know almost immediately what a new position "is all about." Suppose that chess players sort positions through a discrimination net on the basis of features of the position (see Feigenbaum, 1959) and that, furthermore, there is associated with each terminal node in the net (as well as with some of the test nodes) a class-of-moves generator specific to the position's features. This terminal node can be viewed as the player's initial classification of the position. Now, if one could construct a set of tests corresponding to features and move generators specific to them, this would seem to account for the phenomenal ability of masters and grandmasters to suggest relevant moves after the first few seconds of exposure to a new position (cf. de Groot, 1965, Section 61). Moreover, the pauses or abstractive phases in the thought process that occur typically between bursts of analytic activity in the protocols (the transitional phases of de Groot, 1965, Section 34) can be viewed as the search for new features or relations between pieces and squares; these newly perceived relations lead, in turn, to fresh sortings of the position through the discrimination net and to the generation of a new class of moves.

These then are the beginnings of a model of the chessmaster's experience, his highly developed perceptual processes and his memory organization, and would seem to be a realization of Minsky's (1963) general prescription for pattern recognition programs: "A resourceful machine must classify problem situations into categories associated with the domains of effectiveness of the machine's different methods" (Minsky, 1963, p. 411).

A look at the first two paragraphs of Subject WT's protocol suggests that his discrimination net and associated methods may look something like the following, Figure 13:

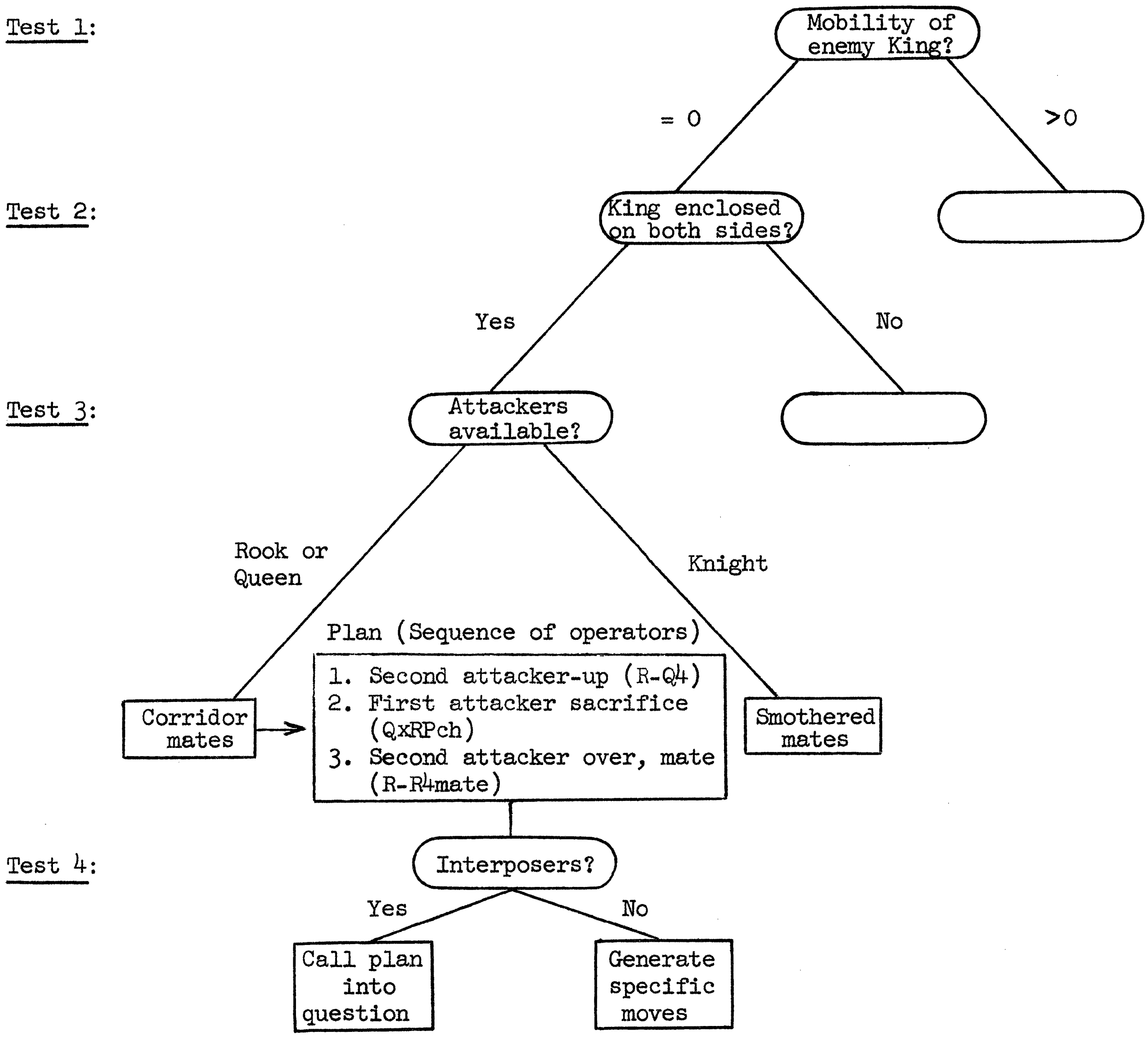


Figure 13. Partial Discrimination Net Sorted to Classes of Checkmates

Structural Implications

Two hypotheses stem directly from the phenomenon of progressive deepening.

(1) First, viewing the analysis process as the construction of a "subjectively convincing argument" (cf. de Groot, 1965, Section 39) leads to the hypothesis that the fraction of the total analysis devoted to the investigation and reinvestigation of particular base moves is a function of:

(a) the favorableness (or unfavorableness) of a move's evaluation in comparison with the other considerable alternatives; and

(b) the certitude of tentativeness with which that evaluation is held, which is primarily a function of the number of considerable alternatives available at each level in the tree.[16]

The more favorable an initial move's evaluation, the more time and effort one is willing to exert to confirm that evaluation--that is, the surer one wants to be that he has not overlooked a strong countermove of the opponent's or a better own-move for himself. In particular, once a favorite has been established (like 1.N-N5 in position 97) the rest of the analysis can be viewed as reducing the uncertainty ensuing from the unknown consequences of alternative own-moves and especially countermoves at each node in depth.

The branchier a path through the analysis tree is perceived to be, the more tentative any evaluation will be and the more a player will employ the method of progressive deepening to go back and pick up the neglected alternatives, to reduce the attendant uncertainty. No rigorous test of either hypothesis will be offered here, but a pass through the protocol does offer the following support:

> Subject WT discovers the move 1.N-N5 in line 13 and, seeing right away that 1.N-N5, QxQ; 2.NxBP is mate, devotes the rest of his time and effort to 1.N-N5. Because of the enormous favorableness of the evaluation, one would predict an immediate reinvestigation of 1.N-N5 and a search for counter strengthening. Lines 16-20 are just this: "Now Black is forced to play [either] 1...R-N2 or 1...Q-N2," the first disjunctive goal-setting and indicator of the branchiness of the path he is about to pursue.

---

[16]This whole formulation is in some respects similar to traditional balance theory and decision theory approaches, which emphasize just the choice behavior aspects of problem solving (see Section IV on "Building the Tree").

In the burst of lines 21-27, the disjunctive goal-setting, "then either 4.QxP mate or 4.NxP" is highly uncertain with respect to own-moves and, predictably, precipitates a search for own-strengthening moves: "how about (1.N-N5, Q-N2;) 2.R-Q8?" coupled with uncertainty about another available alternative, 2.QxQch (line 30).

Lines 31-36 reveal a disjunctive goal-setting at 3...KxR or 3...QxR. The former leads to mate, the latter to a negative evaluation. This is quickly followed by a search for own-strengthening and, indeed, at this point WT tries to reassure himself with a new reason that 1.N-N5 is the key move after all: 1.N-N5 prevents ...QxQ "while at the same time getting another piece into the action" (lines 39-40).

As though to gather his courage further, WT falls back on moves that are absolutely forcing: 1.N-N5, Q-N2; 2.PxQch forces 2...RxP. This continuation leads to a mate (lines 42-45) and WT once again looks for a counter-strengthening. Not until lines 54-56 does he encounter another negative evaluation, whereupon he seeks another own-strengthening (lines 57-64).

WT's discovery of a new relationship in line 77--that the Black Queen is pinned--does seem to reassure him of the soundness of his favored 2.R-Q8. One final negative evaluation, however, in lines 83-87, leads to a final own-strengthening and a self-assured choice-of-move decision in line 93. "I'll play 1.N-N5."

(2) Second, it is hypothesized that the constraints on immediate memory limit the depth and breadth of analysis a player can carry out before he must return to some base position, usually the position on the board in front of him.

Some recent studies of Yntema and Mueser (1960, 1962) and Yntema (1963) focus on memory load and "how well a person can follow a changing situation, keeping track of it in his head" (Yntema, 1963, p. 7). In one variant of their experimental paradigm a subject receives a sequence of messages, each message changing the current state of one variable. The choice of which variable will change on each trial is randomly determined. Periodically the subject is queried as to the current state of one of the variables. The fraction of correct responses is the dependent variable and is taken as the measure of the subject's memory load capacity. These investigators found that the fraction of questions answered correctly decreases with an increase in the number of variables, as expected, but even with so few variables as three, subjects miss on the average of 30% of the questions. In short, a human's storage capacity for changing information is far from perfect, even when the number of variables changing is small.

The chess player in his task of mental analysis must also keep track of the current states of many "variables,"many of which change--unlike the one-at-a-time changes of Yntema and Mueser--with the receipt of each new "message," i.e., with the consideration of each move that deepens or broadens the investigation. How does the making of a move change a position? Simultaneously, a move establishes a new set of functions or relationships between pieces and/or squares while relinquishing an old set. For example, in position 97, the move 1.N-N5 simultaneously attacks the Black Pawns on K6, KB7, and KR7; occupies KN5, etc.; while it relinquishes the defense of the two center squares Q4 and K5 and of the Pawn on KR2. This same move, 1.N-N5, also ignores or leaves unchanged one of the functions of the Black Queen: namely, the threatened capture of the Queen on KR6. The "states" of these and other functions are exactly what the chess player must keep track of when he analyzes a move. In addition, the chess player is charged with perceiving or discovering the new relationships each move effects as well as judging which ones are criterial and which irrelevant to the position. No wonder the chess player begins anew from the position in front of him so many times.

There is another facet of this phenomenon, namely, progressive deepening as rehearsal. This may allow the player to fixate in relatively long-term memory some new base position--other than the one on the board in front of him--from which he can conduct his analysis to depths and breadths that were formerly prohibitive because of the constraints on immediate memory. Indeed, Melton (1963), in discussing the storage of verbal material, views the frequency of repetition or rehearsals as the important independent variable, "chunking" (Miller, 1956) as the important intervening variable, and the slope of the retention curve as the important dependent variable. Whether rehearsal serves to move items from short-term into long-term memory or to chunk items such that the span of short-term memory is effectively expanded is not at issue here. The point is simply that going over the same main line several times does in fact enable the chess player to probe deeper and broader with fewer errors of analysis; whether new positions are being fixated or paths chunked or both is a question for further research.

The only test of these hypotheses comes from a simple count of the number of moves[17] Subject WT mentions in each of his 17 investigations of 1.N-N5. The mean number of moves per burst is 5 while the range is from 2 to 8. Both the mean and the upper limit of the range fall within the bounds of the magical number 7 ± 2 (Miller, 1956). Two further observations from within the protocol support the hypotheses offered here: first, WT makes

[17] One should prefer to use changing <u>relations</u> as the unit of immediate memory but until some means is worked out for measuring which relations the subject is attending to, the move is invoked as a roughly comparable functional unit.

an error in analysis in lines 25-26 just near the limit of the immediate memory span and in an early enough stage of the analysis of 1.N-N5, Q-N2 that this path is unlikely to have yet been chunked. Second, WT imposes a memory strain on himself when from lines 50 onwards he tries to work out the consequences of both 1...R-N2 and 1...Q-N2 in one theme. He constrains himself to analyze the position with both moves in mind; no wonder he finds his "variables are floating away" (line 87)!

In conclusion, MATER's power stems from its ability to generate a small selective set of moves that merit investigation. Since most of the earlier chess programs (see the review in Newell, Shaw & Simon, 1963a; and Kotok, 1962) spent their analysis time processing the wrong moves, it would seem that MATER II's two major mechanisms for generating relevant moves--its reliance on the sample variations and on the control of key squares--warrant further research. MATER II's major weakness, on the other hand, lies in its poorly organized search strategy for using its selectivity at all points in the analysis process.

On the horizon, proposals have been made for strengthening the program's perceptual capabilities as well as radically altering its search strategy. A discrimination net that sorts positions on the basis of their features and proposes moves according to the terminal category would seem to account for much of the master's phenomenal perceptual ability. A Progressive Deepening Search Strategy sensitive to the uncertainties of the position in terms of the number of available considerable moves and the tentativeness of the evaluations made, and an immediate memory limited to the usual 7 ± 2 chunks of information have all been proposed as necessary mechanisms for achieving closer fits to the human chess player's information-processing abilities.

REFERENCES

Bernstein, A. & Roberts, M. DeV. Computer vs. chess player. Scientific American, 1958, 198, 96-105.

Botvinnik, M. M. One Hundred Selected Games. (Translated by S. Garry.) New York: Dover Press, 1960.

Bruner, J. S., Goodnow, J. J. & Austin, G. A. A Study of Thinking. New York: John Wiley & Sons, Inc., 1956.

de Groot, A. D. Thought and Choice in Chess (rev. transl.). The Hague: Mouton & Co., 1965.

Duncker, K. On problem-solving. (Translated by L. S. Lees from the 1935 original.) Psychol. Monogr., 1945, 58, No. 270.

Feigenbaum, E. A. An information processing theory of verbal learning. RAND Corp. Paper No. P-1817. Santa Monica, Calif.: The RAND Corp., 1959.

Fine, R. The Middle Game in Chess. New York: David McKay, 1952.

Kotok, A. A chess playing program for the IBM 7090. Unpublished bachelor's thesis. Cambridge, Mass.: Massachusetts Institute of Technology, 1962.

Leavitt, H. J. Managerial Psychology (2nd ed.). Chicago:Univ. of Chicago Press, 1964.

Luce, D. & Raiffa, H. Games and Decisions. New York: John Wiley & Sons, Inc., 1957.

Maier, N. R. F. Screening solutions to upgrade quality: a new approach to problem solving under conditions of uncertainty. J. of Psychology, 1960, 49, 217-31.

Melton, A. W. Implications of short-term memory for a general theory of memory. J. Verbal Learning & Verbal Behavior, 1963, 2, 1-21.

Miller, G. A. The magical number seven, plus or minus two: Some limits on our capacity for processing information. Psychol. Rev., 1956, 63, 81-97.

Miller, G. A., Galanter, E. & Pribram, K. H. Plans and the Structure of Behavior. New York: H. Holt & Co., 1960.

Minsky, M. Steps toward artificial intelligence. In E. A. Feigenbaum & J. Feldman (Eds.), Computers and Thought. New York: McGraw-Hill, 1963. Pp. 406-50.

Myron, M. S.& May, W. H. A note on serendipity, aesthetics, and problem solving. Behavioral Science, 1963, 8, 242-43.

Newell, A. Some problems of basic organization in problem-solving programs. In M. Yovitts, G. T. Jacobi & G. D. Goldstein (Eds.), Self-Organizing Systems. New York: Spartan, 1962. Pp. 393-423.

Newell, A. (Ed.) Information Processing Language-V Manual. Englewood Cliffs, N. J.: Prentice-Hall, Inc., 1961.

Newell, A. & Ernst, G. The search for generality. In W. A. Kalenich (Ed.), Proceedings of IFIPS Congress 65, Vol. 1. Washington, D. C.: Spartan Books, Inc., 1965. Pp. 17-24.

Newell, A. & Prasad, N. S. IPL-V chess position program. Unpublished working paper. Carnegie Institute of Technology, 1963.

Newell, A., Shaw, J. C. & Simon, H. A. The processes of creative thinking. In H. E. Gruber, G. Terrell & M. Wertheimer (Eds.), Contemporary Approaches to Creative Thinking. New York: Atherton Press, 1962. Pp. 63-119.

Newell, A., Shaw, J. C. & Simon, H. A. Chess-playing programs and the problem of complexity. In E. A. Feigenbaum & J. Feldman (Eds.), Computers and Thought. New York: McGraw-Hill, 1963a. Pp. 39-70.

Newell, A., Shaw, J. C. & Simon, H. A. Empirical explorations with the logic theory machine: a case study in heuristics. In E. A. Feigenbaum & J. Feldman (Eds.), Computers and Thought. New York: McGraw-Hill, 1963b, Pp. 109-33.

Newell, A. & Simon, H. A. Computer simulation of human thinking. Science. 1961, 134, 2011-17.

Newell, A. & Simon, H. A. An example of human chess play in the light of chess playing programs. Unpublished working paper. Carnegie Institute of Technology, 1964.

Quillian, M. R. Word concepts: a theory and simulation of some basic semantic capabilities. CIP working paper #79. Carnegie Institute of Technology, 1965.

Reitman, W. R. Information-processing models in psychology. Science, 1964, 144, 1192-96.

Reitman, W. R. Cognition and Thought: An Information Processing Approach. New York: John Wiley & Sons, Inc., in press.

Samuel, A. L. Some studies in machine learning using the game of checkers. In E. A. Feigenbaum & J. Feldman (Eds.), Computers and Thought. New York: McGraw-Hill, 1963. Pp. 71-105.

Simon, H. A. & Newell, A. Models: Their uses and limitations. In L. D. White (Ed.), The State of the Social Sciences. Chicago: Univ. of Chicago Press, 1956. Pp. 66-83.

Simon, H. A. & Simon, P. A. Trial and error search in solving difficult problems: evidence from the game of chess. Behavioral Science, 1962, 7, 425-29.

Yntema, D. B. Keeping track of several things at once. Human Factors, 1963, 5, 7-17.

Yntema, D. B. & Mueser, G. E. Remembering the present states of a number of variables. J. Exp. Psychol., 1960, 60, 18-22.

Yntema, D. B. & Mueser, G. E. Keeping track of variables that have few or many states. J. Exp. Psychol., 1962, 63, 391-95.

APPENDIX I

CHESS EXPERIMENT INSTRUCTIONS

"When I ask you to, please turn around and look at the chess position on the board in front of you. The position was taken from a real tournament game. Assume that it is your move in that tournament situation; I will be sitting across the board from you as though I were your opponent. When you decide on a move, please actually make it on the board.

"This experiment contains two important deviations from tournament conditions: First, and very important, is that I want you to think out loud as you analyze the position, to try to say everything that comes into your head, no matter how foolish or pointless it may seem at the time. The main purpose of this experiment is to get as full an account as possible of your thought processes as you are analyzing the position, and not to "test" you as a chess player. If during the course of the experiment you should fall silent for long periods, I may prod you by asking you what you are thinking, to remind you that you should be talking as much as you can.

"The second deviation is that I can tell you that there is a checkmating combination to be found in the position. In a tournament game, obviously, you would not know this for sure until you had found and demonstrated the mate for yourself. If for some reason you decide you're not going to find the mate (or I determine that you've been at it long enough to be in time pressure) then just make whatever move you would make were you playing a tournament game.

"Do you have any questions?

"We'll start with a simple practice position to make sure you have the idea. If there are no questions then, we can proceed with the three principle positions."

APPENDIX II

TWO PROTOCOLS OF DIAGRAM 97 FROM FINE (1952)

1. Subject: WT
   Exp.: GWB
   Date: July 2, 1964
   Time: 6 min., 35 sec.

"All right, I notice that the Black King has very few freedoms, essentially no moves. I immediately think of the weaknesses on the Rook file, and that possibly I may be able to bring a Rook around onto the Rook file and sacrifice my Queen although I notice I must remember that the Queen, Black Queen can interpose, so I have to allow for this possibility.

"Now I look for an immediate threat: I notice that Black is going to play Queen takes Queen (QxQ) if I do nothing. Uhm, I am considering Rook to Queen 8 (1.R-Q8) because I notice that this, I, I think that this may prevent a mate or prevent Queen takes Queen (QxQ), but it does not immediately. Ah, I consider Queen takes Queen (1.QxQ), Queen takes Rook Pawn check (1.QxRPch), Queen to Knight 7 check (1.Q-N7ch). I consider, uhm, I see a smothered mate if I play Knight to Knight 5 (1.N-N5) because Queen takes Queen (1...QxQ) is followed by Knight takes Pawn (2.NxBP) mate.

"Now, Knight to Knight 5 (1.N-N5) threatens mate; it does not seem to be immediately avoidab, a way Black can avoid it. Now, Black is forced to play Rook or Queen to Knight 2 (1...R- or Q-N2) after Knight to Knight 5 (1.N-N5), and I can probably get a mate using the Rook to Queen 8 (R-Q8) motif at the right time or the smothered mate. Now let's see the right order: if I play Knight to Knight 5 (1.N-N5), and Black plays Queen to Knight 2 (1...Q-N2), I can play Pawn takes Queen check (2.PxQch); Rook takes Queen (sic; 2...RxP), Rook to Queen 8 check (3.R-Q8ch), Rook to Knight 2 (sic; 3...R-N1), and then either Queen takes Pawn (4.QxP) mate or Knight takes Pawn (4.NxP), ah, no the Queen is gone, I can play Knight takes Pawn (4.NxP). No, that doesn't work.

"Knight to Knight 5 (1.N-N5), Queen to Knight 2 (1...Q-N2), ah, how about Rook to Queen 8 (2.R-Q8)? I'm not sure whether the original move of Queen to, to ah, Queen takes Queen check (2.QxQch) is good, but let's see what this looks like: Knight to Knight 5 (1.N-N5), Queen to Knight 2 (1...Q-N2), Rook to Queen 8 (2.R-Q8); this threatens ... what does it threaten? Ah, Rook takes Rook check (3.RxRch), Rook takes Rook check (3.RxRch), then if King takes Rook (3...KxR), Queen

takes Queen (4.QxQ) mate; if Queen takes Rook (3...QxR), Queen takes (4.QxRPch), no then the Queen is still on that square.

"Well, I'm certain that Knight to Knight 5 (1. N-N5) must be the key because Queen to, Queen takes Queen (...QxQ) is threatened first. Knight takes Knight (sic; 1.N-N5) prevents it while at the same time getting another piece into the action. Now, Knight to Knight 5 (1.N-N5), Queen to Knight 2 (1...Q-N2), I can play Pawn takes Queen check (2.PxQch)--that's it. Knight to Knight 5 (1. N-N5), Queen to Queen 2 (sic; 1...Q-N2), Pawn takes Queen check (2.PxQch) forces Rook takes Pawn (2...RxP), Rook to Knight 8, Queen 8 check (3.R-Q8ch), Rook to Knight 2 (sic; 3...R-N1), Queen takes Pawn (4.QxP) mate.

"If Knight there, if Knight to Knight 5 (1.N-N5) and if Rook to Knight 2 (1...R-N2), well then, I can either play--let's see now if Pawn takes Rook (2.PxR), Queen takes Rook there check (2.QxRch)--then I can play Rook to Queen 8 (3.R-Q8).

"Let's see if I can get this in one theme, instead of scattering it. Knight to Knight 5 (1.N-N5), Queen to Knight 2 (1...Q-N2), Rook there (2.R-Q8) doesn't work because of, after Rook takes, I mean he can't play Rook takes Rook (2...RxR) because that takes the guard from the Queen but I'm not threatening anything, I can't play Rook takes Rook (3.RxR) 'cause just Queen takes Rook (3...QxR), then I don't have enough threats because the Queen guards all three key squares. However, on the line where he plays Knight to Queen 2 (sic; 1...R-N2), Rook to Queen 8 (2.R-Q8) is a threat because Rook takes Queen (3.RxQ), ah, leads to mate because there's nothing guarding the Queen. And if he plays Queen up (1...Q-N2), the difference is the fact that on Pawn takes Queen (2.PxQ), he can't move his King. Only if he'd played Rook there I could have played Pawn takes Queen (sic; 2.PxRch) too. I'm just trying to see the differences in the two lines, why it, the same move doesn't work for both, it ply-two.

"Knight to Knight 5 (1.N-N5), Rook to Knight 2 (1...R-N2), Rook to Knight 8 (sic; 2.R-Q8) wins, I think. I'll have to go back and check when I finish. Knight to Knight 5 (1.N-N5), Queen to Knight 2 (1...Q-N2), Rook to Knight, Rook to Queen 8 (2.R-Q8) doesn't seem to win, but Pawn takes Queen check (2.PxQch) does win. Now Knight to Knight 5 (1.N-N5), Rook to Knight 2 (1...R-N2), Pawn takes Rook check (2.PxRch): why is that different? Because in the end it is the Queen that is at Knight 1 instead of the Rook. And the Queen can be guarding, uhm, the Pawn at Rook 2 whereas the Rook cannot. Is that right?

"Knight to Knight 5 (1.N-N5), Rook to Knight 2 (1...R-N2), Pawn takes Rook (2.PxRch), Queen takes (2...QxP), check (3.R-Q8ch), Rook .... No,

it's (the Black Queen) pinned, either of them should be equivalent. All right, so the move which I prefer because it, it works the same in either case would be the Rook to Queen 8 (2.R-Q8) variation instead of the Pawn takes Rook (2.PxRch), also for aesthetic reasons: just gobbling material.

"So I play Knight to Knight 5 (1.N-N5); on either interposition I play Rook to Queen 8 (2.R-Q8). Now the immediate threat is ... oh, now did I, did I mess this up? Rook takes, the immediate threat is Rook takes Rook check (3.RxRch) if the Queen is there, King takes Rook (3...KxR), Queen takes Queen (4.QxQ) mate. If Queen takes Rook (3...QxR) .... I got it backwards again; my variables are floating away. Knight to Knight 5 (1.N-N5), Queen to Knight 2 (1...Q-N2), if Rook there (2.R-Q8), that's not a threat, oh yes, it is a threat: it's threatening Queen takes Queen (3.QxQ) mate 'cause the Rook is pinned. If Rook takes (2...RxR) ... they're so many mates here it doesn't seem to matter which I do.

"I'll play Knight to Knight 5 (1.N-N5)."

E: "All right."

S: "That's obviously the mate."

* * * * * * *

2. Subject: EB
   Exp.: GWB
   Date: July 1, 1964
   Time: 1 min., 2 sec. (lines 1-9)

"Let's see, ah, first thought is to ....

"Lemme first try and get mate on this square, but the Queen is under attack, so therefore I would have to ....

"Now if I retreated, he pushed the Pawn up ....

"Let's see, supposing ....

"Can't lure the Queen away.

"Oh, here.

"Uhm, I play Rook to the eighth, you take it off, then I play Knight here. Now, Knight here threatens mate. O.K., I would move here. (1.N-N5)"

E: "Oh, all right, ah ..."

S: "Is that?"

E: "Yeah, you can stop."

S: (Retrospection) "I wan, I wanted to try to mate on King's Rook 7 (1.QxRPch), ah, because this square (KN7) is protected too many times. I, I just immediately rejected mating on, on Queen, on King's Knight 7, so at first I, ah, thought of trying to get the Queen away by playing Rook here (1.R-Q8), and then if, er, Black were kind enough to take the Rook (1...QxR) I would play Knight here (2.N-N5), and before he could get back I would mate him (on KR7). But then I saw that after Rook here (1.R-Q8) it would just be Queen takes Queen (1...QxQ). Then, ah, since I needed the Knight up here, I, ah, the thought occurred to me that I could mate on this square (KB7) also, and I thought of Knight to King 5 (1.N-K5); then if Queen takes Queen (1...QxQ), I'd mate; but of course he just wouldn't play Queen takes Queen (1...QxQ), so then I thought of Knight to King's Knight 5 (<u>1.N-N5</u>)."

DOCUMENT CONTROL DATA - R&D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1. ORIGINATING ACTIVITY *(Corporate author)*

System Development Corporation,
Santa Monica, California

2a. REPORT SECURITY CLASSIFICATION
Unclassified

2b. GROUP

3. REPORT TITLE

REPORT ON A MATING COMBINATIONS PROGRAMS

4. DESCRIPTIVE NOTES *(Type of report and inclusive dates)*

5. AUTHOR(S) *(Last name, first name, initial)*

Baylor, G. W.

6. REPORT DATE
4 June 1965

7a. TOTAL NO. OF PAGES
64

7b. NO. OF REFS

8a. CONTRACT OR GRANT NO. Not Applicable

b. PROJECT NO.

c.

d.

9a. ORIGINATOR'S REPORT NUMBER(S)

SP-2150

9b. OTHER REPORT NO(S) *(Any other numbers that may be assigned this report)*

10. AVAILABILITY/LIMITATION NOTICES

This document has been cleared for open publication and may be disseminated by the Clearing House for Federal Scientific and Technical Information.

11. SUPPLEMENTARY NOTES

12. SPONSORING MILITARY ACTIVITY

13. ABSTRACT

This paper describes a computer program that analyzes chess positions that contain checkmating combinations. Over the range of positions the program can handle, the program is viewed as a psychological model of human problem-solving behavior. The model has a set of mechanisms for generating a small, highly selective set of moves for analysis and a search strategy for conducting the chess analysis. These parallel the human chess player's search behavior on several points: particularly on the quality of the moves considered and on the heuristics and stop-rules for keeping the ever-branching tree of move possibilities within manageable limits. Some specific hypotheses about the chessmaster's perceptual abilities are offered to account for the quality of the moves that come under consideration while some hypotheses about uncertainty reduction and the nature of the constraints imposed by immediate memory are suggested to account for some of the structural facets of the thought process. These derive from the detailed process comparison of a human's behavior with the model's. On a performance measure the program has discovered some of the most sparkling combinations in the chess literature, in positions requiring analyses ranging from two to eight moves in depth. (author)

DD FORM 1 JAN 64 1473

14. KEY WORDS	LINK A		LINK B		LINK C	
	ROLE	WT	ROLE	WT	ROLE	WT
Computer Program Chess Checkmating Combinations. Mating Human behavior						

MODIFICATION TO:

SP-2150, "Report on a Mating Combinations Program," dated 4 June 1965.

APPROVED

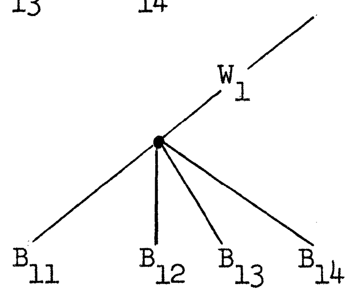
System Development Corporation/2500 Colorado Ave./Santa Monica, California

CURRENT MODIFICATION

Modified Pages	Notes and Filing Instructions
2A	Insert new page 2A dated 18 August 1965.
49	Replace page 49 with new page 49 dated 18 August 1965.

ERRATA*

Modified Pages	Notes and Filing Instructions
13	Third paragraph, line 6, change "is" to "in."
23	Connect B_{13} and B_{14} to the missing node from W_1 as shown:



Modified Pages	Notes and Filing Instructions
43	Second paragraph, change "Will" to "With."
60	Opposite lines 21-27, change QxQch to PxQch and change RxQ to RxP.
60	Under legend, add the following sentence: "Line numbers refer to Protocol 1, Appendix II."
61	Seven lines down from numeral 53, change |RxQ○ to |RxQⓂ.
69	Under reference, de Groot, A. D., change <u>Thought and Choices in Chess</u> to <u>Thought and Choice in Chess.</u>

*ERRATA modifications are to be entered by hand.

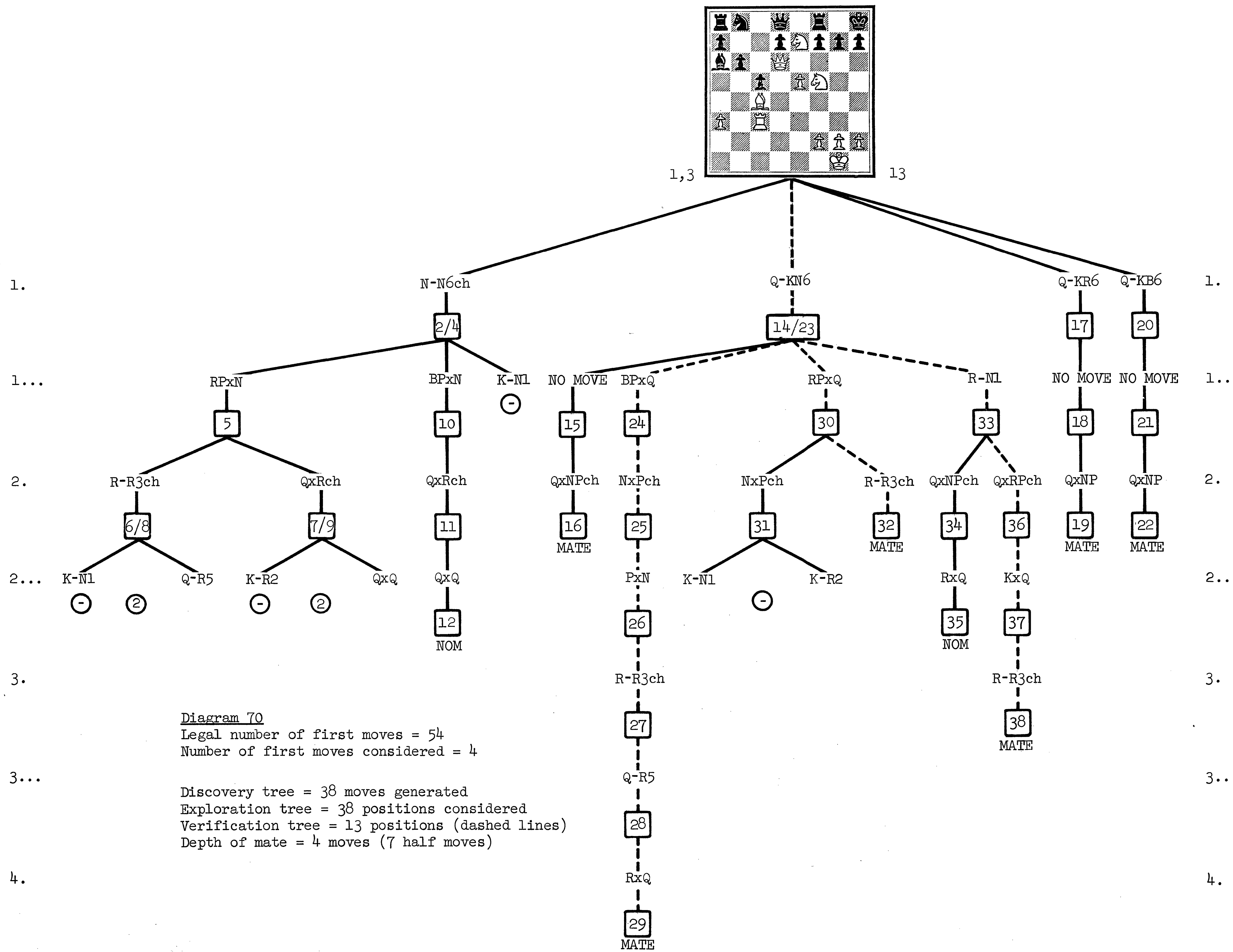


Diagram 70
Legal number of first moves = 54
Number of first moves considered = 4

Discovery tree = 38 moves generated
Exploration tree = 38 positions considered
Verification tree = 13 positions (dashed lines)
Depth of mate = 4 moves (7 half moves)

Figure 7. MATER II's Analysis Tree and Order of Search on Diagram 70, from Fine (1952)

SDC

System Development Corporation / 2500 Colorado Avenue / Santa Monica, California