Tony Leal

SRC

# SYSTEMS RESEARCH CENTER
## CASE INSTITUTE OF TECHNOLOGY

THE CONCEPT OF STRATEGY
AND ITS APPLICATION TO
THREE DIMENSIONAL TIC-TAC-TOE

Ronald L. Citrenbaum

SRC 72-A-65-26

*Systems Theory Group*

# THE CONCEPT OF STRATEGY AND ITS APPLICATION TO THREE DIMENSIONAL TIC-TAC-TOE

Ronald L. Citrenbaum

SRC 72-A-65-26

# THE CONCEPT OF STRATEGY
# AND ITS APPLICATION TO
# THREE DIMENSIONAL TIC-TAC-TOE

A Thesis Submitted to
Case Institute of Technology
In Partial Fulfillment of the Requirements
for the Degree of
Master of Science in Electrical Engineering

by
Ronald L. Citrenbaum

1965

ABSTRACT

The game of three-dimensional Tic-Tac-Toe ("Qubic") has been analyzed from a point of view distinct from the normal minimax-intermediate-evaluation standpoint. Certain pruning strategies have been developed based on the concept of forcing moves, confining attention to planes inside the board. The pruning is drastic enough to enable analysis to the end of the game, reducing the need for intermediate evaluation to a minimum.

Two computer programs, one on the PDP-4 (with a 340 D scope display unit) and one on the UNIVAC 1107, have been described. The level of sophistication of the two programs is different (based on two different levels of strategies discussed in the paper), and a comparison of games played by the two makes an interesting study.

ACKNOWLEDGEMENTS

The author wishes to acknowledge the wise counsel and helpful suggestions of his thesis advisor, Professor R.B. Banerji. In addition, the author is grateful to Dr. T.G. Windeknecht for several enlightening discussions and to Dr. R.L. Wigington, Mr. N.A. Ball, Mr. H.Q. Foster, and Mr. J. Epstein for help in programming the PDP-4 computer.

The research was supported by Grant No. AF-AFOSR-125-63 from the U.S. Air Force Office of Scientific Research and the National Science Foundation (Grant No. GP-658) for a project entitled "Research in Artificial Intelligence".

TABLE OF CONTENTS

## 1. INTRODUCTION

In the summer of 1962, a checker playing computer defeated one of the nations foremost players. Amazingly, the machine was programmed by a man who knew comparatively little about the complex strategies of the game.

This and similar achievements fall into the realm of artificial intelligence, a promising field in which computer programs are being constructed to exhibit behavior that may be called "intelligent behavior" when it is observed in human beings. Such research, it is hoped, will lead to the development of techniques and devices to relieve human beings of various complex tasks.

Of the many different research paths being explored in the field of artificial intelligence, perhaps the most fascinating to the researcher is game playing. Game environments are useful in studying the nature and structure of complex problem solving processes, and, in addition, provide man the amusing opportunity to challenge his wits against a machine. The relatively highly regular and well defined problem environments characteristic of games in general provide an excellent foundation for the use of intelligence and symbolic reasoning skills. Symbolism rather than numerics come directly under study.

A thorough understanding of the theory of the game playing should help us to formulate problems which are closer to realistic applications. Many game playing situations (such as in chess) exhibit a complexity of structure beyond those of the problem solving situations one normally encounters in everyday life (provided the intangibles of the real-life situation can somehow be removed). A primary difference between game playing and actual problem solving situations lies in the origin of the uncertainties. In a game, the uncertainties of the problem result from the large number of alternatives available, while the objective as well as the rules are rather simply stated. However, in many actual problem solving situations, the uncertainties are essential attributes resulting from an imperfect knowledge of the environment. Furthermore, the problem itself is stated in a fairly complex manner, involving a large number of variables. Hence, the study of game playing might be relevant to only those real-life problems in which the structure remains complex even after the removal of the intangibles.

Perhaps an even better approach to problem solving would be the theoretical development of a formal theory offering a guideline on how to deal with these problems.[8,11] Such a theory must of necessity be of a general nature since

problem solving situations are exceedingly difficult to classify and group into types. Most realistic problems are poorly structured and have their own specific features. The results and procedures recommended by a formal theory would have to be supplemented with additional specifications to meet the needs of the specific problem at hand. The theory must offer the principles to be observed when dealing with a given problem as well as indicate the structure of the problem solving process. The specifics of the problem have to be designated by observance of the immediate problem under consideration.

The game which has received the most attention thus far in artificial intelligence is chess. Obviously, any game which is sufficiently complex and subtle in its implications to have allowed a deepening analysis through centuries of intensive study and play without becoming exhausted is an excellent target for machine studies. A successful chess playing machine would signify an achievement penetrating to the core of human intellectual endeavor. In 1949, Shannon[1] presented a paper in which he pointed out that although chess is a finite game (there are only a finite number of positions, each of which admits a finite number of move alternatives), application of a minimaxing technique would be impossible on

even the fastest digital computer because of the large number of possibilities that must be explored. He estimated that there are approximately $10^{128}$ move possibilities as contrasted to only $10^{16}$ microseconds in a century. Shannon's proposal was that minimax continuations be explored to a certain practical depth and static evaluations then be made to determine the move with the highest effective value. The greater the depth of analysis, the better the chess that would be played. He proposed a numerical measure, based on various features that chess experts consider important, be formed by summing a number of factors that could be computed for any position. This has come to be known as intermediate evaluation. While Shannon did not present a particular computer program his specifications helped pave the way for programs by Turing[15] in 1951, Kisler, Stein, Ulam, Walden, and Wells[2] (Los Alamos group) in 1956, Bernstein, Roberts, Arbuckle, and Belsky[3] in 1957, and by Newell, Shaw, and Simon[4] in 1958. The latter effort breaks away slightly from the now conventional minimax - intermediate evaluation schema and approaches the game more as a model of human problem solving in the chess environment. The authors are apparently convinced that human-like problem solving methods would be more effective in chess problem solving than other

computational schemes that had been proposed and tried. Their conviction is supported by the comparatively fine behavior of their chess playing program.

An excellent rebuttal to their approach might be voiced by Samuel[5], whose checker-playing program might be considered the classic achievement in game playing in artificial intelligence to date. Through the use of learning routines the program has progressed from an initial classification of "poor player" to the point it can defeat recognized champions. It's position evaluation scheme and minimaxing techniques place it clearly in the "pure" artificial intelligence area, free from human-like problem solving mechanisms. Perhaps the most interesting result of Samuel's program is the possible conclusion that man is capable of solving problems without knowing precisely how he solves them.

A game need not be as complex as chess or checkers to prove of interest. Various programs have been written for simpler games such as two dimensional (3x3) tic-tac-toe. The "game-tree" here is small enough for the computer to determine the optimal move in the very beginning of the game. Such a machine is currently on display at the New York World's Fair. By its trivial nature, two dimensional tic-tac-toe provides an excellent structure to test and observe

simple rote learning procedures[7].

A more sophisticated attempt at machine learning has been presented by Wexelblat[14] using the game of Renjyu. This game is played on the vertices of a 19x19 board; a win being 5 in a row in either a horizontal or vertical direction. Wexelblat's learning procedure considers only the end-game for its strategic evaluations, virtually ignoring all offensive and defensive moves in the opening and middle-game. Since a game of Renjyu between two good players may have several blocked threats before the final one, it is apparent that much of the strategy is missed by failing to consider the middle-game.

A well-known game of relatively the same order of difficulty as Renjyu is three dimensional tic-tac-toe played in the cells of a 4x4x4 cube. This is a particularly attractive game for investigation as it contains the basic characteristics of an intellectual activity in which heuristic processes play a basic role. Some of the important characteristics of three-dimensional tic-tac-toe include:

1. The game is or can be made familiar to a substantial body of people. This is important so that the behavior of the program can be made understandable to them.
2. The rules are definite and are known. It is this

characteristic which many practical problems fail to meet. Important problems such as war games and economic processes have no definite known rules.

3. A definite goal exists - winning the game. In addition an intermediate or sub-goal which has a bearing on the final goal also exists. This will be discussed later.

4. The game is finite, yet not deterministic in the practical sense. An exploration of all possible paths would envolve approximately $10^{23}$ choices of moves, which at three choices per microsecond (far faster than the speediest existing computer) would still require a hundred million years.

5. There are no probabilistic decisions such as roll of dice, toss of coin, etc.

6. There is no concealing of information by either player as to his last move. Thus the state of the game is always known by both players.

The game of three-dimensional tic-tac-toe has been investigated by Daly[6]. In his paper "Computer Strategies for the Game of Qubic", he applies the techniques of minimax and intermediate evaluation to yield an effective game playing machine. His approach is similar in this respect to that of

Samuel in checkers. Daly attempts no learning procedure, however, because of the limited number of abstract strategies and concepts available, and the limitations of the computer used.

Although Daly recognizes that a strategy might be drawn from concepts concerning planar arrangements in the cube, he concludes that such patterns are not suited for machine use, though they might be useful to human players. Perhaps this is too hasty a brush-off of human-like heuristics in favor of the minimax-intermediate evaluation approach in a game which admittedly contains comparatively few abstract strategies and concepts. Even in the far more complicated game of chess, Newell, Shaw, and Simon[4] consider human heuristics superior to machine-like play. In their introduction they state:

> "We have now completed our survey of attempts to program computers to play chess. There is clearly evident in this succession of efforts a steady development toward the use of more and more complex programs and more and more selective heuristics, and toward the use of principles of play similar to those used by human players. We believe that any processing system - a human, a computer or any other - that plays chess successfully will use heuristics generally similar to humans."

The planar forcing techniques mentioned by Daly will be expanded in this paper to yield an alternate approach to the abstract minimax-intermediate evaluation technique based on a deeper reliance on heuristics and human-like problem

solving methods. The success of this alternative formulation indicates that the techniques of minimax and intermediate evaluation may be only one aspect of the general theory of strategy - of which another aspect has been studied here. In what follows, an indication will be made of a possible form which such a theory might take.

## 2. A SYSTEMS THEORY APPROACH TO STRATEGY AND GAME PLAYING

The concept of strategy as applied to games and game theory has been discussed in numerous publications in recent years. Mathematical techniques have been formulated to predict and determine the outcome of many simple types of games. A great deal of the research has been directed toward the application of mathematics to economic theory. However, contrary to the success of mathematics in other sciences, it has done little to improve existing economic knowledge. It's failure has led to a recent view point that the foundation of the present theory is not deep enough and that a theory of problem solving might be the key to the clarification and understanding of complex problems. The theory of problem solving is evolving from a "systems approach" based on the development and analysis of general models for problem solving systems. In a recent paper, Windeknecht[11] illustrated the technique of "isomorphic systems" in which he demonstrated the one-one correspondence between a derived model of one-person games and a finite automation. His mathematical model and corresponding analogy is used to shed light on the outcome of one-person games.

In this paper an attempt is made to define many of the concepts and applicable terminology to postulate a simple

model of games in a discussion of strategy for a particular type of game. The discussion is directed toward a "finite, perfect information, two-person game" in which "chance" is not present (in the sense that die are not thrown, cards are not drawn, etc.). Only games of pure opposition in which the outcome is win, lose, or draw are considered. Games of this type include chess, checkers, tic-tac-toe, nim, etc.

Let us begin by defining what is meant by a "strategy". A strategy of a player in a given game is a complete plan of behavior which specifies the player's decisions for all possible circumstances that may arise during the course of play.

One can easily imagine that a player does not make his individual decisions as the occasions arise during the course of play, but that he chooses a complete plan of behavior before the beginning of the game. In other words, he selects a strategy which determines all the individual decisions that may be required during the actual course of play. Clearly, this does not leave the player with fewer possibilities then if he were to make successive decisions during the course of play. His freedom of action is not restricted. In fact, by choosing a strategy, the player makes more decisions than he actually needs; only part of all possible circumstances can arise during the actual course of play. All rules of behavior

applying to circumstances which do not arise during the course of play are redundant. On the other hand, introducing these additional redundant decisions provides the formal advantage that the various decisions arising during the course of play can be combined into a single decision made before the game. This advantage is retained even if the concept of strategy is modified as indicated later. In general, a strategy not only contains decisions for circumstances which may not arise during the actual course of play, but it also contains decisions for circumstances which can not possibly arise. Owing to the decision provided by a strategy for a given situation, certain subsequent circumstances may be excluded although they would otherwise have been possible. For instance, if, in a game of checkers, a strategy calls for a specific move yielding circumstance $\beta$ from circumstance $\alpha$, then circumstance $\beta'$ arising from $\alpha$ by means of a different move cannot arise. Decisions for such nonrealizable circumstances need not be made, of course, and the concept of strategy as a complete plan of behavior can be modified accordingly. It would seem that for some purposes such as the proof of theorems in formal logics, the original concept of strategy as a complete plan of behavior is more appropriate; while for the study of particular games, the modified concept might be best.

A deeper understanding of the concepts involved in strategy can be obtained by considering the framework about which strategies are built. For this one must precisely define some of the terms necessary in the discussion.

A "game" may be defined as the totality of rules which describe it. The abstract concept of a game is to be distinguished from the individual "plays" of that game in that a "play" is a particular instance at which the game is played from beginning to end. The component element of the game, the "move", is the abstract occasion of a choice by either player between various alternatives under conditions precisely described by the rules of the game. The specific alternative chosen in a given instance in a play of the game is the "choice". Moves are related to choices in the same way as the game is to the play: the game consists of a sequence of moves; the play of a sequence of choices. The "rules" of the game must be considered absolute in the sense that an infraction of the rules, by definition, changes the nature of the game. Under no circumstances should the rules of the game be confused with the players' power to use or reject a particular strategy, be it good or bad. As previously mentioned, this paper is only concerned with those games which are "finite" and contain "perfect information". Such games are limited by their rules

to a finite number of states, and have the property that the complete state of the game is known by each player at every specific move time.

Consider the game $\Gamma$ of two players X and Y. The play of the game consists of a sequence of moves whose number can be denoted ty the integer $v$ (since we are considering finite games). The moves themselves can be denoted by the sets $M_x$ and $M_y$.

$$M_x = \{ x_1, x_2, \ldots x_n \}$$
$$M_y = \{ y_1, y_2, \ldots y_m \} \qquad \text{where } v = n+m$$

Any move $x_i (i=1,2,\ldots n)$ or $y_i (i=1,2,\ldots m)$ is a choice made from a number of alternatives. Move $x_i$ is thus chosen from the set

$$\{ x_i(1), x_i(2), \ldots x_i(\alpha_i) \}$$

where $\alpha_i$ is the total number of alternatives $\alpha$ for move i of player X.

Let $\Omega$ be the set of all possible situations which may arise in $\Gamma$. Each "particular situation" (i.e. element of $\Omega$) will be referred to as a "state of $\Gamma$". Thus, the set $\Omega$ will be called the complete state set and will be finite in many cases of interest. Any element of $\Omega$ may be obtained by a proper succession of elements from $M_x$ and $M_y$ as specified by

the rules of the game. Because of this unique formation of the elements of the set, the state set $\Omega$ is said to contain "connected" elements. Any pair of states a,b (where a,b $\varepsilon$ $\Omega$) are said to be "connected" if and only if there exists a sequence of inputs $\{x_1, x_2, \ldots x_j\}$ to the system such that

$$f(f(\ldots f(f(f(a,x_1)x_2),x_3)\ldots)x_{j-1}),x_j) = b$$

where f is the state transition function mapping a state and an input (a move) into a new state. The "order of a connection" is the number of moves that separate the states in question. Thus, the order of connection of a to b is $n$ if $n$ moves are required to take the game from state a to state b.

Let us denote those elements of $\Omega$ which correspond to a win for X as $\Omega_x$ and those which correspond to a win for Y as $\Omega_y$. It is obvious that these two subsets are disjoint. A tie game or draw can be denoted by a third subset $\Omega_T$. Note that whereas $\Omega_T$ is complete in tic-tac-toe, a tie in checkers might depend on a prescribed number of inconsequential moves, no one of which leads to a state in $\Omega_T$. More will be said about tie games later.

The prime objective of player X is to achieve an $\Omega_x$ state before his opponent can achieve an $\Omega_y$ state. Should the prime objective be impossible, a lesser goal would be a

draw game; i.e. keeping Y from achieving a state in $\Omega_y$.

Consider the state transition function f which maps the Cartesian product of a given state and an X move into another possible state (i.e. some state which is a first order connection to the given state). Similarly let g be the corresponding function for Y.

Expressing the above in functional notation we have:

"Move choice"
$$\begin{cases} f : A \Rightarrow \Omega \\ g : B \Rightarrow \Omega \end{cases}$$

where:

$$A \subseteq \Omega^* \left\{ \bigcup_{i=1}^{n} x_i \right\}$$

$$B \subseteq \Omega^* \left\{ \bigcup_{i=1}^{m} y_i \right\}$$

Goals of X and Y:

Primary goal of X: $\Omega_x \subset \Omega$

Primary goal of Y: $\Omega_y \subset \Omega$

Secondary goal of X and Y: $\Omega_T \subset \Omega$

Relations between goals:

$$\Omega_x \cap \Omega_y = \phi$$

$$\Omega_x \cap \Omega_T = \phi$$

$$\Omega_y \cap \Omega_T = \phi$$

The game itself may be modeled as a simple system of state transitions dependent upon the functions f and g.

$$\begin{cases} s_o = b \quad (b \in \Omega) \\ s_{2i-1} = f(s_{2i-1}, x_i) \\ s_{2i} = g(s_{2i-1}, y_i) \end{cases}$$

The primary objective of player X can be expressed as a recursive function which takes into account the move choice of player Y. Define the "X-strategy" as a function.

$$F_x: \ \Omega \Rightarrow \left\{ \bigcup_{i=1}^{n} x_i \right\}$$

Given an X-strategy one can associate a mapping between a sequence $\{ y_i \}$ of moves made by Y and a sequence of states $\{b_i\}$ $(b_i \in \Omega)$ such that:

$b_i = g(f(b_{i-1}; F_x(b_{i-1})), y_i)$ ; $i = 1,2,\ldots n$

$b_o$ = the starting state

The strategy $F_x$ will be called a "primary X-strategy" if:

(1) For all i and for all y, $b_i \notin \Omega_T \cup \Omega_y$

(2) For all y there exists n such that $b_n \in \Omega_x$

It is obvious that such a strategy might not be realizable. Thus, define a "secondary X-strategy" as one in which only condition (1) is realized. The definition of an X-strategy

brings out the feedback attribute characteristic of games. A state of the game along with the strategy determines the player's move. The move together with the previous state determines the new state. This process continues until the game terminates. It may be represented in block diagram as shown in Figure 2.1

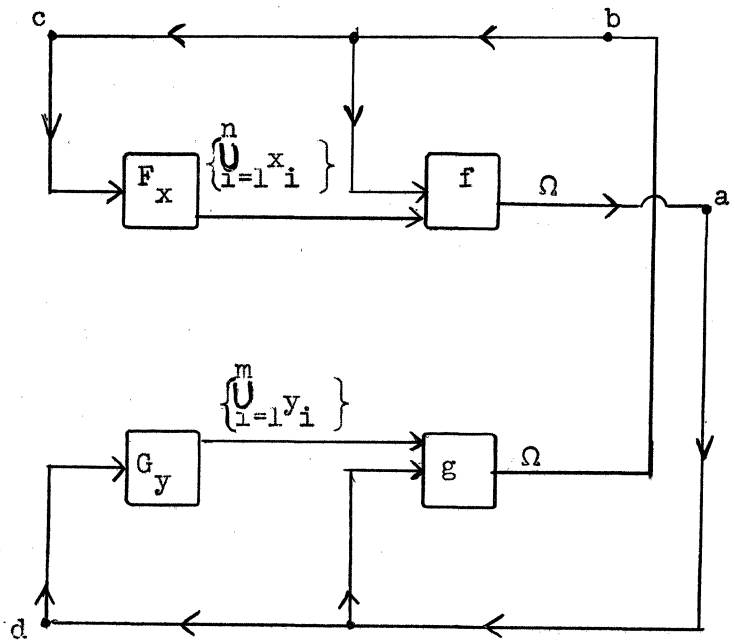



Figure 2.1

The game shown in Figure 1 begins at point "a" if Y moves first, and at point b if X moves first. A move sequence requires cycling through the points b,c,a,d respectively.

Let us now consider the characteristics of a primary X-Strategy. This strategy considers sequences in which for

any $y_i(k) \in y_i$ in the move sequence, an $x_i(k) \in x_i$ is available such that eventually the sequence will terminate in $\Omega_x$. Furthermore, the definition excludes the possibility of the sequence ever entering into a state within the set $\Omega_y$. In some very simple games, such as nim, a primary X-strategy may be applied on the first move, guaranteeing a win for the first player. In others such as tic-tac-toe (3x3) no such strategy exists, and a win may be achieved only through a poor move on the part of the opponent. Thus, a secondary X-strategy would be applicable to tic-tac-toe.

In a more complicated game such as checkers, it is not obvious whether or not a primary or even secondary X-strategy exists on the first move. Intuition points to the likely possibility that the first player should be able to do no worse than tie in checkers, but this has yet to be proven. Perhaps the best strategic attack to games in which an X-strategy is not immediately evident is to decide on a set of states $\Omega_s$ called the strategic sets which are advantageous in the sense that each forms the basis of a primary X-strategy. A strategic set of states $\Omega_s \subset \Omega$ has the property that for any sequence $M_y$,

$$M_y = \left\{ y_q, y_{q+1}, \cdots y_p \right\}$$

there exists a sequence $M_x$,

$$M_x = \left\{ x_q, x_{q+1}, \ldots x_p \right\}$$

such that $b_{2p+1} \varepsilon \Omega_x$ if $b_q \varepsilon \Omega_s$ (where the subscript on $b \varepsilon \Omega$ corresponds to the subscripts on moves in $M_x$ and $M_y$ yielding this state).

Once a primary X-strategy has been started, the two-person game can be considered as having degenerated into a one-person game since the outcome is dependent only on the moves of the second player. A one-person game has the advantage that the player has complete control over the state transformations; once a transformation is chosen, only one outcome is possible.

A sub-strategic set of states $\Omega_{s'} \subset \Omega$ may be defined as those states having the property that for any sequence $M_y = \left\{ y_q, y_{q+1}, \ldots y_p \right\}$ there exists a sequence $M_x = \left\{ x_q, x_{q+1}, \ldots x_p \right\}$ such that $b_{2p+1} \varepsilon \Omega_x$ if $b_q \varepsilon \Omega_s$. The sub-strategic set $\Omega_{s'}$ is related to the secondary X-strategy in the same manner as the strategic set $\Omega_s$ is related to the primary X-strategy. As previously pointed out, a draw is possible in many games without entering a state which is an element of $\Omega_T$. In many cases, the draw is so declared by the rules of the game to prevent an infinite

sequence of moves. A sequence of this type, known as a "chain," is defined as follows: Let $\Omega_c$ be an arbitrary subset of $\Omega$. $\Omega_c$ is said to be a chain if and only if there exists an ordering of its elements, say $\Omega_c = \{b_1, b_2, \ldots b_n\}$ such that:

$$\begin{cases} h(b_i) = b_{i+1} \\ h(b_n) = b_1 \end{cases} \quad (i = 1,2,\ldots n-1)$$

where h is a state transition function related to f and g. Obviously, a chain is not a "stable" set in the general sense (i.e. for all games containing a draw) since it is dependent on the whims of two individuals. Thus, game rules usually provide that a draw be declared when a preassigned number of "inconsequential" moves are made in succession. However, it may be conjectured that continuation of sequences of "inconsequential" moves would eventually lead to a chain anyway.

Let us now consider the nature of the state set $\Omega$ more closely. A first order connection to some state $b_o \in \Omega$ might consist of any one of the elements of the set $b_i \subset \Omega$, $(i = 1,2,\ldots n)$. The set $b_i$ may thus be considered a subset in $\Omega$ which can be reached from the state $b_o$. Emanating from every element of the $b_i$ is another subset in $\Omega$, $b_{ij}$ $(j = 1,2,\ldots m)$, each of whose elements is a first order connection to a state in $b_i$ and a second order connection to

$b_o$. To illustrate, $b_{34}$ represents the third of n possible choices as a first order connection to $b_o$, and the fourth of m possible choices as a first order connection to $b_3$. Continuing in this manner, the state of the game may be represented at any time as an element $b_{ijkl...q}$ of the set $\Omega$. This labeling of states might be termed a "connective numbering system".

The connective numbering system model of a game leads to the concept of a "tree", an artifice which is used extensively in game theory literature. The alternatives available to progress from one state to a first order connection give rise to the "branches" of the tree. A simple tree configuration with the connective numbering system is shown below in Figure 2.2.

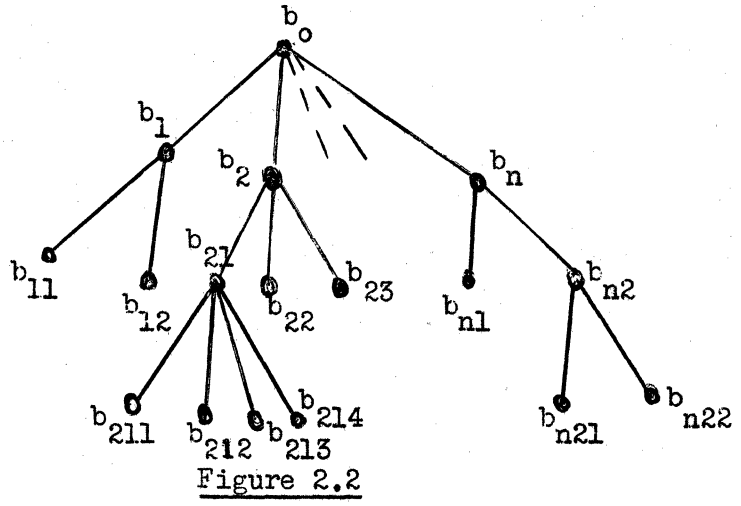



Figure 2.2

The tree representation clearly shows a prime difficulty in dealing with games: the number of possibilities grows exponentially with move number as viewed from the initial

state. Actually, the system has an additional complexity.

One need only be vaguely familiar with games to realize that some states may be reached by several alternatives. Thus $b_{21}$ might be the same state as $b_{68}$ or $b_{134}$. Consequently, the tree degenerates in a "net" which has the characteristic that flow may only proceed in one direction as shown in Figure 2.3.

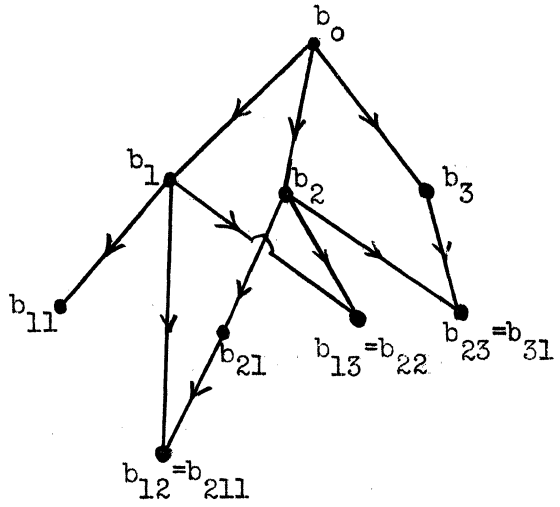



Figure 2.3

The model of a game as a non-planar network of possible states best portrays the idea of "connection" presented earlier.

The classical approach for determining a path through the game network leading to a desired final state in a two-player game is known as the minimax technique. In this method, the following question is asked recursively until

the net is exhausted and the end of the game has been determined: "Where can I move such that my position after my opponents move will be the best I can hope for should he make the best countermove?" While minimax is effective in trivial games, it is of little use in a complex game like chess where the possibilities to be explored exceed $10^{128}$. A digital computer checking at the fantastic rate of 6 moves per microsecond would require an amount of time greater than the expected lifetime of our universe to check all these possibilities. Thus, the approach used in complex games is to reduce the depth of search through the net to some feasible limit and to investigate the relative values of the end points of the search. The operation undertaken at these end points is usually based on the status of various empirically established criteria considered to be important characteristics of the particular game. To be more explicit, consider the sets $\Omega_{I_i} \subseteq \Omega$ $(i = 1,2,...n)$ where $\Omega_{I_i}$ is a set of states having a certain characteristic in common. A good strategy might be to move to a state which is contained in the intersection of a number of the $\Omega_{I_i}$. To illustrate, states $b_1$ and $b_2$ in Figure 2.4 are contained in the intersection of

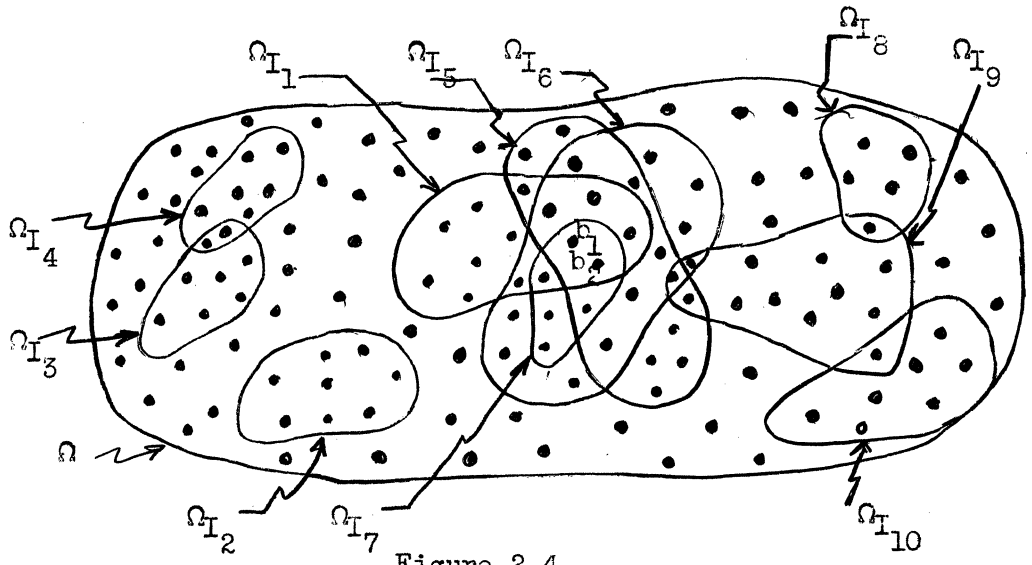



Figure 2.4

four sets belonging to $\Omega_I$ (i.e. $b_1, b_2 \in \Omega_{I_1} \cap \Omega_{I_5} \cap \Omega_{I_6} \cap \Omega_{I_7}$).

It is quite possible that good criteria for intermediate evaluation coupled with a limited opening minimax application might lead to a state situation from which a primary or secondary X-strategy could be applied.

In this case, the outcome of the game is rendered deterministic if and only if the opponent follows the moves that the abbreviated minimax application shows to be his best. However, the entire argument breaks down if the opponent chooses a path other than the minimax prediction. (In a game where minimax is used to find a path from a present state to the end of the game, a move other than one predicted as his best by the minimax technique will not aid the opponent. On the other hand, in a game where minimax is limited

in depth, the evaluation of the alternatives may be in error, and the wrong path might be chosen. Such a possibility always exists due to the empirical nature of the evaluation criteria.) Thus, the technique of limited minimax and intermediate evaluation might not be a good strategy to follow in complex games.

A very important factor which the minimax technique does not consider is the "ability" of the opponent. If a great deal is known about an opponent's reaction to specific situations, statistics can be set up which predict with a certain probability which alternative he will choose. It is certainly possible to conceive of quite elaborate schemes which could be used for saving information about a particular player in order to determine the best approach to take when playing him again. Entire games might be saved in order to obtain an estimation of how he will react to a given situation. This technique might be used in a situation which appears to yield a draw by conservative play using the minimax technique. "Daring" play backed by staunch probabilistic criteria for success might convert such a situation into a win, while risking a loss should the opponent play the minimax strategy. Carrying this reasoning one step further, a player should not use a strategy which can be

detected and predicted by his opponent unless it will lead to a win regardless. In other words, a slight deviation from a consistent overall strategy for the express purpose of perplexing the opponent might enhance the probability of obtaining the goal.

In lieu of the minimax technique, heuristics may be established to reach desired intermediate points or criteria or perhaps to reach as far as the states $\Omega_s$ from which a primary X-strategy is available. The value of a heuristic may be characterized by its efficiency in getting to the desired state and its effectiveness in reducing the path search to achieve this goal. By the use of heuristics, the possible number of paths through the network is usually drastically "pruned". A strategy may thus be considered in this case as the use of a heuristic approach to the goal.

The two primary strategic constituents of the two-person competitive class of games discussed may be summarized briefly as follows:

1. Reasoning with heuristics that select fruitful paths of exploration in a space of possibilities that grows exponentially.
2. Using these heuristics to reach a state in which the fixing of the strategy of one decision maker renders

the system deterministic.

As pointed out in the introduction, the game of three dimensional tic-tac-toe is ideal for purposes of demonstrating the use of heuristic processes. The remainder of the paper will be devoted to a discussion of this game and a step-by-step development of strategies and heuristics enabling a computer to play in a sophisticated manner.

## 3. DESCRIPTION OF THE GAME

Three dimensional tic-tac-toe is a finite, perfect information game played by two opponents on the 64 cells of a 4x4x4 cube (see Figure 3.1). Opponents alternate moves, a move being the occupation of any previously vacant cell in the cube. Any position involving a straight line through the cube containing four cells occupied by the same player constitutes a win for that player. A two dimensional representation of the cube can be obtained by breaking up the 4x4x4 configuration into four 4x4 planes. For identification purposes the cells can be numbered by specifying their coordinates on a three-axis Cartesion system as shown in Figure 3.2. Of the three numbers assigned to each cell, the first identifies the level (L) of the cell, the second identifies the row (R) within the level, and the third identifies the column (C) within the level.

A total of 76 straight lines containing four cells can be shown to exist in the cube. Such lines will be referred to as "files". The 76 files can be classified into three groups as follows:

1. Variation of a coordinate with other two coordinates constant (i.e. File orthogonal to two axes)

    a. Variation of C with constant L and R (16 possibilities)

b. Variation of L with constant C and R (16 possibilities)

c. Variation of R with constant L and C (16 possibilities)

2. Variation of two coordinates with constant third coordinate (i.e. File orthogonal to one axis)

a. Variation of R and L with constant C (8 possibilities)

b. Variation of R and C with constant L (8 possibilities)

c. Variation of L and C with constant R (8 possibilities)

3. Variation of all three coordinates (i.e. File orthogonal to no axes)

a. "Cross" diagonals of cube (4 possibilities)

Due to the presence of the diagonal files, some cells have more files passing through them than others. The eight vertice cells (see Figure 3.1a) and the eight "core" cells (see Figure 3.1b) of the cube are contained in seven files while all the other cells in the cube are found in only four files.

One can study in isolation the eighteen distinguishable 4x4 planes (16 cells) which can be shown to be contained in the cube. There are two distinct groups of classification.

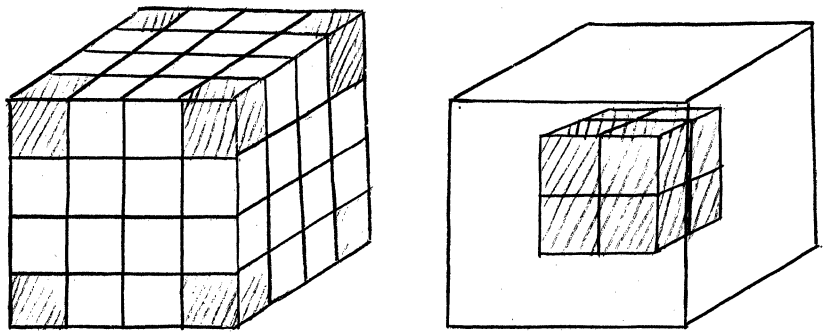
1. Planes with a constant coordinate (i.e. planes parallel to the plane formed by the intersection of any

two axes)

a. Sixteen cells with same L (4 possibilities)
b. Sixteen cells with same R (4 possibilities)
c. Sixteen cells with same C (4 possibilities)

2. Planes formed by "side" and "cross" diagonals of cube (6 possibilities)

A representative example of each of these two main groups are shown in Figure 3.3. As might be expected, all cells are not contained in the same number of planes. Each of the eight vertice cells and the eight core cells is a member of six planes; each of the other cells is contained in four planes.



(a) Vertice cells (b) Core cells

Figure 3.1

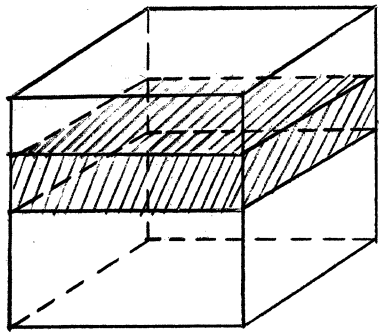
| | | | |
|---|---|---|---|
| 111 | 112 | 113 | 114 |
| 121 | 122 | 123 | 124 |
| 131 | 132 | 133 | 134 |
| 141 | 142 | 143 | 144 |

| | | | |
|---|---|---|---|
| 211 | 212 | 213 | 214 |
| 221 | 222 | 223 | 224 |
| 231 | 232 | 233 | 234 |
| 241 | 242 | 243 | 244 |

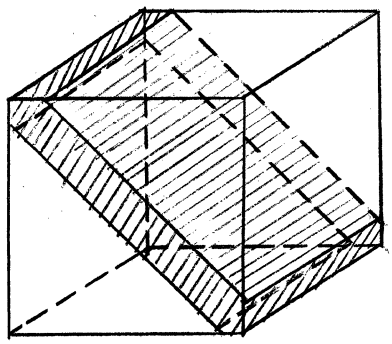
| | | | |
|---|---|---|---|
| 311 | 312 | 313 | 314 |
| 321 | 322 | 323 | 324 |
| 331 | 332 | 333 | 334 |
| 341 | 342 | 343 | 344 |

| | | | |
|---|---|---|---|
| 411 | 412 | 413 | 414 |
| 421 | 422 | 423 | 424 |
| 431 | 432 | 433 | 434 |
| 441 | 442 | 443 | 444 |

Figure 3.2
Carterian Numbering System



(a) Group 1a



(b) Group 2a

Figure 3.3

## 4. ELEMENTARY APPROACH

Since the ideal of minimaxing over an exponentially growing game tree is impractical, one if forced to reduce the size of the search tree in various ways. One of these is the reduction of the depth of search - not going all the way to the end of the game tree, but to stop exploring at some set of intermediate points. The other consists of "pruning" - where instead of looking at all possible branches emanating from a node, one looks at only those which, by some consideration, appear to be more "promising". We shall have later occassion to discuss this latter technique. Meanwhile, we should remind ourselves that in games where the final outcome is not a numerical-value pay-off but merely consists of a "win-or-lose" decision, minimax becomes a rather trivial operation, although one still calls it minimax. Consider a play of such a game between player X and an opponent. In the win-lose situation the value of a state with the opponent to move is a "win" if the value of all the states that can be reached from this state is a win. ("State" as used here refers to a particular board configuration.) The value of a state with player X to move is a "win" if at least one state that can be reached from this state is a win. The relation between this definition of the set of "win" positions and the "strategic set" $\Omega_s$ of Chap. 2 is

to be noticed.

In three dimensional tic-tac-toe let us designate the state of any file by an ordered pair of integers (x,y) where x is the number of cells in the file occupied by the player under consideration, and y is the number of cells occupied by the opponent. A file designated by (x,y) will be called an "x-y file". The player under consideration will be referred to as X, and his opponent as Y.

The novice player in his first attempts at this game usually follows the sole strategy of blocking files which are 0-3 (making them into 1-3 files); or, if none exist, building 2-0 files into 3-0 files. It soon becomes obvious to the novice that the lone formation of a 3-0 file is of little use against an alert opponent. A "stronger" strategy is to strengthen the positioning of the pieces until a state is reached from which two 3-0 files can be formed simultaneously. On defense the objective is to keep the opponent from obtaining the same opportunity. In the terminology of the last chapter, the set of winning states $\Omega_X$ for player X consists of all possibilities in which a 4-0 file will exist prior to the formation of a 0-4 file. A state from which an X move yields two 3-0 files is obviously a third order connection to $\Omega_X$. (i.e. In three moves - two by X and one by Y - the play

may terminate in a win for X.

The formation of two 3-0 files can be achieved by a simple check on the state of the 76 files. Let us define a "related pair" as two 2-0 files which intersect at a vacant cell. Moving to this vacant cell obviously produces a pair of 3-0 files. A strategy based on this small amount of information can be applied to a computer for use against human opposition (see, however, the next chapter). A more complete discussion of the background for this strategy is presented in Appendix I.

Essentially this related pair strategy consists of four basic questions posed in the following order:

a. Does the computer have a 3-0 file?

b. Does the opponent have a 0-3 file?

c. Does the computer have a related pair?

d. Does the opponent have a related pair?

If the answer to any of these four questions is yes, proper action is taken. If the answer to all four questions is no, a move is made from a "preset" move list containing the 64 cells in a predetermined order. The preset move list can be arranged to take advantage of those cells in the cube which are contained in seven different files (cube vertices and core cells) or it can simply be a random ordering of the

64 cells.

A flow chart of the "related pair strategy" is shown in Figure 4.1. In addition to the basic four questions discussed above, the flow chart includes several other pertinent questions. These are numbered in the flow chart and explained below.

1. An illegal move by the opposition would wreak havoc with the computer's file indexes and must be guarded against. Moving into an occupied cell or moving into several cells simultaneously might constitute an illegal move.

2. An initial check should be made to see if the opponent has formed an 0-4 file and thus won the game.

3. An initial check should be made to see if all board positions are filled, leaving the game in a draw.

4. Obviously questions (a) and (b) above used not asked until at least three moves have been made by the appropriate player, and questions (c) and (d) used not be asked until at least four moves have been made.

5. The dotted boxes labeled (5a) and (5b) in Figure 4 "predict" a win for the opponent in one and two (opponent) moves respectively. If desired, the computer can assume that the opponent will not overlook the available 0-3

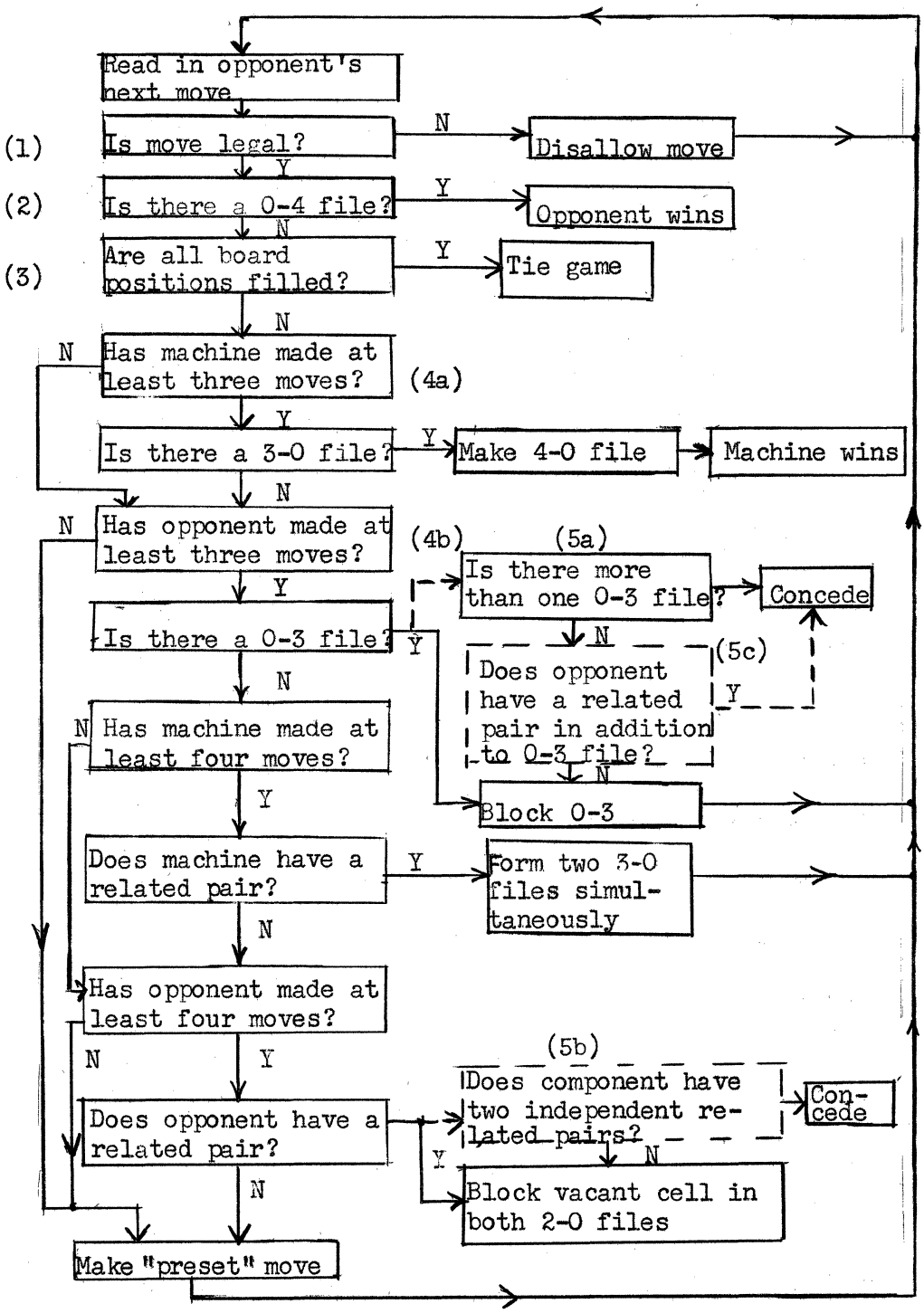
file or related pair, and can concede the game at this point. The dotted box labeled (5c) might also lead to a concession if the following conditions are met:

a. Opponent has a related pair in addition to the 0-3 file.

b. Blocking the 0-3 file has not either blocked the related pair or yielded a 3-0 file for the computer.

Generation of a situation as described in the explanations of (5c) involves a more complex process than the related pair strategy outlined. The simultaneous formation of a 3-0 file and a related pair forms a "stronger" strategy than the lone formation of a related pair since the former leads to a win while the latter can be blocked by an alert opponent. Going one step further, the successive formation of 3-0 files leading eventually to the simultaneous formation of a related pair and a 3-0 file constitutes an even higher strategical level. If no 0-3 file can be formed by the opponent in blocking the succession of 3-0 files, the force sequence will terminate in a victory for the offensive player. Thus, a situation in which the opponent is faced with losing immediately (by not blocking the 3-0 file) or losing further on (when a 3-0 file and related pair are formed

Figure 4.1: Flow Chart of Related Pair Strategy



simultaneously) will be referred to as a "terminal forcing sequence".

## 5. TERMINAL FORCING SEQUENCES

A terminal forcing sequence may initiate a force to a win several moves prior to the termination of the play. Its basic premise at any move is the formation of a 3-0 file in such a manner as to provide a 2-0 file for use on the following move, and the eventual simultaneous formation of a pair of 3-0 files. To simplify the verbal description of the formation of terminal forcing sequences let us define the counterforce set $\Omega_\beta$ as the set of states which contains a 0-3 file whose unoccupied cell is not in a 2-0 file.

Most terminal forcing sequences fall into two general cases. (Examples will be provided in the next chapter.) These cases provide sufficient but not necessary conditions for a terminal forcing sequence.

Case 1: If there are two or more 2-0 files initially, and if a sequence of 1-0 and 2-0 files exists such that they (a) intersect each other at vacant cells; (b) intersect two of the 2-0 files at vacant cells, and (c) intersect the remaining 2-0 files at occupied cells; then a terminal forcing sequence can be executed provided that (d) the sequential blocking does not establish a state in $\Omega_\beta$.

A Case 1 terminal forcing sequence can be executed by

moving to a vacant cell in a 2-0 file such as to leave a new 2-0 file. This is always possible because each 2-0 file intersects a 1-0 file at a vacant cell. Since all initial 1-0 files in the sequence are constrained to intersect each other at vacant cells, this procedure can be repeated until a 3-0 file is formed while making a 2-0 file out of the 1-0 file intersecting an original 2-0 file. The resulting state will contain both a related pair and a 3-0 file. However, due to condition (d) on the preceding page, the probability of long range terminal forcing sequences is drastically reduced because of the possibility that the opponent will form an $\Omega_\beta$ state while blocking a 3-0 file.

Case 2: If there is at least one 2-0 file initially, and if a sequence of 1-0 files exists such that they intersect each other at vacant cells, and at least one intersection involves more than two files; then a terminal forcing sequence can be executed provided that the sequential blocking does not establish a state in $\Omega_\beta$.

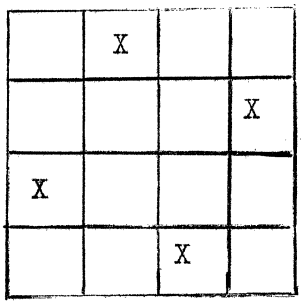
The force procedure for Case 2 differs from that of Case 1 in that a second 2-0 file to terminate the force must be set up. The procedure for this will be made clear by examples later on. Although a terminal forcing sequence is maximally

effective when encompassing the entire 64 cell cube, it is most easily recognized, and there are far fewer cases to look for in a given 4x4 plane of the cube (eighteen such planes can be shown to exist). The terminal forcing sequences described in this chapter can be used to obtain an excellent "planar strategy". In the next chapter, in addition to describing the planar strategy, examples and illustrations of the two cases of the terminal forcing sequences are given.

## 6. PLANAR STRATEGY

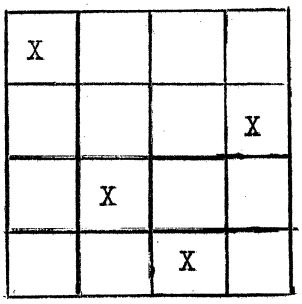
A terminal forcing sequence can often be recognized by a given advantage in a 4x4 plane of the cube. Consider a plane in which X occupies four cells while Y occupies none. Such a plane will be called a 4-0 plane. The term "excess-four plane" will denote a plane in which X occupies four cells more than Y. For purposes of simplicity the plane under consideration will be assumed to be disjoint from the remainder of the cube. It can be shown by a process of exhaustion that every 4-0 plane which contains at least one 2-0 file is a state in a terminal forcing sequence. (All but a very few of the possible cases are covered by cases 1 and 2 in the preceding chapter.) Of the 1820 possible configurations of four occupied cells in a set of sixteen, 1804 contain at least one 2-0 file and thus yield a terminal forcing sequence. There are only two basic configurations of 4-0 planes, each having eight symmetries, which do not have at least one 2-0 file. They are shown in Figure 6.1. Because of their resemblance to chess patterns, the configurations are labeled the "knight's position" and the "castle's position", respectively. Although these two positional arrangements do not contain a 2-0 file, they are still quite interesting from a forcing standpoint. More will be said about them further on

in this chapter.




(a)

"Knight's position"

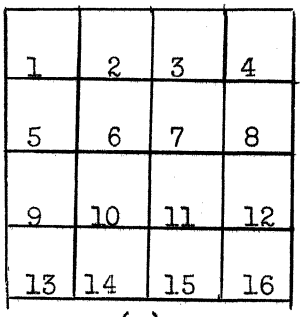



(b)

"Castle's position"

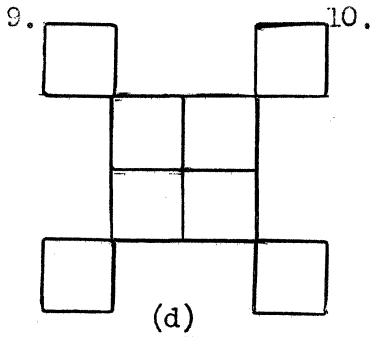
Figure 6.1

Several examples of the two cases of the terminal forcing sequence will be illustrated using a 4-0 plane. For purposes of discussion, the sixteen cells and the ten files in the plane will be numbered as shown in Figure 6.2.



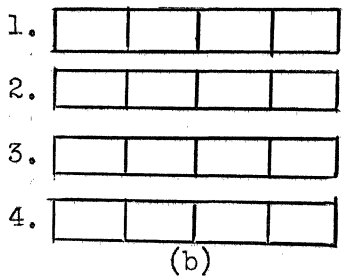

(a)

Cell numbering



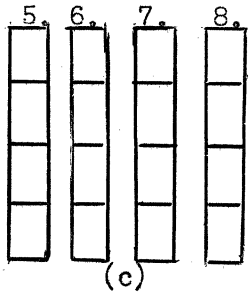

(d)

"Diagonal" files




(b)

"Row" files




(c)

"Column files

Figure 6.2

Sequences of file intersections will be denoted by a shorthand notation as follows. The notation $10_2 \rightarrow 3 \rightarrow 2 \rightarrow 1_2$ will be interpreted to read: "File 10, a 2-0 file, intersects file 3, a 1-0 file, at a vacant cell. File 3, a 1-0 file, intersects file 2, a 1-0 file, at a vacant cell. File 2, a 1-0 file, intersects file 1, a 2-0 file, at a vacant cell."

Example 1 Situation (figure 6.3a): X has two 2-0 files (4, 10) intersecting at the vacant cells (16, 4) of a 1-0 file (8).

Force Procedure: Form a sequence of 2-0 and 1-0 file intersections which begin and end with a 2-0 file. This will be a terminal forcing sequence according to Case 1. The intersection sequence in this example is: $10_2 \rightarrow 8 \rightarrow 4_2$

The move sequence is thus as follows:

1. Move to cell 4, the intersection of 2-0 file 10 with 1-0 file 8. The opponent will move to cell 7 to block the 3-0 file.

2. Move to cell 16, the intersection of the newly formed 2-0 file 8 and 2-0 file 4. Both of these files are now 3-0, guaranteeing a win.

The completed terminal forcing sequence with the move order preserved by subscript notation (i.e. $X_n$

is the $n^{th}$ move by X in the force procedure) is shown in Figure 6.3b.

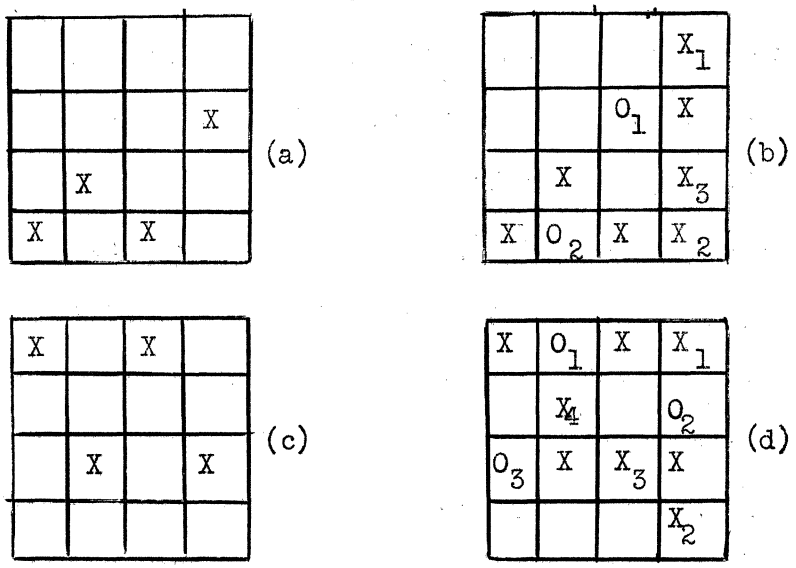



Figure 6.3

Example 2 Situation (figure 6.3c): X has two 2-O files (1, 3) which are intersected by many 1-O files (5, 6, 7, 8, 9, 10).

Force Procedure: Form an intersection sequence as follows: $1_2 \rightarrow 8 \rightarrow 9 \rightarrow 3_2$.

The succession X moves are thus to cells 4, 16, 11, and either 6 or 9 depending on which of the two 3-O files O chooses to block. The complete force is in Figure 6.3d. Notice that the intersection sequence is not unique. Some of the other possibilities are:

a. $1_2 \rightarrow 10 \rightarrow 5 \rightarrow 3_2$

b. $1_2 \rightarrow 6 \rightarrow 9 \rightarrow 3_2$

c. $1_2 \rightarrow 10 \rightarrow 7 \rightarrow 3_2$

d. The reverse of all listed possibilities (i.e. $3_2 \rightarrow 5 \rightarrow 10 \rightarrow 1_2$ etc.)

We shall later discuss the importance of having such alternatives in view of the possibility of $\Omega_\beta$ states.

Example 3 Situation (Figure 6.4a): X has two 2-O files (1, 7) and four 1-O files (3, 4, 5, 6) with a sequence of intersections at vacant cells.

Force Procedure: $1_2 \rightarrow 5 \rightarrow 4 \rightarrow 6 \rightarrow 3 \rightarrow 7_2$

This force sequence as shown in Figure 6.4b can be shorted by a recursive procedure in which the intersection sequence is studied again after each X - O move pair. Doing this, the force sequence can be shortened by one X move. The new move sequence, shown in Figure 6.4d is obtained by using the intersection sequence $5_2 \rightarrow 4 \rightarrow 9 \rightarrow 7_2$ after application of the $1_2 \rightarrow 5$ opening (see Figure 6.4c) of the original intersection section.

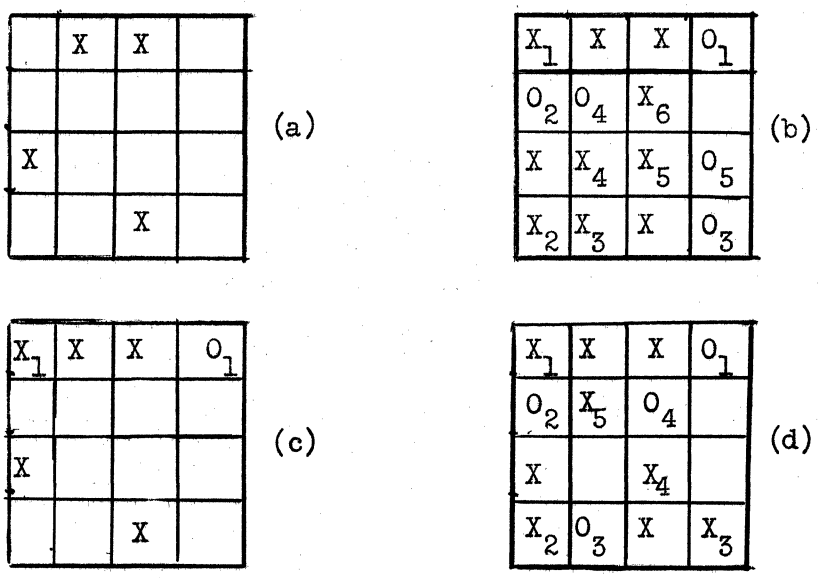



Figure 6.4

Example 4 Situation: X has four 2-O files (2, 3, 10, 6) and three 3-O files (7, 8, 9) (Figure 6.5a)

Force Procedure: $3_2 \rightarrow 9 \rightarrow 8 \rightarrow 10_2$ (Figure 6.5b)

Example 5 Situation: X has one 2-O file and a file intersection at cell 16 involving two 1-O files and one 2-O file (Figure 6.5c).

Force Procedure: (a) $9_2 \rightarrow 4$, 8 (b) $4_2 \rightarrow 5 \rightarrow 3 \rightarrow 8_2$ (Figure 6.5d) The force is broken up into two sections. A force is made to reach the triple intersection making two 1-O files into 2-O files. Then a second force is initiated there following the Case 1 terminal forcing sequence procedure.

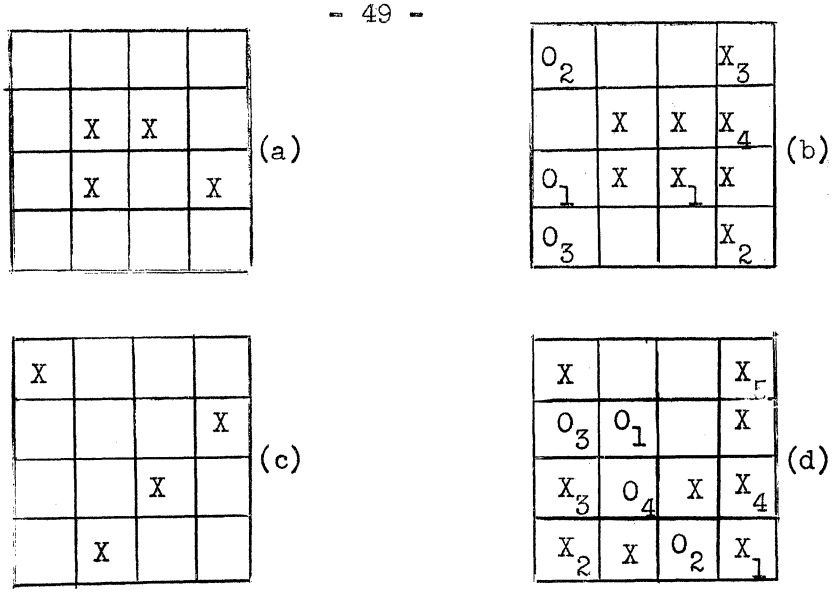



Figure 6.5

Examples 1, 2, and 3 illustrate Case 1 sequences of growing length; example 4 illustrates a Case 1 sequence with more than two initial 2-0 files. From example 5 it is clear that terminal forcing sequence Case 2 is an extension of Case 1, carrying with it an additional procedure.

The planar terminal forcing sequences include far more situations than 4-0 planes or even excess-four planes. A largely majority of 3-0 planes and excess-three planes containing at least one 2-0 file can be shown to lead to a terminal forcing sequence. (A complete list of these 3-0 planes is given in Appendix II.) Thus, the excess-three planes might correspond to some of the desirable intermediate points denoted by $O_I$ in Chapter 2. An example of a Case 1 type 3-0 plane is shown in Figure 6.6 a,b and a Case 2 type in

Figure 6.6 c,d,e. In the Case 2 situation the move to

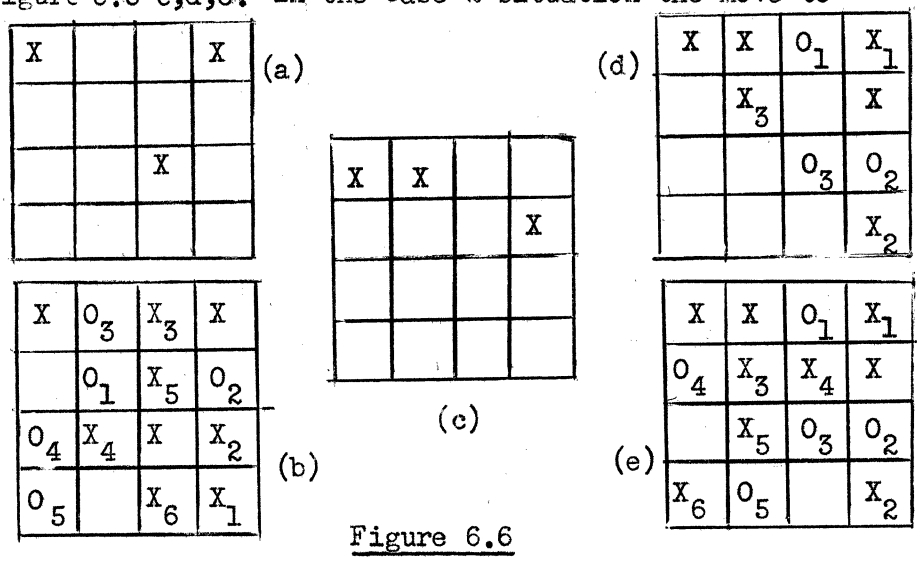



Figure 6.6

the triple file intersection at cell 6 cannot be made until a file passing through it is 2-0. The intersection sequence here is: (a) $1_2 \rightarrow 8 \rightarrow 9 \rightarrow 2$, 6 (figure 6.6d) (b) $2_2 \rightarrow 10 \rightarrow 6_2$ (Figure 6.6e).

Notice that the second portion of the intersection sequence cannot be $2_2 \rightarrow 1 \rightarrow 6_2$ because a 0-3 file is formed in this sequence. This is the $\Omega_\beta$ restriction found in the terminal forcing sequence cases.

Thus far the planar discussion has been limited to situations in which the plane is completely isolated from the rest of the cubic configuration. Quite often there will be some 2-0 or 0-2 files in the cube lying outside the plane under consideration and having a vacant cell in that plane. Let us first consider the case in which such a 0-2 file is

present. The forcing sequence must then be constructed to insure that this vacant cell is never available for occupation by the opponent since this will lead to an $\Omega_\beta$ situation. The cell can be avoided altogether or simply used as an X move in the terminal forcing sequence. In the instance in which a 2-0 file in the cube has a vacant cell in the plane under study, the probability of the existence of a terminal forcing sequence being available is greatly increased. Moving to the vacant cell in the 2-0 file adds to the advantage in the plane since the opponent's move is forced out of the plane to block the 3-0 file. Applying this procedure one could convert a nonforcing excess-three plane to an excess-four plane, most all of which yield a terminal forcing sequence. Extending this reasoning one step further, an excess-two plane (i.e. a 2-0 plane, 3-1, plane etc.) containing a vacant cell which is in a 2-0 file outside the plane, and even an excess-one plane containing two vacant cells which are in 2-0 files outside the plane constitute the threat of a possible terminal forcing sequence and thus are also states in $\Omega_I$.

The strong characteristic of all terminal forcing sequences is the complete disregard for any offensive threat on the part of the opponent (with the notable exception of a 0-3 file). The formation of a related pair by the opposition during a

terminal forcing sequence does not alter the outcome. No opportunity will arise for the opponent to utilize the related pair and form a pair of 0-3 files.

In addition to terminal forcing sequences, other "weaker" move sequences leading to a win can be shown to exist. Recall the "knight's" position shown in Figure 6.1a. This 4-0 plane does not yield a terminal forcing sequence as it has no 2-0 files. Assume this plane is isolated from the cube and consider a move by X to cell 1 forming two 2-0 files (files 1 and 5). It can easily be verified that regardless of the opponent's reply to this move (unless it forms an 0-3 file) a terminal forcing sequence can be initiated on the next move. Similarly a move to cell 16 in the castle's position in Figure 6.1b will lead to a terminal forcing sequence on the following move. Many excess-three planes which are not states in a terminal forcing sequence have similar properties. Consider, for example, the 3-0 plane shown in Figure 6.7a. An X move to cell 6 results in a win regardless of the $O_1$ move. Win sequences for several $O_1$ moves are shown in Figure 6.7 b,c,d,e.

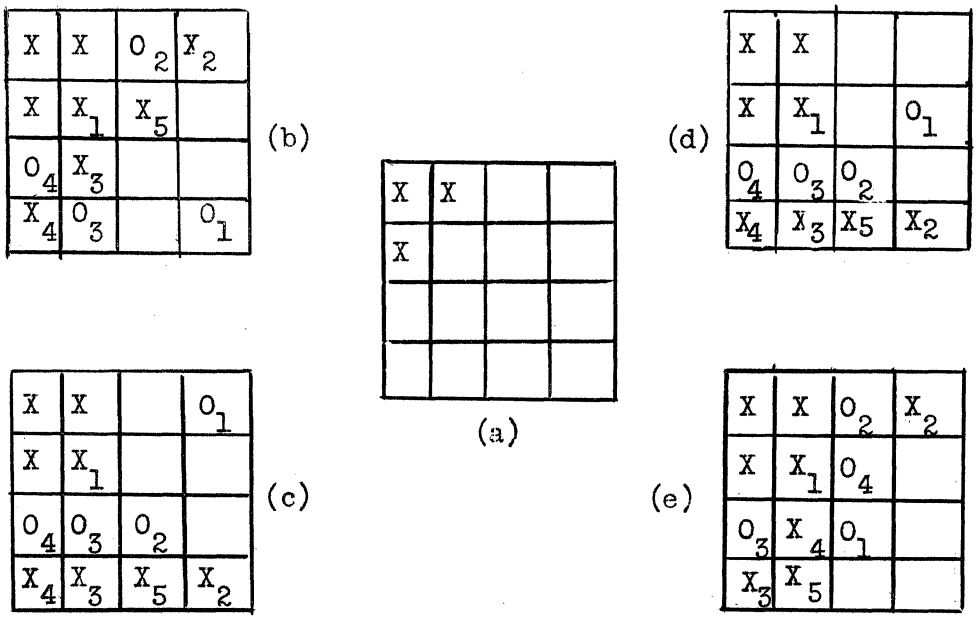



Figure 6.7

The obvious weakness in a non-forcing opening move lies in permitting the opponent to initiate his own terminal forcing sequence elsewhere in the cube during this delay.

A slightly "stronger" configuration of the non-terminal forcing sequence opening move type is shown in Figure 6.8a (The reader might note that this 3-0 plane actually yields a terminal forcing sequence, but this fact will be overlooked.) An X move to cell 16 forms a related pair forcing the $O_1$ move to cell 2, 4, 8, thus yielding a 4-1 plane which is the initial state in a terminal forcing sequence. The completed win possibilities are shown in Figure 6.8 b,c,d. Actually, however, the related pair formed by the X move does not prevent the opponent from initiating his own terminal forcing sequence

instead of blocking the related pair. In this sense it is no more effective than the example discussed in Figure 6.7. The advantage of the related pair force on the $O_1$ move is that it controls the opponents move to a few selected cells in the plane under consideration, rather than letting him build 2-O files in the cube which might have vacant cells in the plane.

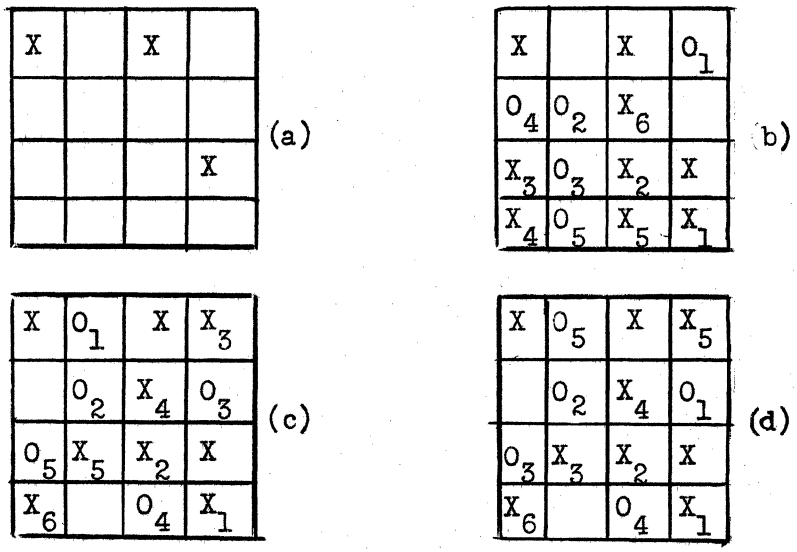



Figure 6.8

Planar situations may arise in which the related pair "force" occurs on the second or third move in the "pseudo-forcing" sequence. Consider the example in Figure 6.9a. The The opponents first move is forced by a 3-O file in file 7. The $O_2$ move, however, may be to either cell 10, 12 or 13 to block the related pair set up by files 3 and 10. Each of these three possible moves gives rise to a terminal forcing sequence for X as is shown in Figures 6.9 b,c,d, respectively.

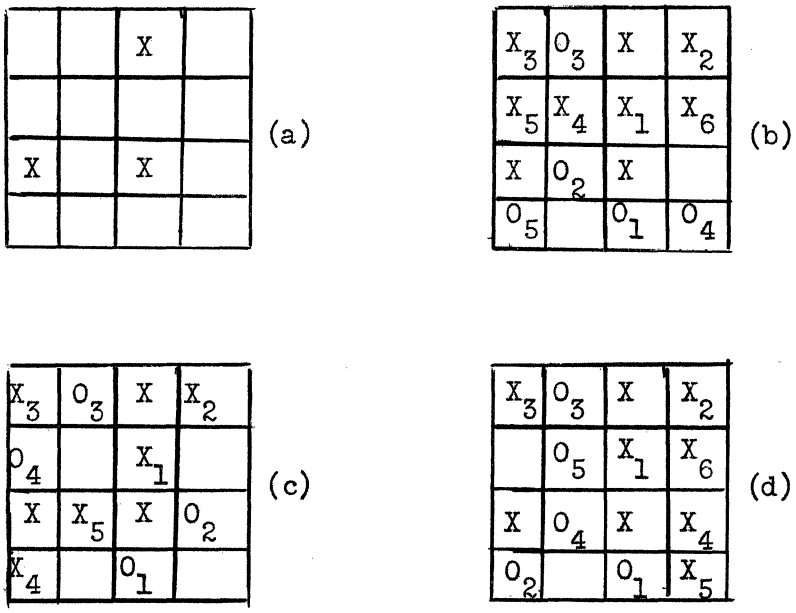



Figure 6.9

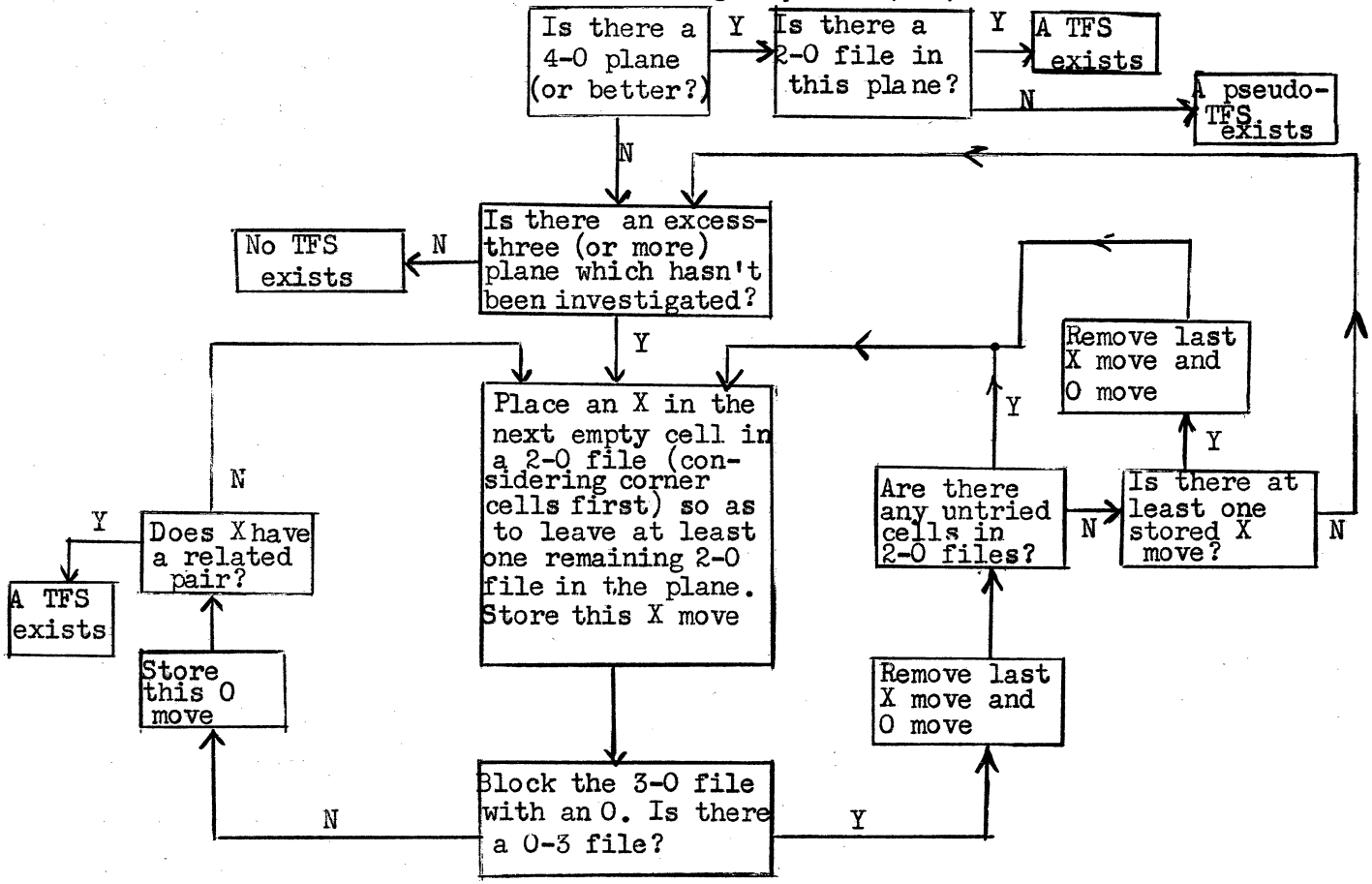
Let us term any move sequence of the types described in figures 6.7, 6.8, 6.9 which lead to a win as "pseudo-terminal forcing sequences". To summarize, a pseudo-terminal forcing sequence is a sequence of moves which leads to a win while containing a move which is not a strict force (e.g. a related pair force). A win sequence can be executed provided that the opponent cannot start his own terminal forcing sequence at the non-forcing move juncture.

A flow chart test to determine the existence of a terminal forcing sequence in an isolated plane is shown in Figure 6.10. A more practical test would include checking for the possibility of vacant cells in the plane which are contained in 2-0 or 0-2 files outside the plane as previously discussed. The flow

chart does not make use of Case 1 and Case 2 conditions of the terminal forcing sequences since these "rules" are sufficient but not necessary conditions. Instead, a planar "search technique" is utilized. Basically this technique may be outlined as follows:

1. Check the states of all 18 planes in the cube.
2. If there is a 4-0 plane a pseudo-terminal forcing sequence exists,and if the plane has a 2-0 file, a terminal forcing sequence is present. For 5-0 planes, 6-0 planes,etc.,a terminal forcing sequence always exists (since there is always at least one 2-0 file).
3. If there is an excess-three (or more) plane try all possibilities to manufacture a terminal forcing sequence. (This is done by forming a tree like structure of vacant cells in 2-0 files.)
4. Save move sequence of the terminal forcing sequence discovered for use in play so that the next move in the sequence will be readily available.

Figure 6.10. Terminal Forcing Sequence (TFS) Flow Chart



As an example the flow chart test is applied to the 3-0 planar situation shown in Figure 6.11a. Dots are used to signify vacant cells in 2-0 files. The step-by-step sequence leads to a related pair at cell 10 in Figure 6.11f and is quite smooth, the only setback occurring in Figure 6.11e where the opponent forms a 0-3 file. A great deal of extra work is saved by the heuristic which considers placing an X in vacant vertice cells of the plane which are in 2-0 files before other possibilities. Without this heuristic the reader might verify that the solution to this example is quite involved. Also, this introduces a considerable amount of "pruning" into the game tree, albeit at the sacrifice of some involved strategies of greater depth. A more complex example is shown in Figure 6.12a. Here the $X_2$ move is a poor one and must eventually be changed. Fourteen steps are required for this solution.

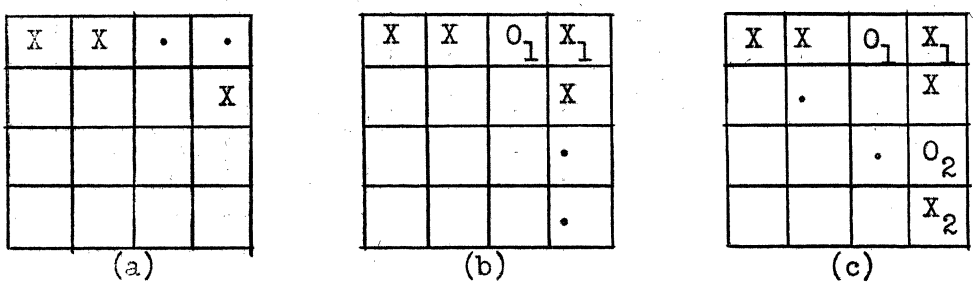



Figure 6.11

| | | | |
|---|---|---|---|
| X | X | $O_1$ | $X_1$ |
| · | $X_3$ | · | X |
| | | $O_3$ | $O_2$ |
| | | | $X_2$ |

(d)

| | | | |
|---|---|---|---|
| X | X | $O_1$ | $X_1$ |
| $X_4$ | $X_3$ | $O_4$ | X |
| · | · | $O_3$ | $O_2$ |
| · | · | | $X_2$ |

(e) O has a O-3 file, erase $X_4$ and $O_4$

| | | | |
|---|---|---|---|
| X | X | $O_1$ | $X_1$ |
| $O_4$ | $X_3$ | $X_4$ | X |
| | · | $O_3$ | $O_2$ |
| · | · | | $X_2$ |

(f) related pair at cell 10

Figure 6.11 (Continued)

| | | | |
|---|---|---|---|
| X | X | · | · |
| | · | | |
| | | X | |
| | | | · |

(a)

| | | | |
|---|---|---|---|
| X | X | $O_1$ | $X_1$ |
| | · | | |
| | | X | |
| | | | · |

(b)

| | | | |
|---|---|---|---|
| X | X | $O_1$ | $X_1$ |
| | $O_2$ | | · |
| | | X | · |
| | | | $X_2$ |

(c) no 2-O file, erase $X_3$

| | | | |
|---|---|---|---|
| X | X | $O_1$ | $X_1$ |
| | $O_2$ | | $X_3$ |
| | | X | |
| | | | $X_2$ |

(d) no 2-O file, erase $X_3$

| | | | |
|---|---|---|---|
| X | X | $O_1$ | $X_1$ |
| | $O_2$ | | $O_3$ |
| · | · | X | $X_3$ |
| | | | $X_2$ |

(e)

| | | | |
|---|---|---|---|
| X | X | $O_1$ | $X_1$ |
| · | $O_2$ | | $O_3$ |
| $X_4$ | $O_4$ | X | $X_3$ |
| · | | | $X_2$ |

(f)

Figure 6.12

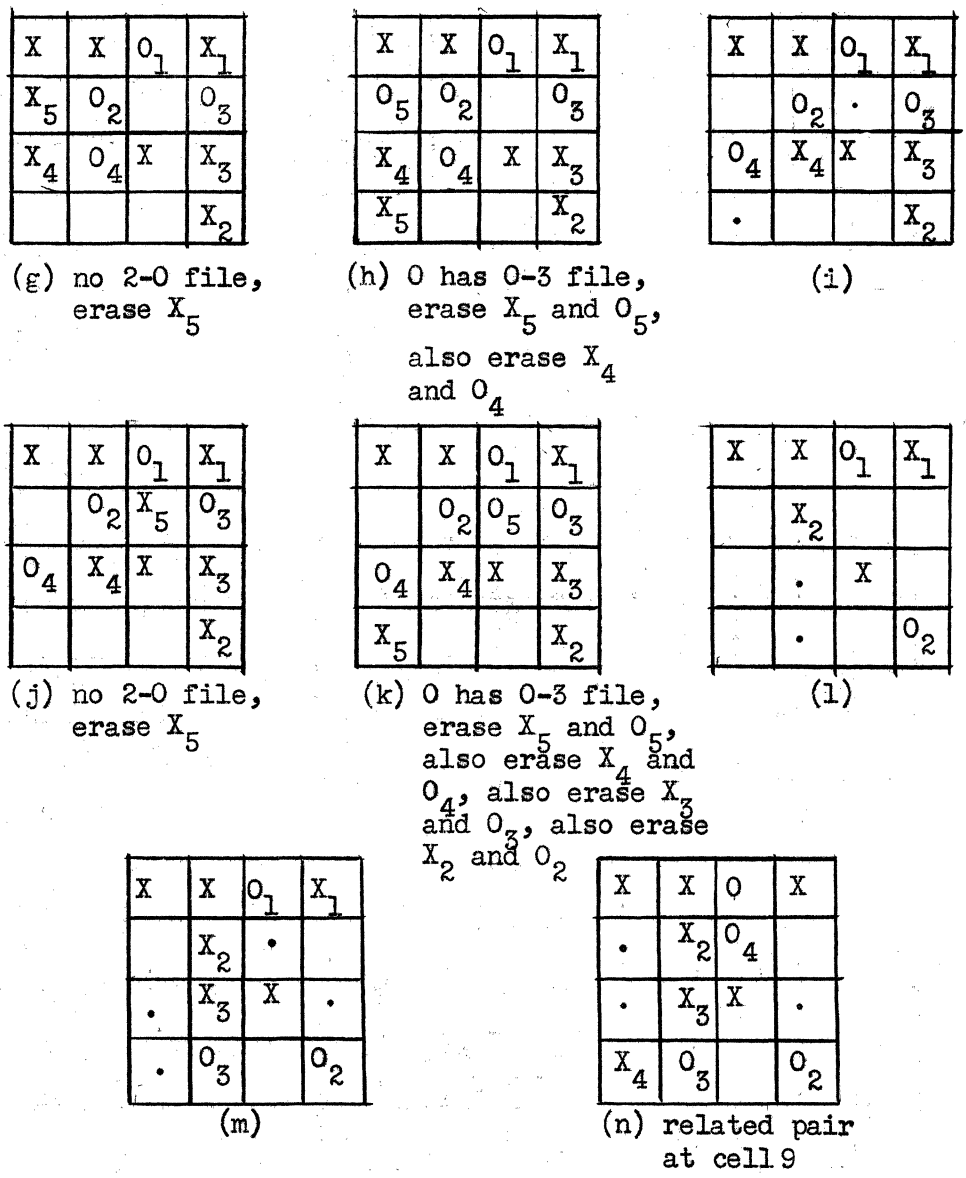



Figure 6.12 (Continued)

## 7. CUBIC STRATEGY

In this chapter we will consider a few game situations which lie beyond the planar realm but are relatively simple to detect. Elementary sequences of Case 1 and 2 terminal forcing sequences are of this nature. The lowest form of Case 1 has already been illustrated in Figure 5.3b. The specific conditions for this force may be stated as follows: If a 1-0 file intersects two 2-0 files at vacant cells, a terminal forcing sequence exists if no $\Omega_\beta$ state is formed by virtue of the opponent's blocking moves.

The check for this situation is relatively simple. It requires monitoring the short intersection sequences of 2-0 and 1-0 files in the cube. When applied to a planar situation it is possible that the Case 1 situation will be present even when there is no X-excess in the plane. In Figure 7.1 a 4-5 plane, an X move to cell 4 or 16 (vacant cells in a 1-0 file) simultaneously forms a 3-0 file and a related pair, yielding a win. Interesting configurations of this type in which the three files (two 2-0 files and one 1-0 file) are in different planes may occur.

| | | | |
|---|---|---|---|
| X | | X | · |
| O | | | X |
| O | O | X | |
| O | | O | · |

Figure 7.1

Another "cubic heuristic" may be derived as a direct result of those excess-three planes which do not yield terminal forcing sequences. Consider the situation in which two intersecting planes are excess-three and excess-two respectively. Further assume that the excess-three plane does not yield a terminal forcing sequence and that there are no cells in the excess-two planes which are vacant cells in 2-0 files outside the plane. It is entirely possible that a sequence of forces in the excess-three plane exists which builds the excess-two planes into an excess-three (or more) plane yielding a terminal forcing sequence. Such a force sequence might be termed a"plane-plane terminal forcing sequence". This result may be expanded to multiple plane intersections but the extension will not be considered here due to the complexities and search time involved. Actually, a multiple plane intersection situation leading to a terminal forcing sequence might be more easily described in terms of a Case 1 or 2 terminal forcing sequence anyway.

## 8. SUMMARY - GENERAL MOVE STRATEGY

At this point let us piece together the fragments of strategy discussed in the preceding chapters. The goal of winning may be replaced by pursuing a wider objective or sub-goal, achieving a state in a terminal forcing sequence. The problem is thus - How does one achieve a state in a terminal forcing sequence? This question is not easily answered. Perhaps the outstanding characteristic of a forcing sequence in the overall cube is the presence of intersecting 2-0 and/or 1-0 files. In the plane the sequence may be recognized by the pattern of occupied cells or by the excess-cell advantage. The objective in both cases (cubic and planar) centers around the early formation of 2-0 and 1-0 files (which in turn lead to the build up of plane excesses). Thus it appears that a good opening move in a play of the game might be to a cell which lies in seven files and six planes. (Recall that this characteristic is true of the eight vertice and eight core cells of the cube. All other cells lie in four files and four planes.)

Although there is no concrete theoretical evidence, it appears from empirical observation that the player who opens the play has a definite advantage over his opponent. (This conclusion is quite sound intuitively since the first player

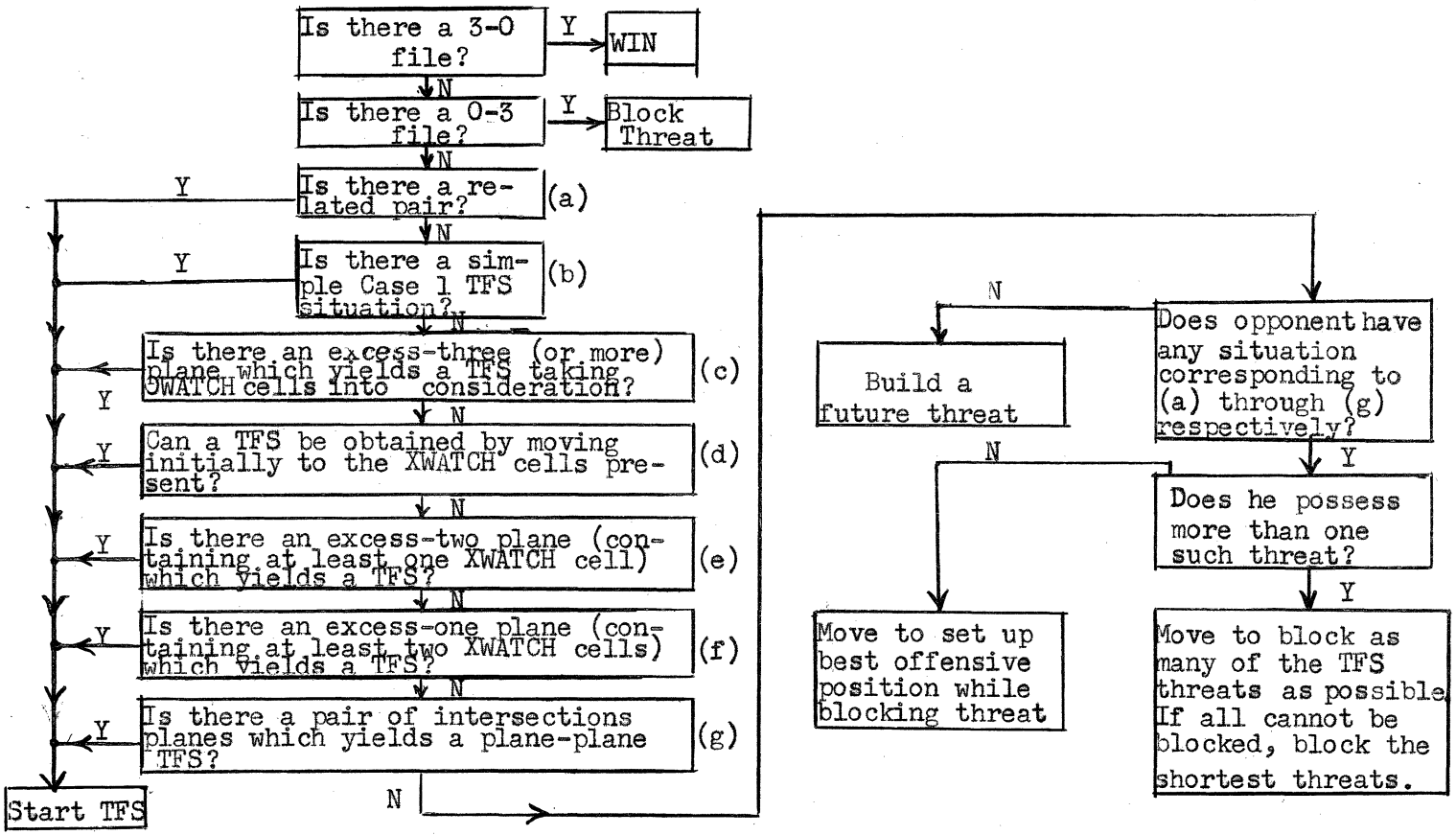
is the only one to ever have an advantage in the number of occupied cells.) Thus, it is probably best for the second player to place more emphasis on defense than offense at least in his first few opening moves. A good reply to the opening move, should the first player occupy one of the sixteen cells mentioned, might be a mirror image to this move through the axes of symmetry of the cube. This would block one of the seven 0-1 files and generate six 1-0 files at the same time. An "offensive" move in which no 0-1 files are blocked but seven 1-0 files are formed might be a better choice if the first player did not move to one of the sixteen "choice" cells.

After the first several moves, several questions should be considered while pondering a proper move. These questions, constituting a general strategy, are posed in the flow chart in Figure 8.1. This chart represents a far more sophisticated strategy than that shown in Figure 4.1. Several of the mechanical questions (i.e. how many moves are on the board) introduced in Figure 4.1 are not repeated in this flow chart although they still are pertinent. The term XWATCH cells used in the flow chart refers to vacant cells in the plane under consideration which are in 2-0 files outside the plane. Similarly, OWATCH cells are such vacant cells in the 0-2 files.

The final block in the flow chart - "Build a future threat" - may be fulfilled by either of two possibilities. First, the move might be made to an excess-plane which does not presently yield a terminal forcing sequence by forming a new 2-0 file without making a 3-0 file. When possible the move should be made at the intersection of two or more excess-planes so that the resulting threat will be more formidable and better disguised. The second possibility is to form a 2-0 file in the cube while moving into a plane in which the opponent enjoys an occupied cell excess. A defensive move of this type might be advisable in a game in which the opponent is known to be a strong player and has the advantage of moving first.

The relationship between offense and defense is clearly evident on the flow chart. If a win is not immediately available, a check must be made on the corresponding chances of the opponent. Should neither player have a 3-0 file available, an offensive check is made for a terminal forcing sequence. If none is found to be available, a defensive check is made checking the opponents possibilities. If no terminal forcing sequence is found on either the offense of defense, a "building" move is made as described in the previous paragraph.

Figure 8.1: General Move Strategy Flow Chart



## 9. COMPUTER APPLICATION AND RESULTS

After an initial period of debugging, a PDP-4/340 D Computer-Display combination was programmed in MIDAS 4 Assembly Language to play three dimensional tic-tac-toe on an elementary level. The machine's capability is limited to the strategy shown in the flow chart in Figure 4.1. While this appears to be quite a strong limitation in view of the powerful terminal forcing sequence techniques presented, the machine has been quite successful against human opposition. This success is probably influenced by the lack of an actual three dimensional model of the cube for use by the opponent. A four level two-dimensional view (cutting the 64 cells into four disjoint 4x4 planes) is displayed on the 340 D unit and serves as the model. In addition, an isometric view of the cube showing all 64 cells is displayed. The opponent's moves are instigated by pointing a "light pen" (photocell) attached to the display unit to the desired cell in the four-level view. Immediately upon triggering the light pen both the opponents move and the computer's countermove appear simultaneously. In addition, a message appears on the display to explain why the computer made the move it did (i.e. blocked a related pair, etc.) As the flow chart shows, the computer makes a "preset"

move when no threat of a win or related pair exists on either offense or defense. The preset move is the first vacant cell found on a list of the 64 cells randomly ordered for each game.

Attempts by opponents to obtain suitable positioning for terminal forcing sequences are quite often thwarted by unintentionally "good" preset moves. Also, in a zest to fortify their own offensive position and in deference to the random quality of the machine's moves, many opponents disregard defense. These factors have enabled the computer to build up a large winning margin, probably exceeding 85%. This statistic is even more impressive as the computer always moves last. Furthermore, the computer rarely loses on its first play against a particular opponent. Its infrequent losses usually occur only after the opponent realizes the limitations of the program and sets his strategy accordingly. Against novice opponents the computer rarely requires more than sixteen moves (thirty-two occupied cells) to complete a win. The state of the play is sufficiently complex at this point for many players to fail to recognize a related pair. Naturally, the computer's logic mechanisms are not effected by the complexity of the cubic configuration (other than in move time required - which is

nearly negligible in the related pair strategy).

A more sophisticated program has been written in ALGOL 60 to play using some of the techniques discussed in the section on planar strategy. The computer used in this instance is the Univac 1107. This machine has not been pitted against human opposition due to the input-output complexities inherent in the available system. A complete play of the game would require a tie-up of the computer for an excessive period of time. Thus, as an alternative, the computer has been programmed to play against itself, using the same strategies in each instance. An entire play of the game, start to finish, usually takes less than one minute. Briefly, the capabilities of the program include:

a. Elementary strategy - as used by the PDP-4 program.

b. Planar strategy - including both excess-two and excess-three (or more terminal forcing sequences with XWATCH and OWATCH provisions. Not including pseudo terminal forcing sequences.

c. Building strategy - building excess-two planes into excess-three planes which are possible threats as TFS.

As in the PDP-4 program, the opening moves are made from a preset move list. Many complete plays have been analyzed and the results are quite interesting. A minimum of two moves

from the preset move list is made by each player at the opening of the play. If the moves form a 2-0 plane, the building strategy takes over from there. If not, a few additional preset moves might be necessary. No instance has been encountered in which either player opened with more than four preset moves. "Strong" opening combinations (i.e. those complying to conditions specified in the last section) by the first player almost invariably lead to a win. Even "weak" opening combinations by the first player yield a high winning percentage, the wins usually resulting from correspondingly "weak" openings by the second player. As might be expected, strong openings by the second player in reply to weak openings by the first player often lead to a win by the second player.

A particularly interesting result of a computer versus computer play was the achievement of a tie-game. It is the first such occurrence the author has ever encountered. Daly[1] states that neither he nor any of his associates has ever seen a legitimate tie-game and that he is not certain such a play is even possible. The complete play of this unusual game is given in Appendix III. It is quite fascinating to note that related pair and 3-0 files are being formed and blocked even as the final cube positions

are being filled.

In a few instances, the computer accidentally formed multiple related pairs simultaneously. A game in which this occurs is also shown in Appendix III. A strategy to locate such possibilities has not been drawn up as the time necessary for its execution might prove excessive.

## 10. CONCLUSIONS

A minimax procedure becomes essential in the solution of a game situation when some intermediate evaluation scheme is used in conjunction with limitation of depth of search. However, in the strategies under consideration the limitation of depth is unnecessary because of the extremely efficient tree-pruning heuristics used at extrapolations while the intermediate evaluation schemes (if they can be so called) used allow for single ply exploration. The major heuristic used for investigating win sequences is the technique of "forcing". A state in a terminal forcing sequence is one at the opponent's move time such that one of the opponent's alternatives gives rise to an immediate win for player X while the other give rise to a future win. By restricting the search to looking only for sequences of forces one certainly throws away the possibilities of some very subtle maneuvers in the game (i.e. strings of related pairs), but the sacrifice seems worthwhile in view of the fact that the exploration of forcing situations takes place along drastically pruned trees and the need for depth limitations is avoided. Moreover, heuristics are available for the creation of situations in which the possibility of force situations may be highly probable.

Another drastic reduction occurs in the search process for forcing situations by limiting the search, in most cases, to moves made in a suitably chosen fixed plane. The choice of the plane is made on the basis of the bases of the excess of X cells over the opponent's occupied cells. Actual enumeration has shown that in an isolated 3-0 plane, in 40 out of 77 cases (counting symmetrical situations as identical) a terminal forcing sequence is possible. This possibility continues even with a few of the opponents pieces on the plane as long as the difference is three or more. The trend certainly does not continue all the way to (ridiculously) the 9-6 ratio, but it is of benefit to use the heuristic that an excess-three is worth concentrating on. This heuristic applied to 2-0 planes and XWATCH cells is also quite valuable. While developing a force on a plane, the machine has to guard against the formation of a threat by the opponent through his defensive moves. These threats may consist of a 0-3 file either in the plane or outside it. This is one consideration which prevents the complete concentration of the efforts on a plane. However, since these entail only one-step explorations outside the plane, they do not introduce any exponential growth of the tree with the depth of search.

The machine has capabilities of blocking all tactics on the part of the opponent that the machine uses for its own. In addition to the depth searches, searches for related pairs and 3-0 files are carried out in a routine manner and taken advantage of as well as blocked. The one large failure on the part of the machine lies in its inability to either foresee or block forces of depth greater than one on both sides. However, since such cases have necessarily to develop in three dimensions (the plane case being taken care of by the planar terminal forcing sequence tactics), the machine would probably not suffer from this while playing humans. When no forcing sequence presents itself to the machine, the strategy consists of creating a plane with an excess of three. This may be considered the only form of intermediate evaluation used in the program.

No other intermediate evaluation schemes being available, the machine plays from a table of preset moves when none of the possibilities engendered above are available. Such possibilities open up in about three moves or six plies. Hence the fate of the entire game (when the machine plays itself) is determined from the first few initial moves. We have at present no statistics on the efficiency

of these initial moves, nor do we have any proven efficient heuristic for choosing them. It is our hope that the two or three plies made by the human player will be conducive to the machine's ones only on a random basis, but in most games the machine will be able to capitalize on the possibilities opened up in the first few moves better than a human player.

Although some of the move sequences resulting from computer versus computer games in the 1107 program are quite interesting, it is difficult to classify the strategy as being exceptionally "good". It is obviously inferior to the strategy shown in the flow chart in Figure 8.1 since it includes fewer heuristics. As more and more heuristics are added, the machine will play "better", but a trade-off must be considered between "goodness of play" and move time. The present 1107 program requires a move time of a few seconds or less.

The results of the computer versus computer plays leave a clear indication that the first player should win if the play is not a draw. The strategy which will insure this has not been found, but the planar terminal forcing sequence heuristic when used in conjunction with "strong" opening moves comes quite close. It is inconclusive

whether or not a complete play of the game using minimax techniques would result in a draw. This will probably never be known for three dimensional tic-tac-toe because of the gigantic number of explorations involved.

A main weakness of the game as programmed lies in the absence of a strategically based opening move sequence. Perhaps too much emphasis has been placed on terminal forcing sequences in lieu of this phase of the play. The opening move strategy would certainly be a prime place to devote further effort.

It may appear that the program discussed here is specifically designed for three dimensional tic-tac-toe. However, it must be pointed out that the strategy of forcing is applicable in the end-portions of many games[16]. The great effect this strategy has on tree-pruning is worthy of note in the general case. It can also be pointed out that the general applicability of the strategically based state set argument in Chapter 2 is born out by the tic-tac-toe terminal forcing sequence conditions in Chapter 5.

## APPENDIX I: A PROCEDURE FOR DETERMINING RELATED PAIRS

The recognition of a related pair can be achieved by a simple heuristic which may be used by man and machine alike. To describe this heuristic properly, let us make several preliminary definitions. A "file set" is the group of files which passes through any given cell. The cell that the file set passed through is the "file set basis". A related pair can be described as two 2-0 files (or 0-2 files) in the same file set having a vacant file set basis. The file set basis in this case is referred to as a "major cell", and can be designated as "offensive" when the files are 2-0 and "defensive" when the files are 0-2.

It is clear that any of the 64 cells may serve as major cells and that any individual vertice or core cell has a comparatively greater probability (7 to 4) of being a major cell than do other cells.

Three possible heuristic methods for finding related pair may be stated as follows:

1. Check the file sets associated with all occupied X cells. If any of the files in one of these file sets is found to be a 2-0 file, record the vacant cells in that file. If one of the recorded cells appears again as a vacant cell in another 2-0 file, it is a major cell.

Repeat the test for all occupied Y cells and corresponding 0-2 files.

2. Check the file sets associated with all vacant cells. If two (or more) of the files in a file set contain 2-0 files or 0-2 files, the file set basis is a major cell.

3. Check the 76 files for 2-0 files. If any are found, record the vacant cells in those files. If any cell is recorded twice, it is a major cell.

Although each of these procedures may be useful in the course of a game, procedure (3) is the best from both time and relative efficiency standpoints. Procedure (1) is an excellent one to follow in the first several moves of the play since it is only concerned with occupied cells. However, much of its usefulness is wasted by virtue of the fact that a related pair cannot be formed until one player has made at least four moves. In a similar manner, procedure (2) can be applied in a play in which few vacant cells remain on the board. Such a play is quite rare in reality. Application of procedure (3) requires only 76 file checks regardless of the number of moves on the board. The cells contained in each of the 76 files are listed in Table I, and the file sets for any given file set basis are given in Table II. The information contained in Tables I and II,

when used for the purpose of applying the "related pair strategy" is sufficient to yield a non-trivial play of the game. This fact has been verified by the performance of the PDP-4 computer against human opposition.

Table I: Cells Contained in Each File

| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 112 | 113 | 114 | 111 | 121 | 131 | 141 | 111 | 114 | 211 | 212 | 213 | 214 |
| 122 | 123 | 124 | 112 | 122 | 132 | 142 | 122 | 123 | 221 | 222 | 223 | 224 |
| 132 | 133 | 134 | 113 | 123 | 133 | 143 | 133 | 132 | 231 | 232 | 233 | 234 |
| 142 | 143 | 144 | 114 | 124 | 134 | 144 | 144 | 141 | 241 | 242 | 243 | 244 |
| 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 |
| 211 | 221 | 231 | 241 | 211 | 214 | 311 | 312 | 313 | 314 | 311 | 321 | 331 |
| 212 | 222 | 232 | 242 | 222 | 223 | 321 | 322 | 323 | 324 | 312 | 322 | 332 |
| 213 | 223 | 233 | 243 | 233 | 232 | 331 | 332 | 333 | 334 | 313 | 323 | 333 |
| 214 | 224 | 234 | 244 | 244 | 241 | 341 | 342 | 343 | 344 | 314 | 324 | 334 |
| 27 | 28 | 29 | 30 | 31 | 32 | 33 | 34 | 35 | 36 | 37 | 38 | 39 |
| 341 | 311 | 314 | 411 | 412 | 413 | 414 | 411 | 421 | 431 | 441 | 411 | 414 |
| 342 | 322 | 323 | 421 | 422 | 423 | 424 | 412 | 422 | 432 | 442 | 422 | 423 |
| 343 | 333 | 332 | 431 | 432 | 433 | 434 | 413 | 423 | 433 | 443 | 433 | 432 |
| 344 | 344 | 341 | 441 | 442 | 443 | 444 | 414 | 424 | 434 | 444 | 444 | 441 |
| 40 | 41 | 42 | 43 | 44 | 45 | 46 | 47 | 48 | 49 | 50 | 51 | 52 |
| 111 | 112 | 113 | 114 | 121 | 122 | 123 | 124 | 131 | 132 | 133 | 134 | 141 |
| 211 | 212 | 213 | 214 | 221 | 222 | 223 | 224 | 231 | 232 | 233 | 234 | 241 |
| 311 | 312 | 313 | 314 | 321 | 322 | 323 | 324 | 331 | 332 | 333 | 334 | 341 |
| 411 | 412 | 413 | 414 | 421 | 422 | 423 | 424 | 431 | 432 | 433 | 434 | 441 |
| 53 | 54 | 55 | 56 | 57 | 58 | 59 | 60 | 61 | 62 | 63 | 64 | 65 |
| 142 | 143 | 144 | 111 | 121 | 131 | 141 | 111 | 141 | 114 | 124 | 134 | 144 |
| 242 | 243 | 244 | 212 | 222 | 232 | 242 | 222 | 232 | 213 | 223 | 233 | 243 |
| 342 | 343 | 344 | 313 | 323 | 333 | 343 | 333 | 323 | 312 | 322 | 332 | 342 |
| 442 | 443 | 444 | 414 | 424 | 434 | 444 | 444 | 414 | 411 | 421 | 431 | 441 |
| 66 | 67 | 68 | 69 | 70 | 71 | 72 | 73 | 74 | 75 | 76 | | |
| 114 | 144 | 111 | 112 | 113 | 114 | 141 | 142 | 143 | 144 | 111 | | |
| 223 | 233 | 221 | 222 | 223 | 224 | 231 | 232 | 233 | 234 | 121 | | |
| 332 | 322 | 331 | 332 | 333 | 334 | 321 | 322 | 323 | 324 | 131 | | |
| 441 | 411 | 441 | 442 | 443 | 444 | 411 | 412 | 413 | 414 | 141 | | |

Table II: File Sets Corresponding to Each Major Cell

| 111 | 114 | 144 | 141 | 223 | 222 | 232 | 233 | 323 | 322 | 332 | 333 | 411 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 76 | 3 | 3 | 76 | 12 | 11 | 11 | 12 | 22 | 21 | 21 | 22 | 30 |
| 4 | 4 | 7 | 7 | 15 | 15 | 16 | 16 | 25 | 25 | 26 | 26 | 34 |
| 8 | 9 | 8 | 9 | 19 | 18 | 19 | 18 | 29 | 28 | 29 | 28 | 38 |
| 40 | 43 | 55 | 52 | 46 | 45 | 49 | 50 | 46 | 45 | 49 | 50 | 40 |
| 56 | 62 | 65 | 59 | 63 | 57 | 58 | 64 | 57 | 63 | 64 | 58 | 62 |
| 60 | 66 | 67 | 61 | 66 | 60 | 61 | 67 | 61 | 67 | 66 | 60 | 67 |
| 68 | 71 | 75 | 72 | 70 | 69 | 73 | 74 | 74 | 73 | 69 | 70 | 72 |
| 414 | 444 | 441 | 112 | 113 | 124 | 123 | 122 | 121 | 131 | 132 | 133 | 134 |
| 33 | 33 | 30 | 1 | 2 | 3 | 2 | 1 | 76 | 76 | 1 | 2 | 3 |
| 34 | 37 | 37 | 4 | 4 | 5 | 5 | 5 | 5 | 6 | 6 | 6 | 6 |
| 39 | 38 | 39 | 41 | 42 | 47 | 9 | 8 | 44 | 48 | 9 | 8 | 51 |
| 43 | 55 | 52 | 69 | 70 | 63 | 46 | 45 | 57 | 58 | 49 | 50 | 64 |
| 56 | 59 | 65 | - | - | - | - | - | - | - | - | - | - |
| 61 | 60 | 66 | - | - | - | - | - | - | - | - | - | - |
| 75 | 71 | 68 | - | - | - | - | - | - | - | - | - | - |
| 143 | 142 | 211 | 212 | 213 | 214 | 224 | 221 | 231 | 234 | 244 | 243 | 242 |
| 2 | 1 | 10 | 11 | 12 | 13 | 13 | 10 | 10 | 13 | 13 | 12 | 11 |
| 7 | 7 | 14 | 14 | 14 | 14 | 15 | 15 | 16 | 16 | 17 | 17 | 17 |
| 54 | 53 | 18 | 41 | 42 | 19 | 47 | 44 | 48 | 51 | 18 | 54 | 53 |
| 74 | 73 | 40 | 56 | 62 | 43 | 71 | 68 | 72 | 75 | 55 | 65 | 59 |
| 241 | 311 | 312 | 313 | 314 | 324 | 321 | 331 | 334 | 344 | 343 | 342 | 341 |
| 10 | 20 | 21 | 22 | 23 | 23 | 20 | 20 | 23 | 23 | 22 | 21 | 20 |
| 17 | 24 | 24 | 24 | 24 | 25 | 25 | 26 | 26 | 27 | 27 | 27 | 27 |
| 19 | 28 | 41 | 42 | 29 | 47 | 44 | 48 | 51 | 28 | 54 | 53 | 29 |
| 52 | 40 | 62 | 56 | 43 | 75 | 72 | 68 | 71 | 55 | 59 | 65 | 52 |
| 412 | 413 | 424 | 423 | 422 | 421 | 431 | 432 | 433 | 434 | 443 | 442 | |
| 31 | 32 | 33 | 32 | 31 | 30 | 30 | 31 | 32 | 33 | 32 | 31 | |
| 34 | 34 | 35 | 35 | 35 | 35 | 36 | 36 | 36 | 36 | 37 | 37 | |
| 41 | 42 | 47 | 39 | 38 | 44 | 48 | 39 | 38 | 51 | 54 | 53 | |
| 73 | 74 | 57 | 46 | 45 | 63 | 64 | 49 | 50 | 58 | 70 | 69 | |

## APPENDIX II: TERMINAL FORCING SEQUENCES IN ISOLATED 3-0 PLANES

As the terminal forcing sequence has been set forth as the "strongest" strategy available in a given plane of the cube, its earliest occurance is worthy of investigation. This situation in which reasonable doubt in first encountered with respect to terminal forcing sequences is in the 3-0 plane. (Recall that 1804 of the 1820 4-0 planar configurations lead to pseudo-terminal forcing sequences.) Of the 560 different configurations arising from three occupied cells in a set of 16, 77 can be shown to be independent after symmetry considerations. The 77 configurations consist of 63 having eight symmetries and fourteen having four symmetries. The symmetries consist of two "diagonal flips" for each of four $90^{\circ}$ rotations. An example of the eight symmetries for a simple four cell plane is shown in Figure II.1. The $90^{\circ}$ rotations are shown in a-d and the "diagonal flips" for each of these rotations in e-h, respectively. Those 3-0 planes which have only four symmetries are the ones which are symmetrical about a diagonal.

A program written in ALGOL 60 for the Univac 1107 for generating the 77 3-0 planes is shown at the end of this appendix. A simplified flow chart of the

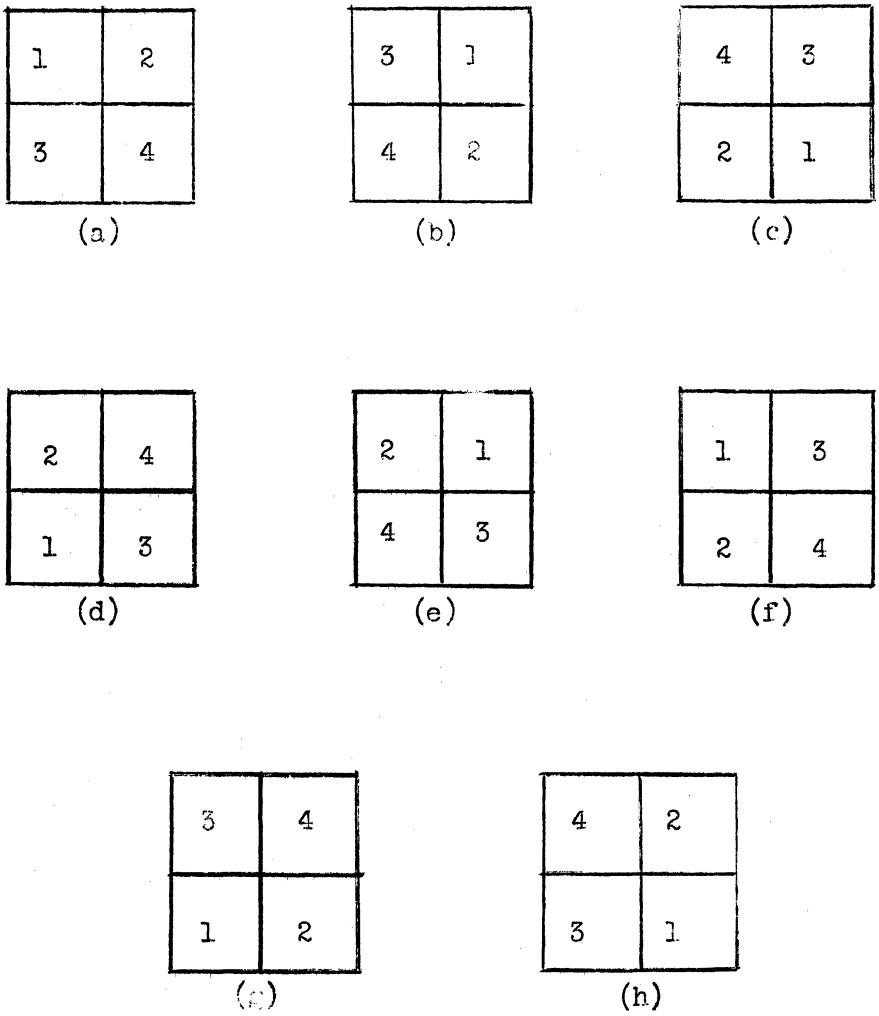



Figure II.1

Table III: List of the 77 3-0 Planes

| Cells in Plane | No.of 2-0 Files | TFS? | Possible Opening Moves | Cells in Plane | No. of 2-0 Files | TFS? | Possible Opening Moves |
|---|---|---|---|---|---|---|---|
| 1, 2, 9 | 2 | Yes | 4,13 | 1, 9,15 | 1 | Yes | 13 |
| 1, 2,10 | 2 | Yes | 4, 6 | 1,10,11 | 2 | Yes | 9,16 |
| 1, 2,11 | 2 | Yes | 4, 6 | 1,10,14 | 1 | Yes | 6 |
| 1, 2,13 | 2 | Yes | 4, 9 | 1,10,15 | 0 | No | - |
| 1, 2,14 | 2 | No | - | 1,11,14 | 1 | No | - |
| 1, 2,15 | 1 | No | - | 1,11,15 | 2 | Yes | 7,16 |
| 1, 2,16 | 2 | Yes | 4, 6 | 1,12,14 | 0 | No | - |
| 1, 3,13 | 2 | Yes | 4, 5 | 1,14,15 | 1 | No | - |
| 1, 3,14 | 1 | No | - | 2, 3,14 | 2 | No | - |
| 1, 3,15 | 2 | No | - | 2, 5,10 | 1 | No | - |
| 1, 3,16 | 2 | Yes | 4,11 | 2, 5,14 | 1 | No | - |
| 1, 5, 9 | 1 | Yes | 13 | 2, 5,15 | 0 | No | - |
| 1, 5,10 | 1 | Yes | 13 | 2, 6, 9 | 1 | Yes | 10 |
| 1, 5,14 | 1 | Yes | 13 | 2, 6,10 | 1 | Yes | 14 |
| 1, 5,15 | 1 | No* | 6 | 2, 6,11 | 2 | Yes | 1,10 |
| 1, 6, 9 | 2 | Yes | 13 | 2, 6,14 | 1 | Yes | 10 |
| 1, 6,10 | 2 | Yes | 2,11,14,16 | 2, 6,15 | 1 | No | - |
| 1, 6,14 | 2 | Yes | 10,16 | 2, 7,14 | 1 | No | - |
| 1, 6,15 | 1 | No | - | 2, 7,15 | 1 | No | - |
| 1, 7,14 | 0 | No | - | 2,10,11 | 2 | Yes | 6 |
| 1, 7,15 | 1 | Yes | 11 | 2,12,14 | 1 | No | - |
| 1, 9,10 | 2 | Yes | 11,13 | 2,13,16 | 1 | No | - |
| 1, 9,11 | 3 | Yes | 10,13 | 5, 6,15 | 1 | No* | 1, 7 |
| 1, 9,14 | 1 | No* | 4 | 5,12,14 | 0 | No | - |
| | | | | The planes below -4 symmetries | | | |
| 5,14,15 | 1 | No | - | 1, 2, 5 | 2 | No* | 6 |
| 6,10,15 | 1 | No | - | 1, 3, 9 | 2 | No* | 11 |
| 6,11,14 | 2 | Yes | 10 | 6, 7,10 | 3 | Yes | 2,14 |
| 6,12,14 | 1 | Yes | 10 | 1, 4,13 | 3 | Yes | 2,3,5,7,9,10 |
| 6,13,16 | 2 | Yes | 1,14,15 | 1, 6,11 | 1 | Yes | 16 |
| 6,14,15 | 2 | No | - | 1, 6,16 | 1 | Yes | 11 |
| 7,12,14 | 0 | No | - | 1, 7,10 | 1 | Yes | 4,13 |
| 7,14,16 | 1 | Yes | 13 | 1, 7,16 | 1 | Yes | 6,11 |
| 9,11,15 | 2 | Yes | 7 | 1, 8,14 | 0 | No | - |
| 9,14,15 | 1 | No | - | 1,12,15 | 0 | No | - |
| 10,11,14 | 1 | Yes | 6 | 2, 5, 6 | 2 | No* | 1 |
| 10,13,16 | 2 | Yes | 4,14,15 | 2, 5,11 | 0 | No | - |
| 10,14,15 | 2 | No | - | 3, 5, 9 | 0 | No | - |
| 11,12,16 | 3 | Yes | 1, 4 | 3, 9,11 | 2 | No* | 1,7,10 |
| 13,14,16 | 1 | Yes | 15 | | | | |

Table IV: Summary of Table III

| | Number of 2-0 files in plane | Total No. of planes of this type | Number of TFS planes of this type |
|---|---|---|---|
| Eight-Symmetry 3-0 planes | 3 | 2 | 2 |
| | 2 | 24 | 19 |
| | 1 | 31 | 13 |
| | 0 | 6 | 0 |
| | Totals | 63 | 34 |
| Four Symmetry 3-0 planes | 3 | 2 | 2 |
| | 2 | 4 | 0 |
| | 1 | 4 | 4 |
| | 0 | 4 | 0 |
| | Totals | 14 | 6 |
| All 77 3-0 planes | 3 | 4 | 4 |
| | 2 | 28 | 19 |
| | 1 | 35 | 17 |
| | 0 | 10 | 0 |
| | Totals | 77 | 40 |

program is shown in Figure II.2. In addition, a program is given for generating the different possible 4-0 planes. The "transformation" operations the programs refer to are the rotations and flips used to obtain the eight planar symmetries.

The cell configurations of the 77 3-0 planes are listed in Table III (with cells numbered as in Figure 5.2). Also the number of 2-0 files in the plane and the existance or absence of a terminal forcing sequence are specified. When a terminal forcing sequence exists, all possible opening moves leading to the win are given. An asterisk indicates that a pseudo-terminal forcing sequence is known to exist. As no exhaust search has been made for pseudo forces, this list is by no means complete in this respect.

A summary of the information presented in Table III is shown in Table IV. From Table IV it can be seen that all 3-0 planes having three 2-0 files yield a terminal forcing sequence. As the number of initial 2-0 files decreases, the probability of a terminal forcing sequence in the eight-symmetry planes lessens. In the four-symmetry planes, however, all planes which initially contain either one or three 2-0 yield

a force while those which contain zero or two 2-0 files do not. This result is most unusual.

While the information in Table 4 was not utilized by the existing computer program, it gives rise to further heuristics which may prove to be reliable. (i.e. All four-symmetry 3-0 planes having only one 2-0 file lead to a terminal forcing sequence.)

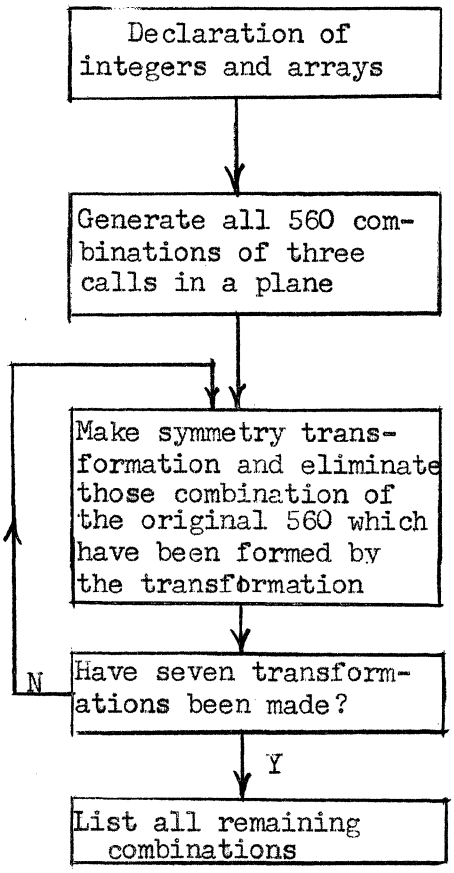



Figure II.2. Flow Chart of ALGOL Program for 3-0 Planes

```
ALG PROB1
    ALGOL        MARCH 10,1965          INTERFACE   FEBRUARY 15,1965     PASS2   DECEMBER 23, 1964
         1       COMMENT THIS PROGRAM IS DESIGNED TO FIND ALL THE POSSIBLE WAYS
         2          AFTER SYMMETRY CONSIDERATIONS THAT THREE OCCUPIED CELLS CAN BE
         3          PLACED IN A PLANE OF SIXTEEN CELLS.$
         4       INTEGER I,J,K,Q,Z,R$
BLOCK 1     LEVEL 1                                                                                    B0
         5       INTEGER ARRAY A,B,C,E,F,G(1..560),MZ,MA,MB,MC,MD,ME,MF(0..16)$
         6       SWITCH L=LA,LB,LC,LD,LE,LF,LG$
         7       READ (MZ,MA,MB,MC,MD,ME,MF)$
         8              WRITE('FIRST TRANSFORMATION',MZ)$
         9             WRITE('SECOND TRANSFORMATION',MA)$
        10              WRITE('THIRD TRANSFORMATION',MB)$
        11              WRITE('FOURTH TRANSFORMATION',MC)$
        12              WRITE('FIFTH TRANSFORMATION',MD)$
        13              WRITE('SIXTH TRANSFORMATION',ME)$
        14              WRITE('SEVENTH TRANSFORMATION',MF)$
        15       FOR I=(1,1,14) DO BEGIN FOR J=(2,1,15) DO BEGIN FOR K=(3,1,16) DO BEGIN                   B1    B2
        16       IF I LSS J AND J LSS K THEN BEGIN Q=Q+1$ A(Q)=I$ B(Q)=J$ C(Q)=K END END                   B3    B4    E4    E3
        17          END END$                                                                               E2    E1
        18           FOR I=(1,1,560) DO BEGIN E(I)=MZ(A(I))$F(I)=MZ(B(I))$G(I)=MZ(C(I))                    B5
        19           END$ GO TO R5$                                                                         E5
        20       LA..FOR I=(1,1,560) DO BEGIN E(I)=MA(A(I))$F(I)=MA(B(I))$G(I)=MA(C(I))                    B6
        21           END$ GO TO R5$                                                                         E6
        22       LB..FOR I=(1,1,560) DO BEGIN E(I)=MB(A(I))$F(I)=MB(B(I))$G(I)=MB(C(I))                    B7
        23           END$ GO TO R5$                                                                         E7
        24       LC..FOR I=(1,1,560) DO BEGIN E(I)=MC(A(I))$F(I)=MC(B(I))$G(I)=MC(C(I))                    B8
        25           END$ GO TO R5$                                                                         E8
        26       LD..FOR I=(1,1,560) DO BEGIN E(I)=MD(A(I))$F(I)=MD(B(I))$G(I)=MD(C(I))                    B9
        27           END$ GO TO R5$                                                                         E9
        28       LE..FOR I=(1,1,560) DO BEGIN E(I)=ME(A(I))$F(I)=ME(B(I))$G(I)=ME(C(I))                    B10
        29           END$ GO TO R5$                                                                         E10
        30       LF..FOR I=(1,1,560) DO BEGIN E(I)=MF(A(I))$F(I)=MF(B(I))$G(I)=MF(C(I))                    B11
        31           END$ GO TO R5$                                                                         E11
        32       LG.. GO TO VERY$
        33       R5..FOR I=(1,1,560) DO BEGIN
        34          IF E(I) GTR F(I) THEN BEGIN R=E(I)$ E(I)=F(I)$ F(I)=R END$                             B12   B13   E13
        35          IF F(I) GTR G(I) THEN BEGIN R=F(I)$ F(I)=G(I)$ G(I)=R END$                             B14   E14
        36          IF E(I) GTR F(I) THEN BEGIN R=E(I)$ E(I)=F(I)$ F(I)=R END END$                         B15   E15   E12
        37          GO TO R0$
        38       R0..I=0$
        39       R1..I=I+1$ IF I EQL 561 THEN GO TO R6$ J=0$
        40          IF E(I) EQL 0 THEN GO TO R1$
        41       R2..J=J+1$ IF J EQL 561 THEN GO TO R1$
        42          IF E(I) EQL A(J) AND F(I) EQL B(J) AND G(I) EQL C(J) THEN BEGIN
        43          A(J)=0$ B(J)=0$ C(J)=0$ E(J)=0$ F(J)=0$ G(J)=0$ GO TO R1 END$                           B16   E16
        44          GO TO R2$
        45       R6.. Z=Z+1$ GO TO L(Z)$
```

```
          46      VERY..
          47                WRITE('LISTING OF ALL 3 IN A SQUARE POSSIBILITIES')$
          48         I=0$ Q=0$
          49      R8..I=I+1$ IF I EQL 561  THEN GO TO VERY2$
          50         IF A(I) EQL 0 THEN GO TO R8$ Q=Q+1$
          51         WRITE(Q,A(I),B(I),C(I))$
          52         GO TO R8$
          53      VERY2..FINISH                                                  EN
  END BLOCK 1
COMPILATION COMPLETED
```

```
FIRST TRANSFORMATION
          0          4          8         12         16          3          7         11         15          2
          6         10         14          1          5          9         13
SECOND TRANSFORMATION
          0         16         15         14         13         12         11         10          9          8
          7          6          5          4          3          2          1
THIRD TRANSFORMATION
          0         13          9          5          1         14         10          6          2         15
         11          7          3         16         12          8          4
FOURTH TRANSFORMATION
          0          1          5          9         13          2          6         10         14          3
          7         11         15          4          8         12         16
FIFTH TRANSFORMATION
          0          4          3          2          1          8          7          6          5         12
         11         10          9         16         15         14         13
SIXTH TRANSFORMATION
          0         16         12          8          4         15         11          7          3         14
         10          6          2         13          9          5          1
SEVENTH TRANSFORMATION
          0         13         14         15         16          9         10         11         12          5
          6          7          8          1          2          3          4
```

```
MONITOR SYSTEM -- 65K84 03/11/65
   RUN     R CITRENBAUM                                                    TIME:     10:45:14   DATE:   15 MAR 65


 ALG PROB2
    ALGOL
        1       MARCH 10,1965            INTERFACE   FEBRUARY 15,1965      PASS2   DECEMBER 23, 1964
        2       COMMENT THIS PROGRAM IS DESIGNED TO FIND ALL THE POSSIBLE WAYS
        3         AFTER SYMMETRY CONSIDERATIONS THAT FOUR  OCCUPIED CELLS CAN BE
        4         PLACED IN A PLANE OF SIXTEEN CELLS.$
        4       INTEGER I,J,K,Q,Z,R$
BLOCK 1     LEVEL 1                                                                                    B0
        5       INTEGER ARRAY A,B,C,D,E,F,G,H(1..1820),MZ,MA,MB,MC,MD,ME,MF(0..16)$
        6       SWITCH L=LA,LB,LC,LD,LE,LF,LG$
        7       READ (MZ,MA,MB,MC,MD,ME,MF)$
        8              WRITE('FIRST TRANSFORMATION',MZ)$
        9             WRITE('SECOND TRANSFORMATION',MA)$
       10              WRITE('THIRD TRANSFORMATION',MB)$
       11              WRITE('FOURTH TRANSFORMATION',MC)$
       12              WRITE('FIFTH TRANSFORMATION',MD)$
       13              WRITE('SIXTH TRANSFORMATION',ME)$
       14              WRITE('SEVENTH TRANSFORMATION',MF)$
       15      FOR I=(1,1,13) DO BEGIN FOR J=(2,1,14) DO BEGIN FOR K=(3,1,15) DO BEGIN            B1    B2
       16        FOR R=(4,1,16) DO BEGIN                                                          B3
       17      IF I LSS J AND J LSS K AND K LSS R                                                 B4
       18                             THEN BEGIN Q=Q+1$ A(Q)=I$ B(Q)=J$ C(Q)=K$D(Q)=R             B5
       19        END END END END END$                                                             E5    E4    E3    E2
                                                                                                  E1
       20          FOR I=(1,1,Q)   DO BEGIN E(I)=MZ(A(I))$F(I)=MZ(B(I))$G(I)=MZ(C(I))           B6
       21        $ H(I)=MZ(D(I))
       22          END$ GO TO R5$                                                               E6
       23      LA..FOR I=(1,1,Q)   DO BEGIN E(I)=MA(A(I))$F(I)=MA(B(I))$G(I)=MA(C(I))           B7
       24        $ H(I)=MA(D(I))
       25          END$ GO TO R5$                                                               E7
       26      LB..FOR I=(1,1,Q)   DO BEGIN E(I)=MB(A(I))$F(I)=MB(B(I))$G(I)=MB(C(I))           B8
       27        $ H(I)=MB(D(I))
       28          END$ GO TO R5$                                                               E8
       29      LC..FOR I=(1,1,Q)   DO BEGIN E(I)=MC(A(I))$F(I)=MC(B(I))$G(I)=MC(C(I))           B9
       30        $ H(I)=MC(D(I))
       31          END$ GO TO R5$                                                               E9
       32      LD..FOR I=(1,1,Q)   DO BEGIN E(I)=MD(A(I))$F(I)=MD(B(I))$G(I)=MD(C(I))           B10
       33        $ H(I)=MD(D(I))
       34          END$ GO TO R5$                                                               E10
       35      LE..FOR I=(1,1,Q)   DO BEGIN E(I)=ME(A(I))$F(I)=ME(B(I))$G(I)=ME(C(I))           B11
       36        $ H(I)=ME(D(I))
       37          END$ GO TO R5$                                                               E11
       38      LF..FOR I=(1,1,Q)   DO BEGIN E(I)=MF(A(I))$F(I)=MF(B(I))$G(I)=MF(C(I))           B12
       39        $ H(I)=MF(D(I))
       40          END$ GO TO R5$                                                               E12
       41      LG.. GO TO VERY$
       42      R5..FOR I=(1,1,Q)   DO BEGIN
       43          IF E(I) GTR F(I) THEN BEGIN R=E(I)$ E(I)=F(I)$ F(I)=R END$                   B13   B14   E14
       44          IF F(I) GTR G(I) THEN BEGIN R=F(I)$ F(I)=G(I)$ G(I)=R END$                   B15   E15
```

```
          45          IF G(I) GTR H(I) THEN BEGIN R=G(I)$ G(I)=H(I)$ H(I)=R END$                    B16   E16
          46          IF E(I) GTR F(I) THEN BEGIN R=E(I)$ E(I)=F(I)$ F(I)=R END$                    B17   E17
          47          IF F(I) GTR G(I) THEN BEGIN R=F(I)$ F(I)=G(I)$ G(I)=R END$                    B18   E18
          48          IF G(I) GTR H(I) THEN BEGIN R=G(I)$ G(I)=H(I)$ H(I)=R END$                    B19   E19
          49          IF E(I) GTR F(I) THEN BEGIN R=E(I)$ E(I)=F(I)$ F(I)=R END END$                B20   E20   E13
          50          GO TO R0$
          51       R0..I=0$
          52       R1..I=I+1$ IF I EQL 1821 THEN GO TO R6$ J=0$
          53          IF E(I) EQL 0 THEN GO TO R1$
          54       R2..J=J+1$ IF J EQL 1821 THEN GO TO R1$
          55          IF E(I) EQL A(J) AND F(I) EQL B(J) AND G(I) EQL C(J)
          56       AND H(I) EQL D(J) THEN BEGIN
          57            D(J)=0$ H(J)=0$                                                             B21
          58          A(J)=0$ B(J)=0$ C(J)=0$ E(J)=0$ F(J)=0$ G(J)=0$ GO TO R1 END$                 E21
          59          GO TO R2$
          60       R6.. Z=Z+1$ GO TO L(Z)$
          61       VERY..
          62                WRITE('LISTING OF ALL 4 IN A SQUARE POSSIBILITIES')$
          63          I=0$ Q=0$
          64       R8..I=I+1$ IF I EQL 1821 THEN GO TO VERY2$
          65          IF A(I) EQL 0 THEN GO TO R8$ Q=Q+1$
          66          WRITE(Q,A(I),B(I),C(I),D(I))$
          67          GO TO R8$
          68       VERY2..FINISH                                                                  F0
  END BLOCK 1
COMPILATION COMPLETED
```

```
FIRST TRANSFORMATION
          0          4          8         12         16          3          7         11         15          2
          6         10         14          1          5          9         13
SECOND TRANSFORMATION
          0         16         15         14         13         12         11         10          9          8
          7          6          5          4          3          2          1
THIRD TRANSFORMATION
          0         13          9          5          1         14         10          6          2         15
         11          7          3         16         12          8          4
FOURTH TRANSFORMATION
          0          1          5          9         13          2          6         10         14          3
          7         11         15          4          8         12         16
FIFTH TRANSFORMATION
          0          4          3          2          1          8          7          6          5         12
         11         10          9         16         15         14         13
SIXTH TRANSFORMATION
          0         16         12          8          4         15         11          7          3         14
         10          6          2         13          9          5          1
SEVENTH TRANSFORMATION
          0         13         14         15         16          9         10         11         12          5
          6          7          8          1          2          3          4
```

## APPENDIX III: INTERESTING COMPUTER VERSUS COMPUTER PLAYS

This appendix contains three complete computer versus computer plays obtained by the 1107 program. The plays chosen include:

1. A tie-game.
2. A win by the first player (X) resulting from the simultaneous formation of two independent related pairs.
3. A win by the second player (O).

These three plays are shown in Figures III.1 a,b,c respectively. (The views shown in Figure III.1 correspond to the four level model of the cube shown in Figure 3.2.) The reason behind each computer move in these plays (as printed out by the computer itself) is listed below:

Game One (Figure III.1a)

| | | | |
|---|---|---|---|
| $X_1$: | Preset move | $O_1$: | Preset move |
| $X_2$: | Preset move | $O_2$: | Preset move |
| $X_3$: | Preset move | $O_3$: | Preset move |
| $X_4$: | Forms an excess-three plane | $O_4$: | Blocks a TFS |
| $X_5$: | Forms an excess-three plane | $O_5$: | Forms an excess-three plane |

| | | | |
|---|---|---|---|
| $X_6$ : | Blocks a TFS | $O_6$ : | Forms an excess-three plane |
| $X_7$ : | Forms an excess-three plane | $O_7$ : | Blocks a related pair |
| $X_8$ : | Forms an excess-three plane | $O_8$ : | Blocks a related pair |
| $X_9$ : | Blocks a TFS | $O_9$ : | Blocks a 3-O file |
| $X_{10}$: | Blocks a related pair | $O_{10}$: | Forms an excess-three plane |
| $X_{11}$: | Blocks a TFS | $O_{11}$: | Blocks a TFS |
| $X_{12}$: | Preset move | $O_{12}$: | Preset move |
| $X_{13}$: | Blocks a O-3 file | $O_{13}$: | Moves to OWATCH cell in excess-two plane |
| $X_{14}$: | Blocks a O-3 file | $O_{14}$: | Forms an excess-four plane |
| $X_{15}$: | Blocks a related pair | $O_{15}$: | Forms an excess-three plane |
| $X_{16}$: | Blocks a O-3 file | $O_{16}$: | Blocks a 3-O file |
| $X_{17}$: | Blocks a O-3 file | $O_{17}$: | Blocks a related pair |
| $X_{18}$: | Blocks a O-3 file | $O_{18}$: | Blocks a related pair |
| $X_{19}$: | Moves to XWATCH cells in excess-two plane | $O_{19}$: | Blocks a 3-O file |
| $X_{20}$: | Forms an excess-four plane | $O_{20}$: | Blocks a related pair |
| $X_{21}$: | Blocks a related pair | $O_{21}$: | Preset move |

| | | | |
|---|---|---|---|
| $X_{22}$: | Moves to OWATCH cell in excess-three plane | $O_{22}$: | Blocks a 0-3 file |
| $X_{23}$: | Moves to OWATCH cell in excess-three plane | $O_{23}$: | Blocks a 0-3 file |
| $X_{24}$: | Forms an excess-four plane | $O_{24}$: | Blocks a 0-3 file |
| $X_{25}$: | Forms an excess-four plane | $O_{25}$: | Preset move |
| $X_{26}$: | Preset move | $O_{26}$: | Preset move |
| $X_{27}$: | Blocks related pair | $O_{27}$: | Preset move |
| $X_{28}$: | Preset move | $O_{28}$: | Preset move |
| $X_{29}$: | Preset move | $O_{29}$: | Preset move |
| $X_{30}$: | Preset move | $O_{30}$: | Preset move |
| $X_{31}$: | Blocks 0-3 file | $O_{31}$: | Blocks 3-0 file |
| $X_{32}$: | Preset move | $O_{32}$: | Preset move |

All board positions being occupied, the game is declared a tie.

Game Two (Figure III.1b)

| | | | |
|---|---|---|---|
| $X_1$ : | Preset move | $O_1$ : | Preset move |
| $X_2$ : | Preset move | $O_2$ : | Preset move |
| $X_3$ : | Preset move | $O_3$ : | Preset move |
| $X_4$ : | Forms an excess-three plane | $O_4$ : | Blocks a TFS |

| | | | |
|---|---|---|---|
| $X_5$ : | Moves to XWATCH cell in excess-two plane | $O_5$ : | Blocks a 3-O file |
| $X_6$ : | Blocks a O-3 file | $O_6$ : | Blocks a TFS |
| $X_7$ : | Forms an excess-three plane | $O_7$ : | Blocks a TFS |
| $X_8$ : | Blocks a TFS | $O_8$ : | Forms an excess-three plane |
| $X_9$ : | Blocks a TFS | $O_9$ : | Blocks a TFS |
| $X_{10}$: | Blocks a TFS | $O_{10}$: | Moves to OWATCH cell in excess-two plane |
| $X_{11}$: | Blocks O-3 file | $O_{11}$: | Blocks related pair |
| $X_{12}$: | Moves to XWATCH cell in excess-two plane | $O_{12}$: | Blocks 3-O file |
| $X_{13}$: | Blocks a TFS | $O_{13}$: | Forms excess-three plane |
| $X_{14}$: | Blocks a related pair | $O_{14}$: | Moves to OWATCH cell in excess-two plane |
| $X_{15}$: | Blocks a O-3 file | $O_{15}$: | Forms an excess-four plane |
| $X_{16}$: | Blocks a related pair | $O_{16}$: | Blocks a 3-O file |
| $X_{17}$: | Blocks a O-3 file | $O_{17}$: | Blocks a related pair |
| $X_{18}$: | Forms two 3-O files | $O_{18}$: | Blocks a 3-O file, concedes |
| $X_{19}$: | Forms a 4-O file, wins | | |

Game Three (Figure III.1c)

| | | | |
|---|---|---|---|
| $X_1$ : | Preset move | $O_1$ : | Preset move |
| $X_2$ : | Preset move | $O_2$ : | Preset move |
| $X_3$ : | Forms an excess-three plane | $O_3$ : | Blocks a TFS |
| $X_4$ : | Forms an excess-three plane | $O_4$ : | Blocks a TFS |
| $X_5$ : | Forms an excess-three plane | $O_5$ : | Blocks a TFS |
| $X_6$ : | Forms an excess-three plane | $O_6$ : | Blocks a TFS |
| $X_7$ : | Starts a TFS | $O_7$ : | Blocks a 3-0 file |
| $X_8$ : | Continues TFS | $O_8$ : | Blocks a 3-0 file |
| $X_9$ : | Continues TFS | $O_9$ : | Blocks a 3-0 file |
| $X_{10}$: | Forms two 3-0 files | $O_{10}$: | Blocks a 3-0 file, concedes |
| $X_{11}$: | Forms a 4-0 file, wins | | |

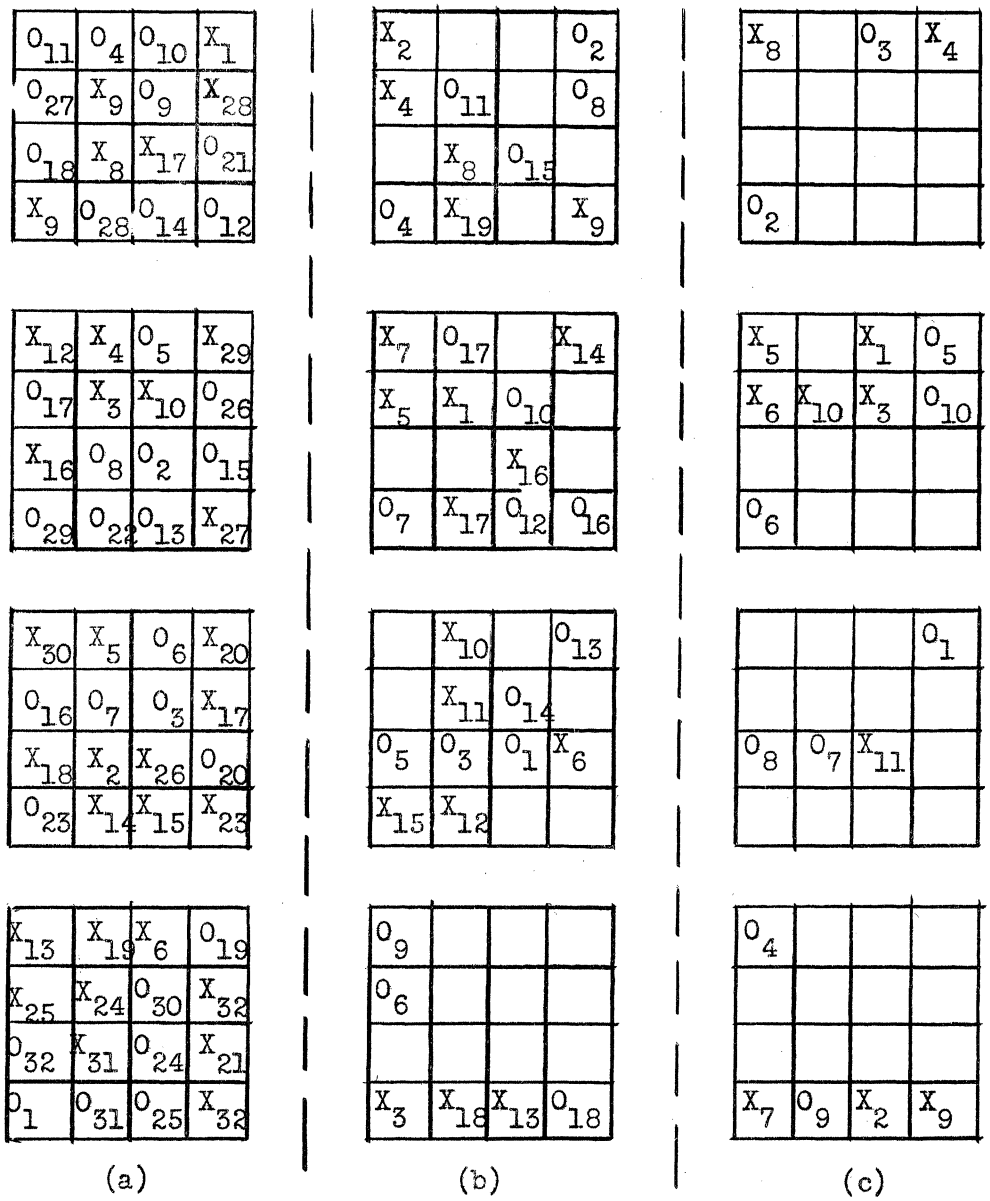



Figure III.1

APPENDIX IV: PDP-4 PROGRAM

By virtue of a program similar to the one given in this appendix, the PDP-4 computer plays against human opponents. The game is played on the 340 display unit which works in conjunction with the PDP-4. Desired figures are programmed to appear on the 340 D using the $2^{20}$ available points of light on the CRT, a square of side 9 1/4 inches. These figures include:

1. Four level two dimensional view of cube (four 4x4 planes)
2. Message giving reason for preceding computer move
3. Isometric view of the cube
4. Clockface timing each opponent move (novelty)

The opponent's moves are read into the computer by either of two means:

1. Pressing the appropriate buttons on the console
2. Pointing light pen (photocell) at desired cell on four level view

When an opponent's move is read into the computer, a decision is made as to the proper countermove and both the move and countermove are displayed on the

four level view and isometric view. At the same time, the message on the screen changes from its former state to describe the reason for the computer's move. The clock resets itself to zero and begins ticking away in real time waiting for the opponent's next move. (The clock has hour, minute, and second hands which update themselves once each second by means of an internal counter in the computer. The counter is accurate to 1/60 second.)

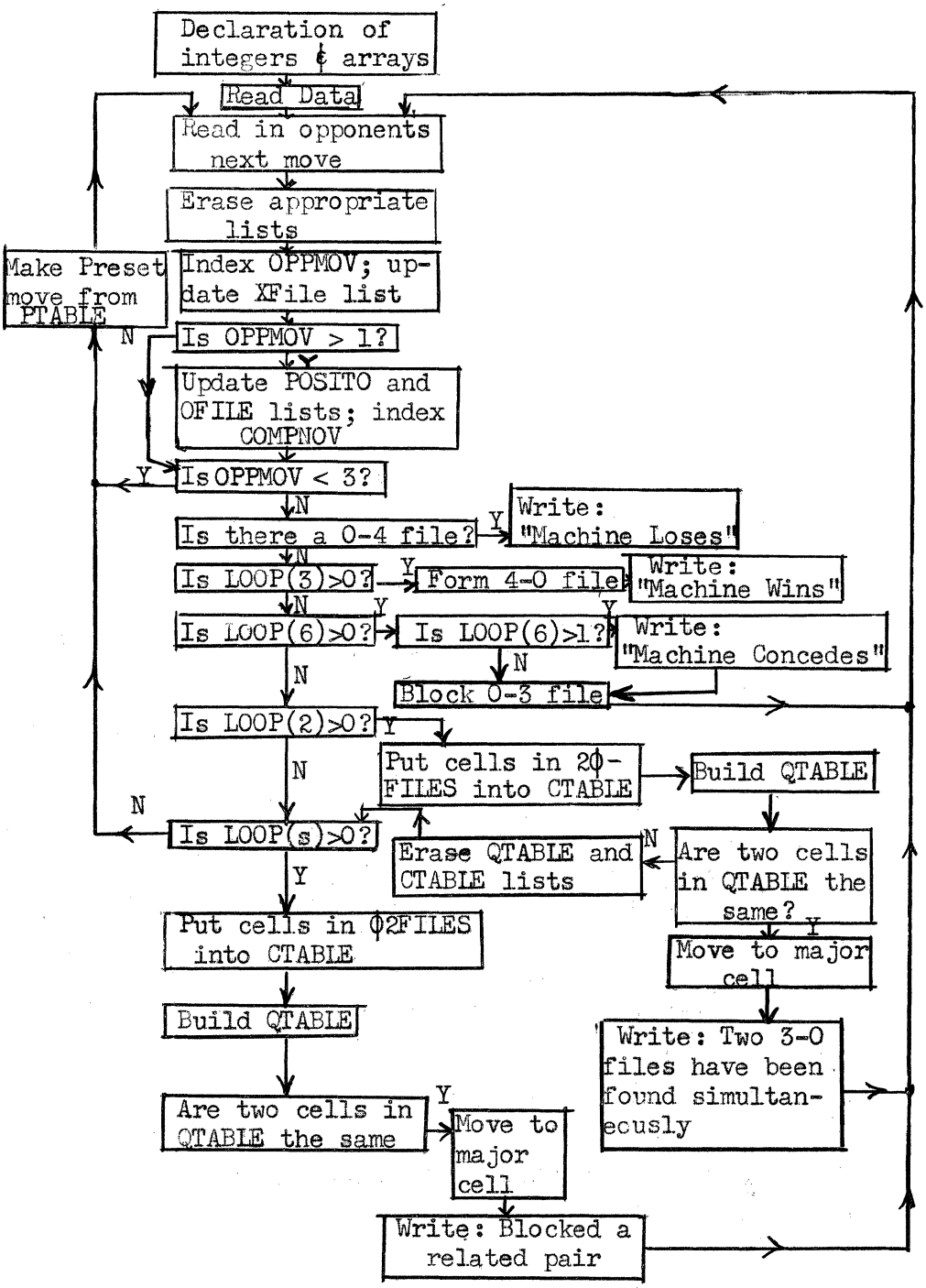
When a win by either the computer or the opponent occurs, a line is drawn through the appropriate four cells in the isometric view.

Due to the length of the original program written in MIDAS 4 Assembly Language for the PDP-4 an ALGOL 60 program using the same related pair strategy is substituted below. The novel characteristics of the MIDAS 4 program are not included in the ALGOL version.

As an aid to understanding the program, a flow chart is given in Figure IV.1, and definitions of the integers and arrays used in the program are given in Appendix V. The data for the three arrays used (FILENOS, FILECELLS, and PTABLE) is the same as that for the program in Appendix V and is included after

that program.

Figure IV.1: Flow Chart of ALGOL Program for Related Pair Strategy



```
MONITOR SYSTEM -- 65K84 03/11/65
   RUN     R CITRENBAUM                                                    TIME:     10:44:39   DATE:   15 MAR 65


 ALG PROB1
    ALGOL       MARCH 10,1965          INTERFACE   FEBRUARY 15,1965      PASS2   DECEMBER 23, 1964
         1      COMMENT THIS PROGRAM IS DESIGNED TO PLAY THREE DIMENSIONAL TIC-TAC-TOE
         2          USING THE RELATED PAIR STRATEGY.
         3             ALL MOVES MADE BY THE OPPONENT MUST BE SPECIFIED BEFORE THE
         4             GAME BEGINS.$
         5      INTEGER DUM,I,OPPMOV,NUMX,COMPMOV,NUMO,DELTA,J,K,M$
BLOCK 1     LEVEL 1                                                                          B0
         6      INTEGER ARRAY FILENOS(1..448),
         7                    FILECELLS(1..304), PTABLE(1..64), NEWMOVE(1..20),
         8                    POSITO,POSITX(1..32), CTABLE(1..60), QTABLE(1..30),
         9                    OFILE, XFILE(0..76), TAB10,TAB01(1..50), TAB20,
        10                    TAB02(1..15), TAB30, TAB03(1..4), LOOP(1..6)$
        11      SWITCH LA=LA0, LA1, LA2, LA3$
        12      SWITCH LB=LB0,LB1,LB2,LB3,LB4$
        13      READ ( FILENOS, FILECELLS, PTABLE, NEWMOVE)$
        14      WRITE('COMPLETE LISTING OF DATA- INCLUDING INFORMATION IN THE LISTS.')$
        15      WRITE ('FILENOS LIST',FILENOS)$
        16      WRITE ('FILECELLS LIST',FILECELLS)$
        17      WRITE ('PTABLE LIST',PTABLE)$
        18      NEXTMOVE..FOR I=(1,1,60) DO CTABLE(I)=0$
        19                FOR I=(1,1,30) DO QTABLE(I)=0$
        20                FOR I=(1,1,50) DO BEGIN TAB10(I)=0$ TAB01(I)=0 END$                B1    E1
        21                FOR I=(1,1,15) DO BEGIN TAB20(I)=0$ TAB02(I)=0 END$                B2    E2
        22                FOR I=(1,1,4 ) DO BEGIN TAB30(I)=0$ TAB03(I)=0 END$                B3    E3
        23                OPPMOV=OPPMOV+1$
        24                NUMX=NEWMOVE(OPPMOV)$
        25                WRITE ('THE OPPONENT HAS MOVED TO CELL',NUMX)$
        26                POSITX(OPPMOV)=NUMX$
        27                DELTA=(7*NUMX)-6$
        28                FOR K=(1,1,7) DO
        29                       BEGIN M=FILENOS(DELTA)$                                     B4
        30                            XFILE(M)=XFILE(M)+1$
        31                           DELTA=DELTA+1 END$                                     E4
        32               IF OPPMOV GTR 1 THEN
        33                      BEGIN COMPMOV=OPPMOV-1$                                     B5
        34                         POSITO(COMPMOV)=NUMO$
        35                DELTA=(7*NUMO)-6$
        36                FOR K=(1,1,7) DO
        37                       BEGIN M=FILENOS(DELTA)$                                     B6
        38                            OFILE(M)=OFILE(M)+1$
        39                           DELTA=DELTA+1 END END$                                 E6    E5
        40                IF OPPMOV LSS 3 THEN GO TO PRESET$
        41                FOR I=(1,1,6) DO LOOP(I)=0$
        42                FOR I=(1,1,76) DO
        43                    BEGIN IF XFILE(I) EQL 0 THEN                                  B7
        44                      BEGIN DUM=OFILE(I)+1$                                       B8
        45                                GO TO LA(DUM) END ELSE                            E8
```

```
46                    IF OFILE(I) EQL 0 THEN
47                     BEGIN DUM=xFILE(I)+1$                                    B9
48                                GO TO LB(DUM) END$                           E9
49      LA0..GO TO ALL$
50      LA1..LOOP(1)=LOOP(1)+1$
51            K=LOOP(1)$
52            TAB10(K)=I$
53           GO TO ALL$
54      LA2..LOOP(2)=LOOP(2)+1$
55            K=LOOP(2)$
56            TAB20(K)=I$
57           GO TO ALL$
58      LA3..LOOP(3)=LOOP(3)+1$
59            K=LOOP(3)$
60            TAB30(K)=I$
61           GO TO ALL$
62      LB1..LOOP(4)=LOOP(4)+1$
63            K=LOOP(4)$
64            TAB01(K)=I$
65           GO TO ALL$
66      LB2..LOOP(5)=LOOP(5)+1$
67            K=LOOP(5)$
68            TAB02(K)=I$
69           GO TO ALL$
70      LB3..LOOP(6)=LOOP(6)+1$
71            K=LOOP(6)$
72            TAB03(K)=I$
73           GO TO ALL$
74      LB4..WRITE('THE COMPUTER HAS LOST.')$
75            WRITE('THE OPPONENT HAS FOUR IN A ROW IN FILE',I)$
76           GO TO VERY$
77      ALL..M=1 END FORLOOP$                                                  E7    C
78            IF LOOP(3) GTR 0 THEN GO TO MAKE40$
79            IF LOOP(6) GTR 0 THEN GO TO BLOCK03$
80            IF LOOP(2) GTR 1 THEN GO TO TEST20$
81            IF LOOP(5) GTR 1 THEN GO TO TEST02 ELSE GO TO PRESET$
82      MAKE40..DELTA=(4*TAB30(1))-3$
83             FOR I=(1,1,4) DO BEGIN CTABLE(I)=FILECELLS(DELTA)$              B10
84                  DELTA=DELTA+1 END FORLOOP$                                 E10   C
85              J=0$
86      BACK2..J=J+1$ I=0$
87      BACK1..I=I+1$
88            IF POSITO(I) EQL 0 THEN GO TO WIN ELSE
89            IF POSITO(I) EQL CTABLE(J) THEN GO TO BACK2
90                  ELSE GO TO BACK1$
91      WIN..WRITE('THE COMPUTER HAS WON AS IT HAS MADE FOUR')$
92            WRITE('IN A ROW BY MOVING TO CELL',CTABLE(J))$
93            GO TO VERY$
94      BLOCK03..IF LOOP(6) GTR 1 THEN
95           BEGIN WRITE('THE COMPUTER CONCEDES THE GAME.  IT HOPES THE')$      B11
96                      WRITE('OPPONENT WILL GIVE IT AN OPPORTUNITY TO PLAY')$
97                 WRITE('  AGAIN')$
98                           GO TO BLOCKA END$                                 E11
99      BLOCKA..DELTA=(4*TAB03(1))-3$
```

```
100             FOR I=(1,1,4) DO BEGIN CTABLE(I)=FILECELLS(DELTA)$                B12
101                 DELTA=DELTA+1 END$                                            E12
102           J=0$
103     BACK4..J=J+1$ I=0$
104     BACK3..I=I+1$
105           IF POSITX(I) EQL 0 THEN GO TO BLOCKR ELSE
106           IF POSITX(I) EQL CTABLE(J) THEN GO TO BACK4 ELSE GO TO BACK3$
107     BLOCKR..NUMO=CTABLE(J)$
108          WRITE('THE COMPUTER HAS BLOCKED A THREE IN A ROW ')$
109               WRITE('       BY MOVING TO CELL',NUMO)$
110            GO TO NEXTMOVE$
111     TEST20..I=0$K=0$
112     BACK5..I=I+1$
113           IF I LEQ LOOP(2) THEN
114                BEGIN DELTA=(4*TAB20(I))-3$ FOR J=(1,1,4) DO                   B13
115                     BEGIN M=J+K$ CTABLE(M)=FILECELLS(DELTA)$                  B14
116                          DELTA=DELTA+1 END FORLOOP$                           E14   C
117                     K=K+4$
118                      GO TO BACK5 END                                         E13
119                ELSE GO TO BIRD$
120     BIRD..J=0$ K=1$
121     BACK6..J=J+1$ I=0$
122           IF CTABLE(J) EQL 0 THEN GO TO COLT$
123     BACK7..I=I+1$
124           IF POSITO(I) EQL 0 THEN
125                     BEGIN QTABLE(K)=CTABLE(J)$ K=K+1$                         B15
126                       GO TO BACK6 END$                                        E15
127           IF POSITO(I) EQL CTABLE(J) THEN GO TO BACK6 ELSE GO TO BACK7$
128     COLT..J=1$ K=2$
129     BACK11..IF QTABLE(J) EQL QTABLE(K) THEN GO TO RELATED ELSE K=K+1$
130           IF QTABLE(K) GTR 0 THEN GO TO BACK11 ELSE J=J+1$ K=J+1$
131           IF QTABLE(K) GTR 0 THEN GO TO BACK11 ELSE GO TO REMAKE$
132     REMAKE..FOR I=(1,1,60) DO CTABLE(I)=0$
133            FOR I=(1,1,30) DO QTABLE(I)=0$
134         IF LOOP(5) EQL 0 THEN GO TO PRESET$
135     TEST02..I=0$ K=0$
136     BACK8..I=I+1$
137           IF I LEQ LOOP(5) THEN
138               BEGIN DELTA=(4*TAB02(I))-3$ FOR J=(1,1,4) DO                   B16
139                    BEGIN M=J+K$                                              B17
140                         CTABLE(M)=FILECELLS(DELTA)$
141                      DELTA=DELTA+1 END$                                       E17
142                  K=K+4$ GO TO BACK8 END$                                      E16
143     ORIOLE..J=0$ K=1$
144     BACK9..J=J+1$ I=0$
145            IF CTABLE(J) EQL 0 THEN GO TO  BULLET$
146     BACK10..I=I+1$
147           IF POSITX(I) EQL 0 THEN BEGIN QTABLE(K)=CTABLE(J)$                 B18
148               K=K+1$ GO TO BACK9 END$                                        E18
149           IF POSITX(I) EQL CTABLE(J) THEN GO TO BACK9 ELSE GO TO BACK10$
150     BULLET..J=1$ K=2$
151     BACK12..IF QTABLE(J) EQL QTABLE(K) THEN GO TO BLOCKREL ELSE K=K+1$
152           IF QTABLE(K) GTR 0 THEN GO TO BACK12 ELSE J=J+1$ K=J+1$
153           IF QTABLE(K) GTR 0  THEN GO TO BACK12 ELSE GO TO PRESET$
```

```
154       RELATED..NUMO=QTABLE(J)$
155                WRITE('THE COMPUTER HAS FORMED A RELATED')$
156                 WRITE('     PAIR AND WILL WIN ON ITS NEXT MOVE.')$
157                 WRITE('     IT HAS JUST MOVED TO CELL',NUMO)$
158              GO TO NEXTMOVE$
159       BLOCKREL..NUMO=QTABLE(J)$
160                WRITE('THE COMPUTER HAS BLOCKED A POSSIBLE')$
161                WRITE    ('      RELATED PAIR BY MOVING TO CELL',NUMO)$
162              GO TO NEXTMOVE$
163       PRESET..I=1$ J=1$
164       BACK14..IF PTABLE(J) EQL POSITX(I) OR PTABLE(J) EQL POSITO(I) THEN
165                     BEGIN J=J+1$ I=1$ GO TO BACK14 END$  I=I+1$                    B19   E19
166              IF I GTR OPPMOV THEN BEGIN NUMO=PTABLE(J)$                            B20
167                     WRITE('THE COMPUTER HAS NOT BEEN FORCED TO GO')$
168                     WRITE('    ON THE DEFENSIVE AND HAS MOVED TO CELL',NUMO)$
169                          GO TO NEXTMOVE END$  GO TO BACK14$                       E20
170       VERY.. WRITE('POSITO TABLE')$
171       FOR I=(1,1,COMPMOV) DO WRITE (POSITO(I))$
172       FOR I=(1,1,OPPMOV ) DO WRITE (POSITX(I))$
173       FINISH                                                                    F0
  END BLOCK 1
COMPILATION COMPLETED
```

## APPENDIX V: 1107 PROGRAM

The computer versus computer games have been played using the Univac 1107 computer programmed in ALGOL 60.

To aid the interested reader in understanding the program, a list of the purposes of the integers and arrays contained in the program are given below. The number in parenthesis for each of the arrays is the number of available elements in that array. The elements of the first eight arrays discussed below are given in the data section shown after the program.

### Integer Listing

| | |
|---|---|
| OPPMOV | Move number of the preceding move of the opponent |
| COMPMOV | Move number of the previous move of the machine |
| NUMX | Cell to which opponent made previous move (numbered from 1 to 64) |
| NUMO | Cell to which machine made previous move (numbered from 1 to 64) |
| GAME | Play under consideration. Allows many plays to be done in succession |
| SAVETR | Cell in TREE which is saved for later examination |
| CNTFIL | Number of 2-0 files in plane under consideration |

| | |
|---|---|
| DEF | Switch which determines whether computer is observing offensive or defensive tactics |
| FT,CK | Indices used for counting types of excess-two and excess-three planar situations on offense and defense |
| PLAYER | Switch determining whether the computer is playing as X or 0 |
| TRY 1, TRY 2 | Switches indicating status of search for excess-two and excess-three planar situations on both offense and defense |
| RELPR | Switch indicating if a related pair is available to the opponent |
| FILE1Ø | Number of 1-0 files in the cube under consideration |
| STABP | Index counting number of planes of the same type |
| CNTSQ | Number corresponding to the move look-ahead number of the plane under consideration |
| FORCESEQ | Switches indicating a TFS has been already found |
| COUNT2Ø | Number of cells in CELL2Ø at any particular time |
| CNT,CNT2 | Indices indicating reason a lookahead for a planar TFS has failed |
| A,B,C,D,E,I,J,K,M,N,Q,R,X,Y,Z, DELTA, DUM | Dummy variables |

## Array Listing

| | |
|---|---|
| FILENOS (448) | List of all files in which each of the 64 cells is a member |
| FILECELLS (304) | List of the four cells in each of the 76 files |
| CELLNOS (40) | List of the four cells in each of the 10 files in a given plane |
| SFILENOS (48) | List of all files in a given plane in which each of the 16 cells of the plane is a member |
| SQFILES (72) | List of four independent files in each of the 18 planes |
| PTABLE (64) | List of the 64 cells for use as a preset move list |
| MX,MY (17) | List of the offensive and defensive planar situations to be considered |
| OSQ,XSQ (18) | List of number of machine and opponent cells occupied in each of the 18 planes |
| WATCH,OWATCH (7 ) | List of cells in a plane which are XWATCH or OWATCH cells |
| OFILE,XFILE (76) | List of number of machine and opponent cells occupied in each of the 76 files |
| CELL2Ø (12) | List of all cells which are in 2-0 files of a plane under consideration |
| SPOSITO,SPOSITX (10) | List of occupied cells by machine and opponent respectively, in a lookahead sequence in a plane under consideration |

POSITO, POSITX (32)
List of occupied cells by machine and opponent in the overall play

FILE2Ø (5) List of the 2-0 files in a plane under consideration

DOFILE,DXFILE (10)
List of number of machine and opponent cells occupied in each of the 10 files of a plane under consideration

TREE (8,10) Array storing move possibilities for lookahead in TFS for a plane under study

CIRC (7,7) Array storing number of types of planar situations available at any given time

STAB (7,7,8) Array storing number of each type of planar situation available

TAB1Ø,TABØ1 (50) List of all 1-0 and 0-1 files in the cube

TAB2Ø,TABØ2 (15) List of all 2-0 and 0-2 files in the cube

TAB3Ø,TABØ3 (4) List of all 3-0 and 0-3 files in the cube

LOOP (6) Number of each type of "x-y" file in the cube

RCTABLE (16) List of the 16 cells in the plane under study

CTABLE (60); QTABLE,OQCELLS(30); DCTABLE(40); STABLE, TTABLE(7); CHK(10)
Lists used for storage purposes

The program followed by the data arrays are given in the next several pages.

```
ALG PROB1
   ALGOL        MARCH 10,1965          INTERFACE   FEBRUARY 15,1965      PASS2   DECEMBER 23, 1964
        1       COMMENT THIS PROGRAM IS DESIGNED TO ALLOW THE COMPUTER TO PLAY THREE
        2          DIMENSIONAL TIC-TAC-TOE AGAINST ITSELF USING PLANAR STRATEGY.
        3               ALL MOVES MADE BY THE OPPONENT MUST BE SPECIFIED BEFORE THE
        4               GAME BEGINS.$
        5       INTEGER OPPMOV, COMPMOV, NUMX,NUMO, DELTA, DUM, SAVETP, CNTFIL, GAME,
BLOCK 1     LEVEL 1                                                                    B0
        6                DEF, X,Y,Z,FT,CH,RELPR,TRY1,TRY2,FILE10,STARP, PLAYER,
        7                CNTSQ, FORCESEQ, COUNT20, CNT, CNT2, A,B,C,D,E,I,J,K,M,Q,R,N$
        8       INTEGER ARRAY FILENOS(1..448),
        9                     FILECELLS(1..304), PTABLE(1..64), NEWMOVE(1..20),
       10                  QSQ,XSQ(1..18), TREE(1..8,1..10),WATCH(1..7),
       11                  STABLE,TTABLE(1..8),MX,MY(1..17),CHK(1..10),
       12                  QCTABLE(1..40), CELL20,SPOSITX,SPOSITO(1..12),
       13                  FILE20(1..5), SFILENOS(1..48), QOFILE,QXFILE(0..10),
       14                  RCTABLE(1..16), STAB(0..7,0..7,1..8), CIRC(0..7,0..7),
       15                  SQFILES(1..72), CELLNOS(1..40), QQCELLS(1..30), QWATCH(1..7),
       16                     POSITO,POSITX(1..34), CTABLE(1..60), QTABLE(1..30),
       17                     OFILE, XFILE(0..76), TAB10,TAB01(1..50), TAB20,
       18                     TAB02(1..15), TAB30, TAB03(1..4), LOOP(1..6)$
       19       SWITCH L4=LA0, LA1, LA2, LA3$
       20       SWITCH LH=LA0,LB1,LB2,LB3,LB4$
       21       DUMP TREE, CNTSQ, CELL20$
       22       READ(SQFILES, SFILENOS, FILENOS,FILECELLS, CELLNOS,PTABLE,NEWMOVE,
       23            MX,MY)$
       24       WRITE('COMPLETE LISTING OF DATA- INCLUDING INFORMATION IN THE LISTS.')$
       25       WRITE ('SQFILES LIST',SQFILES)$
       26       WRITE ('SFILENOS LIST',SFILENOS)$
       27       WRITE ('FILENOS LIST',FILENOS)$
       28       WRITE ('FILECELLS LIST',FILECELLS)$
       29       WRITE ('CELLNOS LIST',CELLNOS)$
       30       WRITE ('PTABLE LIST',PTABLE)$
       31       WRITE('MX SWITCH LIST',MX)$
       32       WRITE('MY SWITCH LIST',MY)$
       33          GAME=0$
       34       NXGM..GAME=GAME+1$ WRITE('WE ARE NOW PLAYING GAME',GAME)$
       35          IF GAME EQL 16 THEN GO TO COMPL$
       36          FORCESEQ=0$ OPPMOV=0$ COMPMOV=0$  PLAYER=1$
       37          FOR I=(1,1,32) DO BEGIN POSITO(I)=0$ POSITX(I)=0 END$                  B1    E1
       38          FOR I=(0,1,76) DO BEGIN OFILE(I)=0$ XFILE(I)=0 END$                    B2    E2
       39       NEXTMOVE..FOR I=(1,1,60) DO CTABLE(I)=0$
       40          FOR I=(1,1,6) DO FOR J=(1,1,10) DO TREE(I,J)=0$
       41          FOR I=(1,1,10) DO CHK(I)=0$
       42                FOR I=(1,1,30) DO QTABLE(I)=0$
       43                FOR I=(1,1,30) DO QQCELLS(I)=0$
       44                FOR I=(1,1,50) DO BEGIN TAB10(I)=0$ TAB01(I)=0 END$              B3    E3
       45                FOR I=(1,1,15) DO BEGIN TAB20(I)=0$ TAB02(I)=0 END$              B4    E4
```

```
46              FOR I=(1,1,4 ) DO BEGIN TAB30(I)=0$ TAB03(1)=0 END$                    B5    E5
47          TRY1=0$ TRY2=0$ CH=0$ FT=0$ RELPR=0$
48              FOR I=(0,1,7) DO FOR J=(0,1,7) DO FOR K=(1,1,8) DO
49                      STAB(I,J,K)=0$
50               FOR I=(0,1,7) DO FOR J=(0,1,7) DO CIRC(I,J)=0$
51        OPPMOV=OPPMOV+1$ IF OPPMOV EQL 1 THEN GO TO PRESET$
52        FOR R=(2,2,64) DO IF OPPMOV EQL R THEN BEGIN PLAYER=0$                       B6
53        POSITO(OPPMOV/2)=NUM0$ FOR I=(1,1,OPPMOV/2) DO BEGIN J=POSITX(I)$            B7
54        POSITX(I)=POSITO(I)$ POSITO(I)=J$END$ GO TO INDEX0$ END$                     E7    E6
55      PLAYER=1$ FOR I=(1,1,(OPPMOV-1)/2) DO BEGIN POSITO((OPPMOV-1)/2)=NUM0$         B8
56        J=POSITX(I)$ POSITX(I)=POSITO(I)$POSITO(I)=J END$ GO TO INDEX0$              E8
57    INDEX0..FOR I=(1,1,76) DO BEGIN XFILE(I)=0$ OFILE(I)=0 END$   J=0$              B9    E9
58    INDEX1..J=J+1$ IF POSITX(J) GTR 0 THEN BEGIN D=7*POSITX(J)-6$                  B10
59        FOR K=(1,1,7) DO BEGIN M=FILENO$(D)$ XFILE(M)=XFILE(M)+1$ D=D+1 END$       B11   E11
60        GO TO INDEX1 END$   J=0$                                                   E10
61    INDEX2..J=J+1$ IF POSITO(J) GTR 0 THEN BEGIN D=7*POSITO(J)-6$                  B12
62        FOR K=(1,1,7) DO BEGIN M=FILENO$(D)$ OFILE(M)=OFILE(M)+1$ D=D+1 END$       B13   E13
63        GO TO INDEX2 END$                                                          E12
64        IF OPPMOV LSS 5 THEN GO TO PRESET$
65              FOR I=(1,1,6) DO LOOP(I)=0$
66              FOR I=(1,1,76) DO
67                  BEGIN IF XFILE(I) EQL 0 THEN                                   B14
68                    BEGIN DUM=OFILE(I)+1$                                        B15
69                            GO TO LA(DUM) END ELSE                               E15
70                   IF OFILE(I) EQL 0 THEN
71                    BEGIN DUM=XFILE(I)+1$                                        B16
72                            GO TO LB(DUM) END$                                   E16
73    LA0..GO TO ALL$
74    LA1..LOOP(1)=LOOP(1)+1$
75          K=LOOP(1)$
76          TAB10(K)=I$
77        GO TO ALL$
78    LA2..LOOP(2)=LOOP(2)+1$
79          K=LOOP(2)$
80          TAB20(K)=I$
81        GO TO ALL$
82    LA3..LOOP(3)=LOOP(3)+1$
83          K=LOOP(3)$
84          TAB30(K)=I$
85        GO TO ALL$
86    LB1..LOOP(4)=LOOP(4)+1$
87          K=LOOP(4)$
88          TAB01(K)=I$
89        GO TO ALL$
90    LB2..LOOP(5)=LOOP(5)+1$
91          K=LOOP(5)$
92          TAB02(K)=I$
93        GO TO ALL$
94    LB3..LOOP(6)=LOOP(6)+1$
95          K=LOOP(6)$
96          TAB03(K)=I$
```

```
97          GO TO ALL$
98      LB4..WRITE('THE COMPUTER HAS LOST.')$
99            WRITE('THE OPPONENT HAS FOUR IN A ROW IN FILE',I)$
100         GO TO VERY$
101     ALL..M=1 END FORLOOP$                                              E14   C
102           IF LOOP(3) GTR 0 THEN GO TO MAKE40$
103           IF LOOP(6) GTR 0 THEN GO TO BLOCK03$
104           IF LOOP(2) GTR 0 THEN GO TO TEST20$
105           IF LOOP(5) GTR 0 THEN GO TO TEST02 ELSE GO TO BUILDSQ$
106     MAKE4n..DELTA=(4*TAB30(I))-3$
107            FOR I=(1,1,4) DO BEGIN CTABLE(I)=FILECELLS(DELTA)$          B17
108                DELTA=DELTA+1 End FORLOOP$                              E17   C
109            J=0$
110     BACK2..J=J+1$ I=0$
111     BACK1..I=I+1$
112           IF POSITO(I) EQL 0 THEN GO TO WIN ELSE
113           IF POSITO(I) EQL CTABLE(J) THEN GO TO BACK2
114                ELSE GO TO BACK1$
115     WIN..WRITE('THE COMPUTER HAS WON AS IT HAS MADE FOUR')$
116           WRITE('IN A ROW BY MOVING TO CELL',CTABLE(J))$
117           GO TO VERY$
118     BLOCK03..IF LOOP(6) GTR 1 THEN
119           BEGIN WRITE('THE COMPUTER CONCEDES THE GAME.  IT HOPES THE')$  B18
120                       WRITE('OPPONENT WILL GIVE IT AN OPPORTUNITY TO PLAY')$
121                 WRITE('  AGAIN')$
122                       GO TO BLOCKA END$                                 E18
123     BLOCKA..DELTA=(4*TAB03(I))-3$
124            FOR I=(1,1,4) DO BEGIN CTABLE(I)=FILECELLS(DELTA)$          B19
125                 DELTA=DELTA+1 END$                                     E19
126           J=0$
127     BACK4..J=J+1$ I=0$
128     BACK3..I=I+1$
129           IF POSITX(I) EQL 0 THEN GO TO BLOCKB ELSE
130           IF POSITX(I) EQL CTABLE(J) THEN GO TO BACK4 ELSE GO TO BACK3$
131     BLOCKB..NUMO=CTABLE(J)$
132           WRITE('THE COMPUTER HAS BLOCKED A THREE IN A ROW ')$
133                WRITE('      BY MOVING TO CELL',NUMO)$
134            GO TO NEXTMOVE$
135     TEST20..I=0$K=0$
136     BACK5..I=I+1$
137           IF I LEQ LOOP(2) THEN
138                 BEGIN DELTA=(4*TAB20(I))-3$ FOR J=(1,1,4) DO           B20
139                       BEGIN M=J+K$ CTABLE(M)=FILECELLS(DELTA)$         B21
140                             DELTA=DELTA+1 END FORLOOP$                 E21   C
141                       K=K+4$
142                       GO TO BACK5 END                                  E20
143                ELSE GO TO BIRD$
144     BIRD..J=0$ K=1$
145     BACK6..J=J+1$ I=0$
146           IF CTABLE(J) EQL 0 THEN GO TO COLT$
147     BACK7..I=I+1$
```

```
148             IF POSITO(I) EQL 0 THEN
149                     BEGIN QTABLE(K)=CTABLE(J)$ K=K+1$                        B22
150                       GO TO BACK6 END$                                       E22
151             IF POSITO(I) EQL CTABLE(J) THEN GO TO BACK6 ELSE GO TO BACK7$
152     COLT..J=1$ K=2$
153     BACK11..IF QTABLE(J) EQL QTABLE(K) THEN GO TO RELATED ELSE K=K+1$
154             IF QTABLE(K) GTR 0 THEN GO TO BACK11 ELSE J=J+1$ K=J+1$
155             IF QTABLE(K) GTR 0 THEN GO TO BACK11 ELSE GO TO REMAKE$
156     REMAKE..FOR I=(1,1,4*LOOP(2)) DO CTABLE(I)=0$
157              FOR I=(1,1,(4*LOOP(2))/2) DO BEGIN OQCELLS(I)=QTABLE(I)$        B23
158                 QTABLE(I)=0 END$                                             E23
159               IF LOOP(5) EQL 0 THEN GO TO BUILDS0$
160     TESTU2..I=0$ K=0$
161     BACK8..I=I+1$
162             IF I LEQ LOOP(5) THEN
163                 BEGIN DELTA=(4*TABO2(I))-3$ FOR J=(1,1,4) DO                 B24
164                     BEGIN M=J+K$                                             B25
165                           CTABLE(M)=FILECELLS(DELTA)$
166                       DELTA=DELTA+1 END$                                     E25
167                   K=K+4$ GO TO BACK8 END$                                    E24
168     ORIOLE..J=0$ K=1$
169     BACK9..J=J+1$ I=0$
170             IF CTABLE(J) EQL 0 THEN GO TO  BULLET$
171     BACK10..I=I+1$
172             IF POSITX(I) EQL 0 THEN BEGIN QTABLE(K)=CTABLE(J)$               B26
173                  K=K+1$ GO TO BACK9 END$                                     E26
174             IF POSITX(I) EQL CTABLE(J) THEN GO TO BACK9 ELSE GO TO BACK10$
175     BULLET..J=1$ K=2$
176     BACK12..IF QTABLE(J) EQL QTABLE(K) THEN GO TO BACK50   ELSE K=K+1$
177             IF QTABLE(K) GTR 0 THEN GO TO BACK12 ELSE J=J+1$ K=J+1$
178           IF QTABLE(K) GTR 0 THEN GO TO BACK12 ELSE GO TO BUILDS0$
179     RELATED..NUMO=QTABLE(J)$
180              WRITE('THE COMPUTER HAS FORMED A RELATED')$
181               WRITE('      PAIR AND WILL WIN ON ITS NEXT MOVE.')$
182               WRITE('      IT HAS JUST MOVED TO CELL',NUMO)$
183             GO TO NEXTMOVE$
184     BACK50..RELPR=QTABLE(J)$ GO TO BUILDS0$
185     BLOCKREL..NUMO=RELPR$
186              WRITE('THE COMPUTER HAS BLOCKED A POSSIBLE')$
187              WRITE    ('       RELATED PAIR BY MOVING TO CELL',NUMO)$
188             GO TO NEXTMOVE$
189     PRESET.. I=1$ R=4*GAME-4$ J=R+1$
190      BACK14..IF J EQL 65 THEN BEGIN WRITE('THE GAME HAS ENDED IN A TIE.')$   B27
191        GO TO VERY END$                                                       E27
192             IF PTABLE(J) EQL POSITX(I) OR PTABLE(J) EQL POSITO(I) THEN
193                   BEGIN J=J+1$ I=1$ GO TO BACK14 END$  I=I+1$                B28   E28
194        IF PLAYER EQL 1 AND I GTR (OPPMOV-1)/2 THEN GO TO M115 ELSE
195        IF PLAYER EQL 0 AND I GTR  OPPMOV/2 THEN GO TO M115 ELSE
196        GO TO BACK14$
197     M115.. NUMO=PTABLE(J)$
198        WRITE('THE COMPUTER HAS MADE A PRESET MOVE TO CELL',NUMO)$
```

```
199         GO TO NEXTMOVE$
200       BUILDSQ..FOR I=(1,1,18) DO BEGIN OSQ(I)=0$ XSQ(I)=0 END$                    B29   E29
201               FOR J=(1,1,18) DO BEGIN D=4*J-3$ FOR K=(1,1,4) DO BEGIN             B30
202           M=SQFILES(D)$ OSQ(J)=OSQ(J)+OFILE(M)$ XSQ(J)=XSQ(J)+XFILE(M) $          B31
203           D=D+1 END END$                                                          E31   E30
204           FOR I=(1,1,18) DO BEGIN IF OSQ(I) LSS 8 AND XSQ(I) LSS 8 THEN BEGIN    B32
205           K=XSQ(I)$ J=OSQ(I)$ CIRC(J,K)=CIRC(J,K)+1$ M=CIRC(J,K)$                 B33
206           STAB(J,K,M)=I END END$                                                  E33   E32
207           Z=0$ DEF=0$ CH=0$
208       OFFCHK..A=0$ Z=Z+1$ IF Z GTR 17 THEN GO TO M110$
209       M105..X=MX(Z)$ Y=MY(Z)$ IF TRY1 EQL 1 THEN GO TO M111$
210       M108..
211          IF CIRC(X,Y) GTR A THEN GO TO M107 ELSE GO TO OFFCHK$
212       M107..CH=CH+1$ CHK(CH)=Z$ IF (X-Y) LSS 3 THEN BEGIN A=A+1$                  B34
213          GO TO M108 END ELSE GO TO M112$                                         E34
214       M111..IF CIRC(X,Y) GTR A THEN GO TO M112 ELSE A=0$
215       M112..A=A+1$ STABP=STAB(X,Y,A)$ GO TO M102$
216       M110..IF RELPR GTR 0 THEN GO TO BLOCKREL$ DEF=1$ Z=0$
217       DEFCHK..A=0$ Z=Z+1$ IF Z GTR 17 THEN GO TO FUTOFF$
218          X=MX(Z)$ Y=MY(Z)$ IF LOOP(5) GTR 0 AND CIRC(Y,X) GTR A THEN GO TO
219          CHANGE ELSE GO TO DEFCHK$
220       CHANGE..IF (X-Y) LSS 3 THEN GO TO FUTOFF ELSE A=A+1$ STABP=STAB(Y,X,A)$
221          GO TO M102$
222       FUTOFF..IF CH EQL 0 THEN GO TO PRESET$  DEF=0$ TRY1=1$ A=0$ FT=0$
223       M106..FT=FT+1$ IF CHK(FT) GTR 0 THEN GO TO M113$ IF TRY2 EQL 1 THEN
224          GO TO PRESET$ TRY2=1$ FT=0$ GO TO M106$
225       M113..Z=CHK(FT)$ GO TO M105$
226       FORCSQ..E=E+1$ NUMO=SPOSITO(E)$    WRITE('CONTINUATION OF FORCE             Q
227           SEQUENCE.')$  WRITE('THE COMPUTER HAS MOVED TO CELL',NUMO)$
228          GO TO NEXTMOVE$
229       M102.. FOR I=(1,1,7) DO BEGIN OWATCH(I)=0$ WATCH(I)=0 END$                  B35   E35
230                 FOR I=(1,1,40) DO OCTABLE(I)=0$
231                 FOR I=(1,1,10) DO BEGIN CELL20(I)=0$ SPOSITX(I)=0$              B36
232                    SPOSITO(I)=0     END$                                        E36
233                 D=4*STABP      -3$ FOR I=(1,1,8) DO BEGIN STABLE(I)=0$         B37
234          TTABLE(I)=0 END$ FOR R=(1,1,4) DO BEGIN                                E37
235       STABLE(R)=SQFILES(D)$ D=D+1 END$                                          B38   E38
236         J=0$ FOR R=(1,1,4) DO BEGIN D=4*STABLE(R)-3$                            B39
237          FOR I=(1,1,4) DO BEGIN K=J+I$ RCTABLE(K)=FILECELLS(D)$ D=D+1 END$      B40   E40
238          J=J+4 END$                                                             E39
239          FOR I=(1,1,4) DO STABLE(I)=0$ IF DEF EQL 1 THEN GO TO DPOS$
240       SPOS..J=1$ K=1$
241       M90..FOR I=(1,1,16) DO BEGIN IF POSITO(J) EQL RCTABLE(I) THEN              B41
242          BEGIN SPOSITO(K)=RCTABLE(I)$ STABLE(K)=I$ J=J+1$ K=K+1$                 B42
243          IF POSITO(J) EQL 0 THEN GO TO M91 ELSE GO TO M90 END END$               E42   E41
244          J=J+1$ IF POSITO(J) EQL 0 THEN GO TO M91 ELSE GO TO M90$
245        M91..J=1$ K=1$
246       M92..FOR I=(1,1,16) DO BEGIN IF POSITX(J) EQL RCTABLE(I) THEN              B43
247          BEGIN SPOSITX(K)=RCTABLE(I)$ TTABLE(K)=I$ J=J+1$ K=K+1$                 B44
248          IF POSITX(J) EQL 0 THEN GO TO M50 ELSE GO TO M92 END END$               E44   E43
249          J=J+1$ IF POSITX(J) EQL 0 THEN GO TO M50 ELSE GO TO M92$
```

```
250     DPOS..J=1$ K=1$
251     M93..FOR I=(1,1,16) DO BEGIN IF POSITX(J) EQL RCTABLE(I) THEN                B45
252        BEGIN SPOSITO(K)=RCTABLE(I)$ STABLE(K)=I$ J=J+1$ K=K+1$                   B46
253        IF POSITX(J) EQL 0 THEN GO TO M94 ELSE GO TO M93 END END$                 E46   E45
254        J=J+1$ IF POSITX(J) EQL 0 THEN GO TO M94 ELSE GO TO M93$
255      M94..J=1$ K=1$
256     M95..FOR I=(1,1,16) DO BEGIN IF POSITO(J) EQL RCTABLE(I) THEN                B47
257        BEGIN SPOSITX(K)=RCTABLE(I)$ TTABLE(K)=I$ J=J+1$ K=K+1$                   B48
258        IF POSITO(J) EQL 0 THEN GO TO M50 ELSE GO TO M95 END END$                 E48   E47
259        J=J+1$ IF POSITO(J) EQL 0 THEN GO TO M50 ELSE GO TO M95$
260     M50..FOR I=(0,1,10) DO BEGIN DOFILE(I)=0$ DXFILE(I)=0 END$  J=0$             B49   E49
261     M56..J=J+1$ IF STABLE(J) EQL 0 THEN GO TO M57$
262        D=3*STABLE(J)-2$ FOR R=(1,1,3) DO BEGIN M=SFILENOS(D)$                    B50
263        DOFILE(M)=DOFILE(M)+1$ D=D+1 END$ GO TO M56$                              E50
264     M57..J=0$
265     M58..J=J+1$ IF TTABLE(J) EQL 0 THEN GO TO M59$
266        D=3*TTABLE(J)-2$ FOR R=(1,1,3) DO BEGIN M=SFILENOS(D)$                    B51
267        DXFILE(M)=DXFILE(M)+1$ D=D+1 END$ GO TO M58$                              E51
268     M59..FOR I=(1,1,5) DO FILE20(I)=0$ J=1$
269        IF TRY2 EQL 1 THEN BEGIN  FOR I=(1,1,10) DO IF DOFILE(I) EQL 1            B52
270        AND DXFILE(I) EQL 0 THEN BEGIN FILE10=I$ D=4*I-3$ FOR J=(1,1,4) DO        B53
271        BEGIN M=CELLNOS(D)$ DCTABLE(J)=RCTABLE(M)$ D=D+1 END$ FOR I=(1,1,7)       B54   E54
272        DO IF SPOSITO(I) EQL DCTABLE(1) THEN GO TO M99$
273        NUMO=DCTABLE(1)$ GO TO M101$
274     M99..NUMO=DCTABLE(2)$
275     M101..WRITE('THE COMPUTER HAS MADE A 2-0 FILE IN A PROMISING')$
276         WRITE('   SQUARE BY MOVING TO CELL',NUMO)$
277        GO TO NEXTMOVE END END$                                                   E53   E52
278        IF TRY2 EQL 1 THEN GO TO M106$
279        FOR I=(1,1,10) DO IF DXFILE(I) EQL 0 AND DOFILE(I) EQL 2 THEN BEGIN
280        FILE20(J)=I$ J=J+1 END$ CNTFIL=J-1$ IF CNTFIL EQL 0  AND TRY1 EQL 0       B55   E55
281        THEN BEGIN
282        WRITE(' NO FORCE EXIST IN SQUARE',STABP)$ GO TO CYC END$                  B56   E56
283        CNTSQ=1$ GO TO M3$
284     M3..K=0$FOR I=(1,1,CNTFIL) DO BEGIN D=4*FILE20(I)-3$                         B57
285        FOR J=(1,1,4) DO BEGIN R=J+K$ M=CELLNOS(D)$                               B58
286         DCTABLE(R)=RCTABLE(M)$ D=D+1 END$ K=K+4 END$                             E58   E57
287        J=0$ K=1$
288     M4..J=J+1$ I=0$         IF DCTABLE(J) EQL 0 THEN GO TO M40$
289     M5..I=I+1$ IF SPOSITO(I) EQL 0 THEN BEGIN CELL20(K) =DCTABLE(J)$ K=K+1$   B59
290        GO TO M4 END $                                                            E59
291        IF SPOSITO(I) EQL DCTABLE(J) THEN GO TO M4 ELSE GO TO M5$
292     M40.. J=0$ C=0$
293         M=4*CNTFIL$ FOR I=(1,1,M) DO DCTABLE(I)=0$
294     M71.. J=J+1$ IF OQCELLS(J) EQL 0 THEN GO TO M80$
295        FOR I=(1,1,16) DO IF RCTABLE(I) EQL OQCELLS(J) THEN BEGIN
296        IF DEF EQL 0 THEN BEGIN                                                   B60
297        R=CNTFIL*2$ FOR K=(1,1,R) DO BEGIN IF RCTABLE(I) EQL                      B61   B62
298        CELL20(K) THEN GO TO M71 END END$C=C+1$                                   E62   E61
299        OWATCH(C)=RCTABLE(I)$ GO TO M71 END$ GO TO M71$                           E60
300     M80.. J=0$ C=0$
```

```
301     M41..J=J+1$ IF QTABLE(J) EQL 0 THEN GO TO M6$
302        FOR I=(1,1,16) DO IF RCTABLE(I) EQL QTABLE(J) THEN BEGIN
303        IF DEF EQL 1 THEN BEGIN                                          B63
304        R=CNTFIL*2$ FOR K=(1,1,R) DO BEGIN IF RCTABLE(I) EQL             B64   B65
305        CELL20(K) THEN GO TO M41 END END$C=C+1$                          E65   E64
306        WATCH(C)=RCTABLE(I)$ GO TO M41 END$ GO TO M41$                   E63
307     M6..COUNT20=2*CNTFIL$
308     WRITE('RCTABLE',RCTABLE)$
309     WRITE('SPOSITO',SPOSITO)$
310     WRITE('SPOSITX',SPOSITX)$
311     WRITE('WATCH',WATCH)$
312     WRITE('OWATCH',OWATCH)$
313        IF TRY1 EQL 1 AND OWATCH(1) GTR 0 THEN BEGIN NUMO=OWATCH(1)$     B66
314        WRITE('THE COMPUTER HAS MADE A 3-0 FILE IN HOPES OF STARTING')$
315         WRITE('   A FORCING SEQUENCE. IT HAS MOVED TO CELL',NUMO)$
316        GO TO NEXTMOVE END$                                              E66
317        IF TRY1 EQL 1 THEN GO TO M106$
318        FOR I=1,4,13,16 DO BEGIN FOR J=(1,1,COUNT20) DO BEGIN            B67
319        IF RCTABLE(I) EQL CELL20(J) THEN GO TO M7 END END$               B68   E68   E67
320        GO TO CNTF$
321     M7.. K=CELL20(1)$ R=CELL20(2)$ CELL20(1)=CELL20(J)$
322        IF J EQL 1 OR J EQL 3 OR J EQL 5 OR J EQL 7 THEN N=J+1 ELSE N=J-1$
323        CELL20(2)=CELL20(N)$ CELL20(J)=K$ CELL20(N)=R$
324     CNTF.. IF CNTFIL LSS 2 THEN GO TO  CNSQ$
325        J=1$ K=2$
326     M9..IF CELL20(J) EQL CELL20(K) THEN GO TO DREL$ K=K+1$
327        IF CELL20(K) GTR 0 THEN GO TO M9$ J=J+1$ K=J+1$
328           IF CELL20(K) GTR 0 THEN GO TO M9$
329                                 GO TO CNSQ$
330     DREL..E=X+1$ IF DEF EQL 1 THEN GO TO DEF03$
331           NUMO=SPOSITO(E)$FORCESEQ=1$
332        WRITE('THE COMPUTER HAS STARTED A FORCE BY MOVING TO CELL',NUMO)$
333        WRITE('SPOSITX=,SPOSITO=,TREE(CNTSQ,1)=,CNT=,CNTFIL=,CNTSQ=',
334        SPOSITX, SPOSITO, TREE(CNTSQ,1),CNT, CNTFIL, CNTSQ)$
335        WRITE(CELL20)$
336        N=Q+1$ SPOSITO(N)=CELL20(J)$ GO TO NEXTMOVE$
337     DEF03..NUMO=SPOSITO(E)$
338        WRITE('BLOCKED A POSSIBLE FORCE BY MOVING TO CELL',NUMO)$
339        GO TO NEXTMOVE$
340     CNSQ.. FOR I=(1,1,10)       DO TREE(CNTSQ,I)=CELL20(I)$
341     M27..CNT=1$ Q=CNTSQ+X$
342     M12..SPOSITO(Q)=CELL20(CNT)$
343        J=0$ FOR I=(0,1,10) DO BEGIN DOFILE(I) =0$ DXFILE(I)=0 END$      B69   E69
344     M13..J=J+1$ IF SPOSITO(J) EQL 0 THEN GO TO M14$
345       FOR I=(1,1,16) DO IF RCTABLE(I) EQL SPOSITO(J)
346        THEN BEGIN D=3*I-2$ FOR R=(1,1,3) DO BEGIN  M=SFILENOS(D)$      B70   B71
347        DOFILE(M)=DOFILE(M)+1$ D=D+1 END$GO TO M13 END $                 E71   E70
348     M14..J=0$
349     M15..J=J+1$ IF SPOSITX(J) EQL 0 THEN GO TO M16$
350       FOR I=(1,1,16) DO IF RCTABLE(I) EQL SPOSITX(J)
351        THEN BEGIN D=3*I-2$ FOR R=(1,1,3) DO BEGIN  M=SFILENOS(D)$      B72   B73
```

```
352         DXFILE(M)=DXFILE(M)+1$ D=D+1 END$ GO TO M15 END$                    E73   E72
353       M16.. FOR I=(1,1,5)       DO FILE20(I)=0$ J=1$
354         FOR I=(1,1,10) DO IF DXFILE(I) EQL 0 AND DOFILE(I) EQL 2 THEN
355         BEGIN FILE20(J)=I$ J=J+1 END$ CNTFIL=J-1$ IF CNTFIL GTR 0 THEN     B74   E74
356         GO TO BK3$ CNT2=0$
357       M17..CNT=CNT+1$ SPOSITO(Q)=0$ IF CNT2 EQL 1 THEN BEGIN K=CNTSQ+Y$   B75
358         SPOSITX(K)=0 END$ IF CNT GTR COUNT20 THEN GO TO REDO$ N=CNT-1$     E75
359         FOR I=(1,1,10) DO BEGIN IF TREE(CNTSQ,I) GTR 0 AND                 B76
360          TREE(CNTSQ,I) LSS 1000 THEN BEGIN SAVETR=TREE(CNTSQ,I)$           B77
361          TREE(CNTSQ,I)=1000$ GO TO M12 END END$                            E77   E76
362       BK3..N=CNT$ J=N+1$ I=N-1$ R=COUNT20$ K=CNTSQ+Y$
363         IF R EQL 2 OR R EQL 4 OR R EQL 6  OR R EQL 8 THEN BEGIN
364         IF N EQL 1 OR N EQL 3 OR N EQL 5 OR N EQL 7 THEN BEGIN             B78
365         SPOSITX(K)=CELL20(J)$ GO TO M21 END ELSE SPOSITX(K)= CELL20(I)$    B79   E79
366         GO TO M21 END$                                                     E78
367         IF N EQL 1 THEN BEGIN SPOSITX(K)=SAVETR$ GO TO M21 END$            B80   E80
368         IF N EQL 3 OR N EQL 5 OR N EQL 7 THEN SPOSITX(K)=CELL20(I) ELSE
369         SPOSITX(K)=CELL20(J)$ GO TO M21$
370       M21..IF C GTR 0 THEN FOR I=(1,1,C) DO IF DEF EQL 0 THEN IF SPOSITX(K)
371         EQL WATCH(I) THEN BEGIN CNT2=1$ GO TO M17 END ELSE IF SPOSITX(K) EQL B81   E81
372         OWATCH(I) THEN BEGIN CNT2=0$ GO TO M17 END$                        B82   E82
373         FOR I=(1,1,16) DO IF RCTABLE(I) EQL SPOSITX(K) THEN BEGIN
374         D=3*I-2$ FOR R =(1,1,3) DO BEGIN  M=SFILENOS(D)$                   B83   B84
375         DXFILE(M)=DXFILE(M)+1$ D=D+1 END$ GO TO M22 END$                   E84   E83
376       M22..J=0$
377       M23..J=J+1$ IF J GTR 10 THEN GO TO M25$
378         IF DXFILE(J) EQL 3 AND DOFILE(J) EQL 0 THEN GO TO M24 ELSE
379         GO TO M23$
380       M24..CNT2=1$ GO TO M17$
381       M25..FOR I=(1,1,10)       DO CELL20(I)=0$
382         FOR I=(1,1,40) DO DCTABLE(I)=0$
383         K=0$ FOR I=(1,1,CNTFIL) DO BEGIN D=4*FILE20(I)-3$                  B85
384         FOR J=(1,1,4) DO BEGIN R=J+K$ M=CELLNOS(D)$                        B86
385         DCTABLE(R)=RCTABLE(M)$ D=D+1 END$                                  E86
386         K=K+4 END$                                                         E85
387         J=0$ K=1$
388       M26..J=J+1$ I=0$          IF DCTABLE(J) EQL 0 THEN GO TO M33$
389       M32..I=I+1$ IF SPOSITO(I) EQL 0 THEN BEGIN CELL20(K) =DCTABLE(J)$ K=K+1$ B87
390         GO TO M26 END$                                                     E87
391         IF SPOSITO(I) EQL DCTABLE(J) THEN GO TO M26 ELSE GO TO M32$
392       M33..COUNT20=2*CNTFIL$ CNTSQ=CNTSQ+1$
393         IF DEF EQL 1 THEN GO TO M73$   K=1$
394       M75..IF OWATCH(K) EQL 0 THEN GO TO CNTF$  J=1$
395       M74..IF CELL20(J) EQL OWATCH(K) THEN GO TO DREL$
396         J=J+1$ IF CELL20(J) GTR 0 THEN GO TO M74 ELSE K=K+1$ GO TO M75$
397       M73..K=1$
398       M76..IF WATCH(K) EQL 0 THEN GO TO CNTF$  J=1$
399       M77..IF CELL20(J) EQL WATCH(K) THEN GO TO DREL$
400         J=J+1$ IF CELL20(J) GTR 0 THEN GO TO M77 ELSE K=K+1$ GO TO M76$
401       REDO..FOR I=(1,1,10)      DO CELL20(I)=0$
402             FOR I=(1,1,10)      DO TREE(CNTSQ,I)=0$
```

```
        403       M28..CNTSQ=CNTSQ-1$
        404          IF CNTSQ EQL 0 THEN BEGIN
        405          WRITE('THERE IS NO CONTINUOUS FORCE IN SQUARE',STARP)$                    B88
        406       CYC..IF CIRC(X,Y) GTR A AND DEF EQL 0 THEN GO TO M107$
        407          IF CIRC(Y,X) GTR A AND DEF EQL 1 THEN GO TO CHANGE$
        408            IF DEF EQL 0 THEN GO TO OFFCHK ELSE GO TO DEFCHK$ END$                  E88
        409          FOR I=(1,1,10) DO BEGIN IF TREE(CNTSQ,I) GTR 0 AND                        B89
        410           TREE(CNTSQ,I) LSS 1000  THEN BEGIN SAVETR=TREE(CNTSQ,I)$                B90
        411          TREE(CNTSQ,I)=1000$ GO TO M30 END END$                                   E90    E89
        412       M29..FOR I=(1,1,10) DO TREE(CNTSQ,I)=0$ R=CNTSQ+X$ J=CNTSQ+Y$
        413          SPOSITX(J)=0$ SPOSITO(R)=0$ GO TO M28$
        414       M30..R=CNTSQ+X$ SPOSITO(R)=0$ J=CNTSQ+Y$ SPOSITX(J)=0$
        415          FOR J=(1,1,10) DO BEGIN I=I+1$ CELL20(J)=TREE(CNTSQ,I)$                  B91
        416          IF CELL20(J) EQL 1000 THEN BEGIN  CELL20(J)=0$ GO TO M31 END$            B92    E92
        417          IF CELL20(J) EQL 0 THEN GO TO M31 END$                                   E91
        418       M31..COUNT20=J-1$ IF J EQL 1 OR J EQL 3 OR J EQL 5 OR J EQL 7 THEN
        419          J=J+1$ CNTFIL=J/2$
        420          GO TO M27$
        421       VERY.. WRITE('POSITO TABLE')$
        422          FOR I=(1,1,(OPPMOV/2)) DO WRITE(POSITO(I))$
        423       WRITE('POSITX TABLE')$
        424          FOR I=(1,1,(OPPMOV/2)) DO WRITE(POSITX(I))$
        425          GO TO NXGM$
        426       COMPL..FINISH                                                               F0
  END BLOCK 1
COMPILATION COMPLETED
```

COMPLETE LISTING OF DATA- INCLUDING INFORMATION IN THE LISTS.

SQFILES LIST

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 76 | 1 | 2 | 3 | 10 | 11 | 12 | 13 | 20 | 21 |
| 22 | 23 | 30 | 31 | 32 | 33 | 76 | 10 | 20 | 30 |
| 1 | 11 | 21 | 31 | 2 | 12 | 22 | 32 | 3 | 13 |
| 23 | 33 | 4 | 14 | 24 | 34 | 5 | 15 | 25 | 35 |
| 6 | 16 | 26 | 36 | 7 | 17 | 27 | 37 | 4 | 15 |
| 26 | 37 | 7 | 16 | 25 | 34 | 76 | 11 | 22 | 33 |
| 3 | 12 | 21 | 30 | 8 | 18 | 28 | 38 | 9 | 19 |
| 29 | 39 | | | | | | | | |

SFILENOS LIST

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 5 | 9 | 1 | 6 | 0 | 1 | 7 | 0 | 1 |
| 8 | 10 | 2 | 5 | 0 | 2 | 6 | 9 | 2 | 7 |
| 10 | 2 | 8 | 0 | 3 | 5 | 0 | 3 | 6 | 10 |
| 3 | 7 | 9 | 3 | 8 | 0 | 4 | 5 | 10 | 4 |
| 6 | 0 | 4 | 7 | 0 | 4 | 8 | 9 | | |

FILENOS LIST

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 76 | 4 | 8 | 40 | 56 | 60 | 68 | 1 | 4 | 41 |
| 69 | 0 | 0 | 0 | 2 | 4 | 42 | 70 | 0 | 0 |
| 0 | 3 | 4 | 9 | 43 | 62 | 66 | 71 | 75 | 5 |
| 44 | 57 | 0 | 0 | 0 | 1 | 5 | 8 | 45 | 0 |
| 0 | 0 | 2 | 5 | 9 | 46 | 0 | 0 | 0 | 3 |
| 5 | 47 | 63 | 0 | 0 | 0 | 76 | 6 | 48 | 58 |
| 0 | 0 | 0 | 1 | 6 | 9 | 49 | 0 | 0 | 0 |
| 2 | 6 | 8 | 50 | 0 | 0 | 0 | 3 | 6 | 51 |
| 64 | 0 | 0 | 0 | 76 | 7 | 9 | 52 | 59 | 61 |
| 72 | 1 | 7 | 53 | 73 | 0 | 0 | 0 | 2 | 7 |
| 54 | 74 | 0 | 0 | 0 | 3 | 7 | 8 | 55 | 65 |
| 67 | 75 | 10 | 14 | 18 | 40 | 0 | 0 | 0 | 11 |
| 14 | 41 | 56 | 0 | 0 | 0 | 12 | 14 | 42 | 62 |
| 0 | 0 | 0 | 13 | 14 | 19 | 43 | 0 | 0 | 0 |
| 10 | 15 | 44 | 68 | 0 | 0 | 0 | 11 | 15 | 18 |
| 45 | 57 | 60 | 69 | 12 | 15 | 19 | 46 | 63 | 66 |
| 70 | 13 | 15 | 47 | 71 | 0 | 0 | 0 | 10 | 16 |
| 48 | 72 | 0 | 0 | 0 | 11 | 16 | 19 | 49 | 58 |
| 61 | 73 | 12 | 16 | 18 | 50 | 64 | 67 | 74 | 13 |
| 16 | 51 | 75 | 0 | 0 | 0 | 10 | 17 | 19 | 52 |
| 0 | 0 | 0 | 11 | 17 | 53 | 59 | 0 | 0 | 0 |
| 12 | 17 | 54 | 65 | 0 | 0 | 0 | 13 | 17 | 18 |
| 55 | 0 | 0 | 0 | 20 | 24 | 28 | 40 | 0 | 0 |
| 0 | 21 | 24 | 41 | 62 | 0 | 0 | 0 | 22 | 24 |
| 42 | 56 | 0 | 0 | 0 | 23 | 24 | 29 | 43 | 0 |
| 0 | 0 | 20 | 25 | 44 | 72 | 0 | 0 | 0 | 21 |

```
 25  28  45  63  67  73  22  25  29  46
 57  61  74  23  25  47  75   0   0   0
 20  26  48  68   0   0   0  21  26  29
 49  64  66  69  22  26  28  50  58  60
 70  23  26  51  71   0   0   0  20  27
 29  52   0   0   0  21  27  53  65   0
  0   0  22  27  54  59   0   0   0  23
 27  28  55   0   0   0  30  34  38  40
 62  67  72  31  34  41  73   0   0   0
 32  34  42  74   0   0   0  33  34  39
 43  56  61  75  30  35  44  63   0   0
  0  31  35  38  45   0   0   0  32  35
 39  46   0   0   0  33  35  47  57   0
  0   0  30  36  48  64   0   0   0  31
 36  39  49   0   0   0  32  36  38  50
  0   0   0  33  36  51  58   0   0   0
 30  37  39  52  65  66  68  31  37  53
 69   0   0   0  32  37  54  70   0   0
  0  33  37  38  55  59  60  71
FILECELLS LIST
  2   6  10  14   3   7  11  15   4   8
 12  16   1   2   3   4   5   6   7   8
  9  10  11  12  13  14  15  16   1   6
 11  16   4   7  10  13  17  21  25  29
 18  22  26  30  19  23  27  31  20  24
 28  32  17  18  19  20  21  22  23  24
 25  26  27  28  29  30  31  32  17  22
 27  32  20  23  26  29  33  37  41  45
 34  38  42  46  35  39  43  47  36  40
 44  48  33  34  35  36  37  38  39  40
 41  42  43  44  45  46  47  48  33  38
 43  48  36  39  42  45  49  53  57  61
 50  54  58  62  51  55  59  63  52  56
 60  64  49  50  51  52  53  54  55  56
 57  58  59  60  61  62  63  64  49  54
 59  64  52  55  58  61   1  17  33  49
  2  18  34  50   3  19  35  51   4  20
 36  52   5  21  37  53   6  22  38  54
  7  23  39  55   8  24  40  56   9  25
 41  57  10  26  42  58  11  27  43  59
 12  28  44  60  13  29  45  61  14  30
 46  62  15  31  47  63  16  32  48  64
  1  18  35  52   5  22  39  56   9  26
 43  60  13  30  47  64   1  22  43  64
 13  26  39  52   4  19  34  49   8  23
 38  53  12  27  42  57  16  31  46  61
  4  23  42  61  16  27  38  49   1  21
 41  61   2  22  42  62   3  23  43  63
  4  24  44  64  13  25  37  49  14  26
 38  50  15  27  39  51  16  28  40  52
  1   5   9  13
```

CELLNOS LIST

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| 11 | 12 | 13 | 14 | 15 | 16 | 1 | 5 | 9 | 13 |
| 2 | 6 | 10 | 14 | 3 | 7 | 11 | 15 | 4 | 8 |
| 12 | 16 | 1 | 6 | 11 | 16 | 4 | 7 | 10 | 13 |

PTABLE LIST

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 10 | 20 | 30 | 40 | 50 | 60 | 58 | 48 | 38 | 28 |
| 18 | 8 | 3 | 13 | 23 | 33 | 43 | 53 | 63 | 57 |
| 47 | 37 | 27 | 17 | 7 | 51 | 41 | 31 | 21 | 11 |
| 1 | 61 | 9 | 19 | 29 | 39 | 49 | 59 | 62 | 52 |
| 42 | 32 | 22 | 12 | 2 | 64 | 54 | 44 | 34 | 24 |
| 14 | 56 | 46 | 36 | 26 | 16 | 6 | 4 | 5 | 15 |
| 25 | 35 | 45 | 55 | | | | | | |

MX SWITCH LIST

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 5 | 6 | 7 | 4 | 5 | 6 | 7 | 3 | 4 | 5 |
| 6 | 7 | 2 | 3 | 4 | 5 | 6 | | | |

MY SWITCH LIST

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 0 | 1 | 2 | 3 | 0 | 1 | 2 |
| 3 | 4 | 0 | 1 | 2 | 3 | 4 | | | |

BIBLIOGRAPHY


1. C.E. Shannon, "Programming a Computer for Playing Chess", Phil. Mag. 41, 256 (March, 1950).

2. J. Kister, P. Stein, S. Ulam, W. Walden, M. Wells, "Experiments in Chess", Journal of the ACM 4, 174 (April, 1957).

3. A. Bernstein, "Program for the IBM 704", Proc. 1958 Western Joint Computer Conference, (May 1958).

4. A. Newell, J.C. Shaw, H.A. Simon, "Chess Playing Programs and the Problem of Complexity", IBM Journal of Research and Development, 2, 320 (October, 1958).

5. A. Samuel, "Some Studies in Machine Learning Using the Game of Checkers", IBM Journal of Research and Development, 3, 210 (July, 1958).

6. W.G. Daly, "Computer Strategies for the Game of Qubic", Master's Thesis, MIT, (February, 1961).

7. D. Lefkovitz, "A Strategic Pattern Recognition Program for the Game Go", Project No. 7062, University of Pennsylvania (July, 1960).

8. M.D. Mesarovic, "Toward a Formal Theory of Problem Solving" ONR Conference of Computer Augumentation of Human Reasoning, Washington D.C., (June, 1964).

9. M. Minsky, "Steps Toward Artificial Intelligence", Proc. IRE Vol. 49, No. 1, (January 1961).

10. E.A. Feigenbaum and J. Feldman, Computers and Thought, McGraw-Hill Book Co., New York, 1963.

11. T.G. Windeknecht, "Problem Solving and Finite Automata", Case Institute of Technology, 1964.

12. J. Von Neuman and O. Morgenstern Theory of Games and Economic Behavior, Princeton Univ. Press, 1947.

13. G. Miller, "A Tic-Tac-Toe Learning Program", Johns Hopkins University (unpublished), 1963.

14. R.L. Wexelblat, "A Simple Program for Computer Learning Based on the Game of Renjyu", Master's Thesis, University of Pennsylvania, (April, 1961).

15. A.M. Turing, "Computing Machinery and Intelligence", Mind, (October, 1959)

16. K.H. Simon, and P. Simon, "Trial and Error Search in Solving Difficult Problems: Evidence from the Game of Chess", Carnegie Institute of Technology, 1962.

# ERRATA

| Error Location | | Correct Entry |
|---|---|---|
| Page | | |
| 7 | Line 10 | ...would involve... |
| 16 | Line 2 | ...function f as a mapping of a Cartesian... |
| 17 | Line 3 | $\Gamma: \begin{cases} s_o = b \\ s_{2i-1} = f(s_{2i-2}, x_i) \\ s_{2i} = g(_{2i-1}, y_i) \end{cases}$ |
| 17 | Next to last line | ...as one which yields a state in $\Omega_T$. |
| 20 | Line 4, third paragraph | ...$b_{2p+1} \in \Omega_T$ if $b_q \in \Omega_s$,... |
| 32 | Figure 3.2 | Cartesian... |
| 33 | Line 2 | ...one is... |
| 33 | Line 5 | ...but stopping to explore at... |
| 36 | Line 16 | ...above are not... |
| 48 | Line 2 | ...three 1-0 files... |
| 73 | Line 4 | ...made on the basis of the excess... |
| 82 | Line 4 | The situation... |
| 82 | Line 7 | ...lead to terminal... |
| 87 | Line 3 | ...Table IV... |
| 87 | Line 5 | ...prove to be valuable... |
| 87 | Flow chart, second block | ...of three cells... |