TOWARD A MODEL OP HUMAN GAME PLAYING [1,2]

Marc Eisenstadt
Yaakov Kareev
Department of Psychology
University of California, San Diego
La Jolla, California, 92037

Abstract

This paper presents an overview of a game playing model which is based on human perceptual and problem solving abilities. The representation of games, acquisition of rules, learning of strategies, and selection of moves are outlined. Details *of* move selection, including scanning of the board, use of familiar patterns to suggest *move* candidates, evaluation of moves, and lookahead are described. Finally, there is a discussion of the means by which the model learns to play a better game.

Introduction

People can be taught a board game by means of natural language instruction, which may or may not be accompanied by a demonstration. Moves made by beginners are almost always legal, often bad, and probably never random. People can improve the quality of their play with experience, and can also be taught (by a book or a tutor) particular moves, patterns of pieces, or strategic concepts. A model of human game playing should exhibit all of these characteristics. One possible model is developed in this paper. The major assumptions of the model are being tested through experiments with human game players. In this paper, we cite our experimental studies in the footnotes.

Our model is embedded in the framework of a larger model of human information processing which has been described in Rumelhart, Lindsay & Norman[19] and Rumelhart & Norman[20]. The database structure is that of a semantic network consisting of nodes connected by bidirectional, labelled relations. A parser translates English into the database. A programming language, SOL, wnich is a subset of English, allows the user to interact with the database. There is no inherent distinction made between data and procedures. Although this paper illustrates many of the mechanisms necessary for the implementation of the game playing model, the complete implementation has not yet been accomplished.

Acquiring the Rules of a New Game

In learning to play a game there are two distinct stages: defining the task (learning the rules of the game), and searching for a solution (playing the game to win). In order to start playing a game, knowledge of three things is necessary: the goal of the game; the universe within which the game takes place; and the rules governing actions within that universe. Two questions arise within this context: (a) What does it take to understand the rules of a game? (b) How are these rules represented once they are acquired?

Understanding the Rules of a Game

It is always possible to state the rules of a game in English. Most people can understand such rules and can start playing a legal game once they have heard them. Clearly, these rules would be meaningless if the words comprising them were not understood. The study of how this understanding is acquired constitutes a major investigation in its own right that will not be performed here. For our model, we assume that this level of understanding has already been achieved. This assumption has two important implications; First, all the concepts considered to be important for understanding the rules of games are incorporated into the model (i.e., entered by hand) in advance. Second, the model only applies to the behavior of adults who can be assumed to possess this knowledge.

The Internal Representation of a Game

Although there is a large variety of board games, all games have much in common. The existence of a general structure which represents the common features is implicitly assumed whenever the rules of a game are described. Building an initial internal representation of a game consists of filling in specific arguments in the variable slots of that general frame. Those variables can be anything from single values to complex structures. Appendix A gives our representation of the frame for games.

Generating Primitive Strategies

Mere knowledge of the rules of the game would result in the generation of random (though legal) moves. But even beginners of a game do not play randomly.3 The nonrandom *nature* of playing is inherent in game playing, since games are by definition a goal directed activity. As the game starts, the player must already have a set of primitive strategies for playing that particular game.

For example, "capture more pieces" is a primitive strategy for checkers, while "get more in a row" is a primitive strategy for Gomoku.4 These primitive strategies have in common the notion of making progress on some dimension such as "capture" or "number of pieces in a row." The dimensions for which primitive strategies are generated are those explicitly mentioned in the rules of the game. A meta-strategy is needed to determine how "progress" is to be characterized for each dimension. One way to do this is to apply means-ends analysis in order to see whether a goal state for a particular dimension may be attained by successive applications of a single operation. If so, that operation is singled out as a primitive strategy. The symmetric nature of most games allows the creation of complementary primitive strategies for impeding the opponent's progress on a given dimension.

Move Selection

Here is a quick overview of the flow of processing performed by our model following the opponent's move. First, it tries to find a reason for the opponent's move: the representation of the current situation is updated in light of that reason. The model may then either respond to the opponent's move, continue with the execution of a previously created plan, or generate a new plan.

The evaluation of newly created patterns identified by a scan of the board singles out the reason for the

opponent's move. Plans are sequences of moves leading to a situation which is more favorable than the current one. The moves within the plan sequences are originally suggested either by familiar patterns or by strategies. The suggested moves undergo evaluation during a lookahead process whose depth is limited by the constraints of a working memory.

Of necessity, all of these processes interact heavily. It will be convenient, however, to discuss individually the following segments of the model: scanning the board, familiar patterns, move evaluation, lookahead, and finding a reason for the opponent's move.

Scanning the Board

The game board serves as an external memory for human players. This external memory is simulated by a 2-dimensional array which remains conceptually distinct from the model. The problem of scanning the board may be thought of as a problem of parsing a 2-dimensional string. Top-down parsing algorathms which have been used as a mechanism for the analysis of line drawings[9, 1, 15,] capture the predictive power and analysis-by-synthesis nature of human visual processing. Bottom-up parsers (e.g., Ledley) capture the essence of organization of visual scenes for which there are no expectations. They may also miss important configurations, which is a disadvantage for an optimal game playing machine, but a necessity for a realistic model of human game playing.

The scanning routine used by our model begins with a bottom-up parse of the board, but becomes a top-down parse as soon as enough information is gathered to generate some expectations. The scan iterates through all board locations whose contents have changed as a result of the last move, treating each such location as the origin of a bottom-up parse. When scanning from a given origin the model peripherally notices all the immediate neighbors of that origin, i.e., all pieces within the 3x3 area containing the origin and its neighbors. It identifies those combinations which either constitute complete patterns or are possible members of some larger patterns.

If a larger pattern is marked as potentially present, the scan then switches to its top-down phase and actively tries to find that larger pattern. The identification of patterns is made by network matching routines which compare the current view with a network containing descriptions of familiar patterns.

The top-down parse starts with an attempt to identify familiar patterns derived from Gestalt principles of human perception. The three Gestalt principles included in the model are those of proximity, similarity, and continuity.11 Proximity is incorporated by having the scan move outwards from the origin in expanding concentric circles. The principle of similarity is embodied by marking "piece next to a similar piece" as a familiar pattern for all games. The principle of continuity is incorporated by regarding lines of pieces as inherently familiar patterns. Consequently, a potential line of pieces seen in the initial 3x3 area will result in the initiation of a top-down search for that line.

Note that although patterns are "familiar," they are not necessarily meaningful within the context of a particular game. Thus, a row of four bishops side by side on a chess board will be called "familiar" by the Gestalt principles, although the intrinsic importance of such a configuration is not at all clear.

Once simple Gestalt groupings are recognized, the top-down scan continues looking for other potential patterns (if other potential ones are indicated). The remaining ones are those which are important within the context of the specific game currently being played.[6]

Familiar Patterns

The ability to recognize familiar configurations of pieces on the board is crucial for the playing of any game. A person who recognizes a familiar pattern can refer to its representation in his long-term memory by using a single label, thus freeing valuable space. Additional savings are obtained if the pattern is associated with a well known sequence of suggested moves, since these suggestions need not be generated in the normal lookahead manner. A large repertoire of familiar patterns is one of the characteristics of skillful players.8,23

Patterns are represented in the model by means of active semantic networks of the type described in Rumelhart G Norman.[20] Such networks may be treated either as data to be consulted during the recognition of patterns (i.e., as a table of rewrite rules during a visual parse) or as procedures which themselves direct the recognition process. In fact, the current scanning algorithm invokes the procedural definitions during its top-down phase, and treats the same definitions as data during its bottom-up phase.

This format of representation appears to be both powerful and general. If we represent specific locations and pieces, the representation becomes equivalent to that of Simon and Gilmartin's MAPP program.[23] Non-specific locations and orientations allow us to have internal representations which are homomorphic to the real world configurations. This was the type of representation used by Murray and Elcock for Gomoku.[17] Recursive definitions of groups of stones in Go, found useful by Ryder,-* are also easy to handle in this format. A sequence of suggested moves for a familiar pattern is represented as a procedure which, when executed, returns with advice relevant within the context of a particular game. For example, if the pattern "four in a row" is discovered in the game of Gomoku, a procedure is activated which looks for a vacant square next to and in a line with the pattern, and recommends moving to that square. Note how this contrasts with the "meaningless" chess pattern of four bishops in a row described earlier.

Evaluation

The evaluation of moves is based on the dimensions stored under a "strategy" node. For example, on the dimension of "length," a string of five in a row has a greater value than a string of four in a row. If we are comparing two moves along the dimension of "capture value," we might compare the number of pieces captured [e.g., in checkers), or the numerical value which some book or tutor has told us to assign to the captured pieces (e.g., in chess). Relevant dimensions are stored explicitly under the strategy node, but are different for different games.

Actual numerical values are not computed in the current model, and at present there is no ordering of the sequence in which different dimensions are encountered, but we allow an ordering of the nodes within a given dimension.

When a pattern has been found by the scan, candidate moves will be suggested if there are explicit recommendations associated with the pattern. If such recommendations are not found, the strategy node provides suggestions by pointing out a dimension and a way of increasing values on that dimension {e.g., adding a piece next to its neighbors to increase length).

A pairwise comparison of successively generated candidates is made along relevant dimensions (as suggested by strategies) in order to choose the best move. If, during the course of these comparisons, a candidate is found which has a value above "satisficing"[18] on any dimension, the evaluation terminates and that-candidate becomes the model's choice for the next move.

When trying to find the best of two candidate moves, it is possible that they will have the same values on the dimension on which they are compared, or that these

values can only be determined if they lead to a familiar configuration of pieces within several moves. If this is the case, we call a lookahead procedure which looks for recommended moves for the opponent and compares the outcomes of these recommended moves (which itself may require another call to the lookahead procedure). Since there are normally several alternative moves, the *only* recommended moves which are considered for evaluation (and possible further lookahead) are those which have the highest probability of being realized. If we could objectively decide on a move for the opponent using an unbiased evaluation scheme, then our move selection technique would be formally equivalent to minimaxing. People typically do not perform this type of analysis, since they are subject to a bias when trying to carry out a plan; they may assume that pieces only have a particular role (e.g., offensive or defensive) or only move with a specific intent (e.g., "he is trying to capture my bishop"). In the model, bias operates by instructing the scanning routine to *look only for* pieces which fulfill certain roles. In this way, we still choose "best" moves for the opponent, but bias yields what we call "subjective minimaxing."

Lookahead

The model's actual lookahead may be regarded as a product of many interacting processes. These processes (which we will discuss in detail below) include: an active working memory which effectively limits the depth of lookahead; heuristics for narrowing the selection of move candidates; a mechanism for making hypothetical moves; a limited backup capability which restricts the model to a "progressive deepening" search. The actual lookahead typically proceeds by means of a forward search, but capability for means-ends analysis is also provided. Plans are treated as an outcome of the lookahead process.

Working Memory. All active processing takes place within the confines of a limited size working memory. The size restriction is regarded as an asset to the model, as it necessitates garbage-collection and chunking of information. The contents of working memory is a homogeneous mixture of data, procedures, and pointers to items in long-term memory. Achieving a given performance level (e.g., remembering 7±2 random digits[16]) depends upon a model's specific implementation. We are currently considering different size limitations, and the final choice is likely to be in the order *of* hundreds of nodes. This conceptualization is an adaptation of the one proposed by Newell and Simon.[18]

Heuristic Candidate Selection. The selection of only a few move candidates for further lookahead is an automatic result of the way in which these candidates are suggested. The scan of the board concentrates on those areas which have changed as the result of a move. Some candidate moves are suggested *by* familiar patterns which result from this localized scan. Other candidate moves are suggested by advice from different dimensions known to the strategy node (e.g., "increase length"). Finally, if good move candidates which were previously considered are still in working memory, a re-analysis of these candidates may be performed. The number of candidates is often small in highly constrained situations, such as dynamic piece exchange. Since working memory size is constant, a small number of candidates enhances the depth of lookahead.

Hypothetical Moves. Since the array representing the board is regarded as an external memory, imaginary moves do not affect this array. A hypothetical move is made by asserting a proposition which includes (among other things) a "beginning-state," an "end-state," and the move which causes the transition between the two states. The end-state for a given move is the result of augmenting the beginning-state for that move with the changes caused by that move- The beginning-state for a move is simply the end-state of the preceding move, if such a preceding move existed. The beginning-state for the first imaginary move is empty, since no changes have yet been imagined. During lookahead, the board scanning routine scans the real world array as described earlier, but it also consults the latest asserted end-state in order to correctly locate imaginary pieces. If many changes are imagined, there is the danger that old ones will be lost as working memory fills up, in which case board scanning errors will occur.

Progressive Deepening. The search path followed by humans during lookahead is best characterized by the term "progressive deepening."[8,18] This term refers to the tendency of people to revisit a given lookahead path in order to explore a single new side path or to extend the depth of the original path. The search path of a subject solving a chess problem thus looks like a succession of many straight paths, with only a few cases of true backup to positions one or two moves back in the path.[18] If the model has to back up, it may do so either by going back to the real board and beginning another lookahead, or by undoing the latest hypothetical move (only the most recently asserted proposition is accessible, and the sequence of moves cannot be deduced from the end-states).

Old propositions are recognized by a pattern matching primitive if they are still within working memory when re-asserted by the model. A simple "termination" flag on old propositions allows the model to investigate other move candidates and thus avoid following the exact same path each time. Propositions become flagged as soon as their end-state imaginary board positions evaluate to some criterion level. Upon re-initiating lookahead, the model re-explores part of the most recent path, branching off when it tries to re-assert a proposition which has been flagged as terminal.

Plans. Beginning with the current real board position, each asserted proposition is connected by *a* "forward" link to the asserted proposition which follows it. One-move backup results in multiple links extending forward from the target of the backup, but a return to the starting position causes a brand new sequence to become linked. Thus, if the sequence of imaginary moves A,B,C,D is followed bv a return to the original position and then by the sequence A,B,C,E, the two structures symbolized by

$$_{A}\text{thenB3thwizCthen}_{>0}$$

and

$$A^{,\text{then}}\text{-*}E^{\text{'then}}\text{>}C^{\text{'then}}\text{>}E$$

will be constructed. A' and A will be complex structures representing two instances of the verb "move" with identical arguments. The other letters have a similar connotation. These linked sequences are the model's representation of plans. The restriction to forward links may be regarded as a result of the way events are perceived in the real world (e.g., it is virtually impossible to whistle a tune backwards).

Means-ends Analysis. The lookahead mechanisms described above were presented mainly in the context of a forward-search analysis of a board position. Not all move selections are arrived at in this way. One or more of the dimensions under the strategy node may suggest a goal state rather than an actual move (e.g., "try to capture a piece"). Since the definition of moves such as "capture" includes the notion of a transition from a peginning-state to an end-state, the model regards the beginning-state in such definitions as a precondition which must be attained before the recommendation may be carried out. For example, in the game of chess the prior state for "x captures y" is "x bears on y." If the prior state exists on the board, then the specific recommendation leads to the assertion of a new move candidate. If the state does not exist, then that state is treated as a new subgoal, and its preconditions are searched for in a similar manner.

The actual search is constrained in the same way as forward search, and therefore we refer to it as "regressive deepening" search.

Since the concept of "forward" linking refers to the order in which the imaginary propositions are asserted, an actual sequence of moves leading to a goal state arrived at through means-ends analysis will be stored in reverse order (i.e., starting move last). The correct sequence may be rederived at each stage. This rederivation may occur a move at a time during the *course* of the game, *or may* occur *in* lookahead by initiating a forward search from the initial move after it has been discovered by means-ends analysis. In practice we expect the model to be able to begin a forward search, and then initiate a means-ends analysis from some hypothetical position which is embedded within the forward s.earch.

Finding a Reason for the Opponent's Move

The opponent's latest move is examined to see whether it is part of a current plan sequence. If so, then it is classified as an "expected" move. We then assign to that move a "reason" taken from the reason for which the model generated this expected move during its earlieT planning sequence. This reason may not be why the opponent actually made that move. This egocentric oversight may cause the model to miss some important configurations, but we believe that humans suffer from exactly this failing.

If the opponent's move is unexpected, the model checks to see whether the change between the previous board and the current board has been favorable for the *opponent. If* either a material change or a pattern identified by this check has a value which surpasses the satisficing criterion, then the model assumes it has found the proper reason for the opponent's move. As before, this may be wrong if there were multiple outcomes of the move. If no satisfactory reason has yet been found, the model tries to see what the opponent can do on his following move, by looking at the current board and pretending to be its own opponent. The most favorable outcome for the opponent after a "lookahead" evaluation is regarded as being the reason for the questionable move.

Note that while all of these processes lead to a seemingly shortsighted and subjective determination of the reason, the model may become more sophisticated by not stopping these procedures upon the discovery of some single high-valued reason, but rather doing all of them, thus finding multiple reasons.

Learning to Play a Better Game

Playing a game may be viewed as an attempt to modify a current state into some goal state by means of some operator (or a plan). Improving one's skill in playing a game involves the emergence of the ability to identify states for which operators exist, and to find operators which are applicable to already known states. States and operations which transform states can be acquired by means of both experience and instructions from an external source (books, a tutor, etc.).

The Lookback Mechanism

The main mechanism for learning through experience proposed here is that of "lookback." Whenever the model encounters a situation which looks definitely better (or worse) than the situation existing before, it reconstructs the former situation and tries to identify what operations were performed which transformed it into the present state. The model may notice a change in states but be unable to reproduce the way in which the change has occurred, especially when it has taken many moves to produce a change. As with lookahead, the deptn of lookback is limited by working memory and aided by the ability to encode whole segments of moves into one meaningful chunk.

Since winning and losing constitute definite changes in states, at these points the model looks back and tries to identify what has made the situation just preceding the completion of the game so powerful. This situation now becomes a subgoal state. The model also attempts to characterize the last move along the dimensions employed during move evaluation. This characterization serves as a plan for transforming the newly found subgoal into a win. Moves that would prevent the win from occurring can also be considered. Winning (or losing) is not the only case where a definite change in states occurs, but it is the most striking one. Whenever the model notices that an already known state has been achieved, the same evaluation process may take place. The result of a successful lookback is the creation of a new node representing the newly identified subgoal. In addition to a description of the subgoal, this node includes a pointer to the goal, and to the sequence of moves leading from the subgoal to the goal.

Whenever an existing operation is about to be executed, the player has definite expectations about the outcome. *If* that outcome fails to materialize, then the operation can be Te-evaluated and modified, or even discarded.

Lookback can occur whenever one encounters a familiar pattern. This is true regardless of whether that familiar pattern is actually there on the board (i.e., consists of real pieces), or whether it partially or wholly consists of imaginary pieces (i.e., pieces "added" to the board by the lookahead process). The working memory constraints acting upon lookahead will make the detection of an imaginary familiar pattern somewhat less likely than the detection of a real familiar pattern, but once such a pattern is detected the learning process may proceed in the regular way.

The Identification of Patterns and Operations

So far the discussion has implicitly assumed that *players* engaged in lookback know what they are looking for, and that the identification of new patterns and plans occurs after the first occurrence of an opportunity to learn which the player is aware of. That assumption is far from being true. Progress in game playing skill is very slow and often frustrating. Working memory constraints arc one reason for the slow progress. Often players just cannot recall the initial state or the sequence of moves which led to the final state.

An even more important obstacle to fast progress (and one which may also greatly increase working memory load) is the player's inability to distinguish between relevant and irrelevant information which exists on the board. The initial state (which eventually becomes a familiar pattern and a possible *final* state) is not on the board by itself. The board often contains many irrelevant pieces. Even the relevant pieces may have properties which are irrelevant for the identification of a particular subgoal.

This situation is analogous to the one existing in psychological experiments on concept learning, and our model goes about learning patterns in a way similar to that in which human subjects learn concepts. Studies of concept learning[14] indicate that people usually generate a hypothesis about the concept they have to learn on the basis of available instances. They regard this hypothesis as the correct concept until they encounter a disconfirmation of the hypothesis, in which case they try another one.

Whenever a subgoal is achieved, the lookback mechanism reconstructs the preceding situation (the "initial state") unless the subgoal has been the result of the application of a plan (in which case no lookback and re-evaluation will occur). Those features which have

changed serve as a clue as to which features may be relevant to the description of the initial state. This description is the model's current concept for the state preceding that subgoal. When the same subgoal occurs again a new description of its initial state is generated. A generalization routine compares this new description with the current concept for the initial state which precedes this subgoal. The common features found by this comparison form a new generalized concept for that initial state. A similar process is applied to the sequences of moves which have led to the final goal. The states and plans identified at the early stages of the game are greatly modified over short periods of time, but they rapidly stabilize.

Tutoring

The preceding section has described the way in which the model learns to play a better game on the basis of experience only. People can become better players by using external sources of knowledge. Zobrist and Carlson[24] emphasize the importance of advice-taking for a model of game playing. Our model has this capability and can be given advice in a fairly natural way. For example, the pattern *"openfour,"*[1] which is crucial in Gomoku, can be taught to the model by entering the statement:

"OPENFOUR" LOOKS LIKE A SEQUENCE OF A
BLANK SQUARE, A ROW OF FOUR PIECES, AND
A BLANK SQUARE.

"LOOKS LIKE" is a predicate which adds a pointer from the name of the pattern to its definition. "SEQUENCE" builds a structure with its members ordered along one axis, and is also a procedure for identifying the existence of a similar structure on the board. "ROW" is similar to "SEQUENCE," except that it allows permutations of its members along a given axis.

The "meaning" of OPENFOUR for the game of Gomoku may be added by saying:

"OPENFOUR" SUGGESTS FOR GOMOKU THAT YOU
PLACE A PIECE ON ANY UNOCCUPIED MEMBER
OF THE "OPENFOUR."

We are currently implementing procedures for giving advice involving functional definitions and lookahead. For example, we would like to be able to define a PIN in chess as follows:

X PINS Y TO Z
ISWHEN: X SEARS ON Y.
IF Y MOVES, X, WHICH DOES NOT BEAR ON Z,
WILL BEAR ON Z,
THE VALUE OF X CAPTURING Z FOR X'S OWNER
IS GREATER THAN THE VALUE OF ANY MOVE
OF Y FOR Y'S OWNER.

The operator "VALUE" takes into account the value of an exchange plus the value of patterns resulting from a given move (it may have to call lookahead to do this).

Notes and References

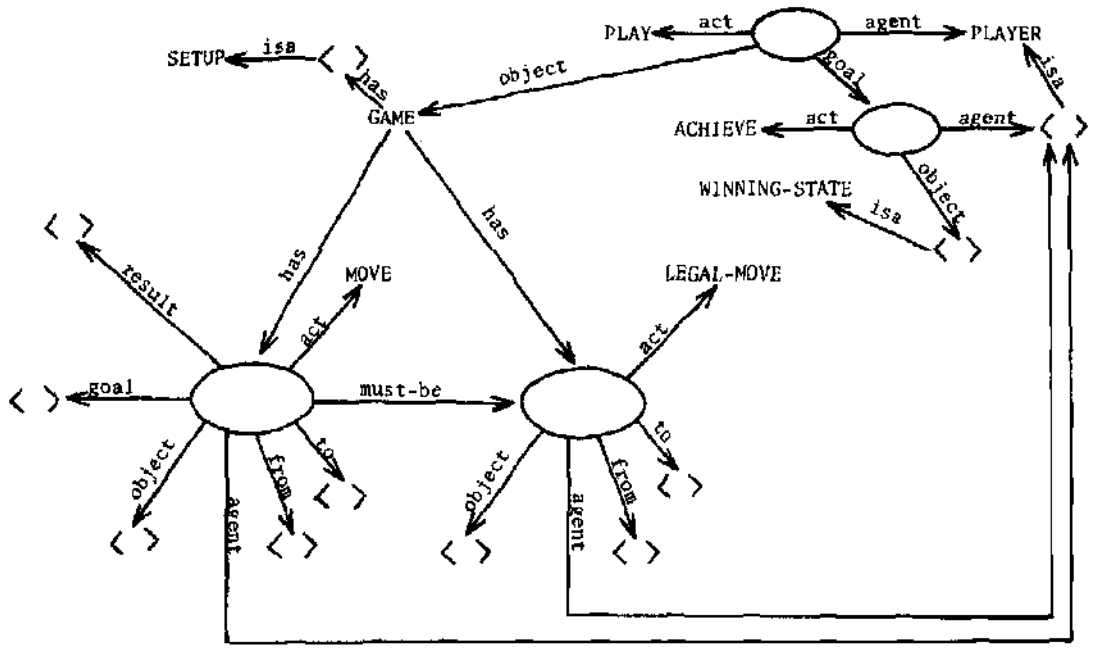
1. The research reported in the present paper is supported by grant GB 3223SX from the National Science Foundation.
2. We wish to thank Donald A. Norman for his helpful comments on an earlier draft of this paper.
3. The assertion that beginners do not play randomly is supported by the following result: An experienced Gomoku player was able to discriminate between randomly generated (legal) moves and moves made by beginners for 41 out of 44 instances (93*) taken from the first Gomoku games played by each of 5 beginners.
4. Gomoku is usually played on a 19x19 board, although other sizes are often used. Two players alternately place *one* of their pieces *on* any vacant square. The first player to get five of his pieces in a row, horizontally, vertically, or diagonally, wins the game.
5. The scan thus finds its nearest horizontal and vertical neighbors before its nearest diagonal neighbors, but finds a neighboring diagonal piece before finding a piece which is 2 squares away from the origin horizontally or diagonally. This conforms nicely with the following datum: 67% of the losses of subjects playing their first 6 games of Gomoku can be accounted for by a diagonal string of 5, even though the number of possible different diagonal strings of 5 is only 37% of the total number of possible strings of 5 on the 9x9 board on which the games took place. The following data indicate that this is because diagonal squares are farther apart than horizontal and vertical squares (by /2~:1) and not because subjects scan horizontally and vertically first: When the board was rotated 45" (to form a diamond), subjects still lost 59% of their games on the "diagonals" (now running horizontally and vertically). In addition, when playing on rectangular boards with sides in the ratio of *2:1* , subjects lost 4 times as many games on the long axis (regardless of whether it was the vertical or the horizontal one) as compared to the number of games lost on the short axis. As expected, most of the games on the rectangular boards were still lost on diagonals (69% of them).
6. If the familiar patterns which direct the top-down scanning routines differ from game to game, then we should be able to construct ambiguous patterns when different types of games are played *on the same* board with the same pieces. We have done this with human game players by using transformed (rotated and reflected) versions of the same configuration of pieces to represent both a Go problem and a Gomoku problem. Subjects were first asked to solve the "Go" problem, and then immediately afterwards to reconstruct the "Go" board configuration from memory. Next, the subjects were asked to solve the "Gomoku" problem, and then to reconstruct the "gomoku" board configuration from memory. In reality, the "Go" problem and "Gomoku" problem were exactly the same relative configuration of pieces. Half the subjects were given the problems in the opposite sequence. A comparison of the reconstructions showed that the subjects remembered different configurations of the pieces for each game. The pieces that were remembered were the ones crucial to the particular game from which they were told that the problem had been taken.

1. The principle of satisficing is that we make the first move which has a value above some criterion level. One node among the ordered set of nodes within each dimension is tagged as being the "satisficing" node for that dimension. Moves which have a value above the satisficing node within any dimension are said to be "above satisficing level."

8. DeGroot, A. D. Thought and Choice in Chess. The Hague: Mouton, 1965.
9. Ein-Dor, P. Elements of a theory of visual information processing. Unpublished doctoral dissertation, Department of Psychology, Carnegie-Mellon University, 1971.
10. Evans, T. G. A grammar-controlled pattern analyzer. In A. J. H. Morell (Ed.), Information Processing 6B, Proceedings of the IFIP .Congress 1968, Vol.2. Amsterdam: North-Holland, 1968, 1592-1S98.
11. Koffka, K. Principles of Gestalt Psychology. New York: Harcourt Brace Jovanovich, 1935.

12. Ledley, R. S. High-speed automatic analysis of biomedical pictures. Science, 1964, 146, 216-223.
13. Leeper, R. W. A study of a neglected portion of the field of learning: The development of sensory organization. Journal of Genetic Psychology, 1935, 46, 41-75.
14. Levine, M. Hypothesis theory and nonlearning despite ideal S-R reinforcement contingencies. Psychological Review, 1971, 78, 130-140.
15. Menninga, L. D. A. syntax-directed approach to pattern recognition and description. Proc. AFIPS FJCC, 1971, 39, 145-151.
16. Miller, G. A. The magical number seven, plus or minus two: Some limits on our capacity for processing information. Psychological Review, 1956, 63, 81-97.
17. Murray, A. M. & Elcock, E. W. Automatic description and recognition of board patterns in Go-Moku. In E. Dale & D. Michie (Eds.), Machine Intelligence 2. New York: American Elsevier, 1968.
18. Newell, A. & Simon, H. A. Human Problem Solving. Englewood Cliffs, N.J.: Prentice-Hall, 1972.
19. Rumelhart, D. E., Lindsay, P. H. § Norman, D. A. A process model for long-term memory. In E. Tulving 5 J. Donaldson (Eds.), Organization of Memory. New York: Academic Press, 1972.
20. Rumelhart, D. E. & Norman, D. A. Active semantic networks as a model of human memory. Proceedings of the Third International Joint Conference on Artificial Intelligence, Stanford, CA, 1973.
21. Ryder, J. L. Heuristic analysis of large trees as generated in the game of Go. Report No. CS-245, Computer Science Department, Stanford University, 1971.
22. Shaw, A. C. Parsing of graph-representable pictures. J.A.C.M., 1970, 453-481.
23. Simon, H. A. & Gilmartin, K. A simulation of memory for chess positions. Complex Information Processing Working Paper #206, Department of Psychology, Carnegie-Mellon University, 1972.
24. Zobrist, A. L. & Carlson, F. R., Jr. An advice-taking chess computer. Scientific American, 1973, 228(6), 92-105.

APPENDIX A: General framework for games.



The empty ovals represent tokens of the "act" to which they point. Empty brackets represent argument slots which are filled in with information either explicitly mentioned or derived from the rules of specific games. The English equivalent of this network is as follows (*note that* statements in brackets are clarifications rather than part of the description itself):

> "Games are played by players. The goal of playing is for one of the players to achieve some winning state. Games have some setup [e.g. board, pieces]. Games have legal moves which are made by players. A legal *move* specifies some object [e.g. 'knight,' 'bishop'], its starting location [i.e. current location], and its final location. [The final location can be defined in terms of its spatial relation to the starting location, and can include any special conditions concerning the path between the two locations (e.g., 'along an unblocked diagonal *to* either an unoccupied square or a Square occupied by an opponent's piece'),] Games have moves which must be legal *moves.* Moves have a goal and a result.

The "must-be" relation between MOVE and LEGAL-MOVE has the following implications: For each node W such that

W-^MOVE and W relation(i)$_{|>x>}$

there exists a node Y such that

Y-J^i LEGAL-MOVE and y relation^

and X is within the range defined by *2,*

The result of each move is a new configuration on the board, plus whatever changes occur in the player's conceptions of the game which are the result of a re-evaluation of the board. The goal of each move is derived from the strategies for generating good moves. If present, such goals are regarded as being the reasons for which particular moves are made.