THE INTELLIGENT NOVICE
Learning to Play Better

Susan L. Epstein
Hunter College of The City University of New York
695 Park Avenue, New York, NY 10021

ABSTRACT

HOYLE is a system under development to explore expertise in the playing of two-person, perfect information games. Rather than a finished expert, HOYLE attempts to capture the evolving expertise of an intelligent novice, an experienced game-player who is learning a new game. HOYLE's thesis is that people have a metatheory for game playing, one that provides them with an initial framework for a new game and enables them to improve their playing skill across time. Under the direction of its metatheory, HOYLE can play any game correctly and then learn to play it better. This paper describes HOYLE's metatheoretical approach to game playing and the hybrid techniques that implement it.

A person who has played other games approaches a newly-introduced game with certain expectations: there will be rules, players, playing pieces, perhaps a board, and the players will take turns. Although he may be new to the game, experience has taught him what there is to be learned about a game and how to learn it. Such an *intelligent novice* can, given the rules, play the new game correctly and, across time, can learn to play it better. He will imitate the expertise of other players and try to apply ideas (e.g., strong openings, forks, feints) that have worked elsewhere. Such incremental learning while doing is not easy. As his own play improves, opponents may challenge him more skillfully. Because other players expect a timely response, he is forced to make decisions and play more quickly than he might choose. Because he may play against more than one player, he may be exposed, confusingly, to many different styles of play. Because his opponents may make errors, he may be modeling his own play on imperfect information.

HOYLE is a system under development to learn to play games well. Rather than purporting to be an expert, HOYLE models the evolving expertise of an intelligent novice. HOYLE differs from most game playing programs in two ways: it is able to play any of a broad class of games correctly, i.e., according to the rules, and it improves its performance through a variety of learning techniques. HOYLE's evolution of expertise is the subject of this paper.

GAMES AND HOW TO PLAY THEM PERFECTLY

HOYLE's domain requires some preliminary definitions. Informally, a game consists of some *material* (e.g., a board, playing pieces), a roster of one or more participants, and a set of rules. The rules describe how the participants are to take turns affecting the position and/or status of the material (e.g., removing pieces from the board or changing their location) until some rule terminates the process. When the process terminates, the rules declare which participant has won or that the process was a draw. If all information about the game is disclosed and equally available to all the players (e.g., no closed hands, no uncertain outcomes as with dice), it is said to be a *perfect information* game. For the games considered here, there are exactly two participants: *Player* (the one who moves first) and *Opponent*. Player and Opponent are the *roles* in a two-person game.

Any game may be represented as a finite, directed graph (the *game graph*) in which a node (*state*) both identifies the participant whose turn it is (the *mover*) and describes a possible arrangement of the material. The participant who is not the mover in a state is referred to here as the *non-mover*. An edge in the graph that goes from one (*originating*) state to another (*resultant*) state represents a *move*, the changes that occur when the mover in the originating state behaves (*takes a turn*) in any one manner permitted by the rules. Together, the material, participants, and game graph are called the *elements* of a game.

There are three kinds of nodes in the game graph: starts, finishes, and intermediate states. A *start* is an initial state of the material before any participant has taken a turn, with Player as mover. A *finish* is a node with no out edges, and is labeled either *win* (Player wins and Opponent loses), *loss* (Player loses and Opponent wins), or *draw* (no winner or loser). Each game has one or more starts and one or more finishes. A node that is neither a start nor a finish is called an *intermediate state*.

In the game graph, a *path* in a game is a sequence $n_1e_1n_2e_2....n_ke_kn_{k+1}$ of nodes $n_1, n_2, \ldots, n_k, n_{k+1}$ and edges $e_1, e_2, \ldots, e_k$, such that e_i is an edge in the graph from n_i to n_{i+1}, for $i = 1, 2, \ldots, k$. If there is a path $n_1e_1n_2e_2\ldots n_ke_kn_{k+1}$ in a game, node n_{k+1} is said to be at depth *k* or *k-ply* from node n_1. If there is any path in the graph from a node back to itself, the game is said to contain a *cycle*. A move from a given node is said to be *invertible* if and only if it is the first edge in some path to a (not necessarily distinct) node where the mover's position is identical to the current one; otherwise it is *uninvertible*. A path in a game from a start to a finish represents one complete experience of the game, and is called a *contest*. During a contest, Player tries to arrive at a win and Opponent tries to arrive at a loss. Since from any state there are usually many legal moves, the challenge in play is for the mover to select the best move, i.e., one that will maximize the mover's opportunity to arrive at a desired state while minimizing the non-mover's opportunity to do so.

It is possible, of course, to represent a two-person, perfect information game as a game graph and play perfectly by backing up the data from an exhaustive search of all possible

sequences of moves (Nilsson, 1980). Even for simple games, however, the game graph is quite large (tic-tac-toe, for example, has more than 5000 reachable nodes), and for difficult games the game graph is computationally intractable. (It has been estimated, for example, that there are 10^{20} nodes in the game graph for checkers, and 10^{43} in that for chess.) Although a perfect player, one with extensive memory and great speed, could play perfect tic-tac-toe through exhaustive search of the game graph from the node representing the current state of the contest, more difficult games will not succumb to this approach.

Another perfect approach to game playing is to consider the theoretical mathematical properties of a two-person, perfect information game and exploit them. Recent mathematical research (Berlekamp et al., 1982), however, indicates that success in many of these games is determined by difficult, and often intractable, mathematical calculations involving abstract concepts called nimbers. Not enough is yet known about computing with nimbers to program for them. Thus neither omniscience nor theoretical computation is a viable option in game playing. Yet some computers, and some people, manage to play some games very well.

GAMES AND HOW TO PLAY THEM WELL

AI research has traditionally approached game playing as a behavior to be modeled and evaluated in competition against people. Several terms require definition here. A *theory* for a game is one or more statements about the properties of and relations among its elements. For example, a possible theory statement for tic-tac-toe is "the center square is the key position." A *heuristic* for a game is an explicit directive to the mover for selecting a move. For example, "if the center square is vacant, move there." A heuristic is an operationalized version of a theory statement; there may be more than one way to construct such an operationalization. Finally, a *strategy* is a set of heuristics for a specific game with a control method for choosing among them. A strategy is thus a procedure that, given any state, returns a move for the mover. To *learn to play a game well* is to produce and refine a strategy for it, constructing, testing, and refining theories.

A participant with a good strategy for a game is an *expert* at it. An expert should be able to take the role of either Player or Opponent and maximize that role within the game graph. This does not mean that the expert should always win, or even never lose; many popular games (e.g., tic-tac-toe) result in a draw when played by experts. Rather, an expert should win when possible, failing that tie, and lose as rarely as the nature of the game permits. (In addition, in games, such as Othello, where there is a final score for each participant, an expert should win by a large margin, and lose by a small one.) This definition of expertise deliberately ignores the skill level of any other participant, and instead defines the strength of a strategy with respect to the game itself. *True expertise* is perfectly backed up knowledge from the entire game graph.

The AI approach to game playing has thus far been to build a program that plays only a single game, and plays it very well. Most of these programs play checkers (Samuel, 1963, 1967), Othello (Rosenbloom, 1982; Lee & Mahajan, 1988), backgammon (Berliner, 1980) or chess (Berliner & Ebeling, 1989; Ebeling, 1986; Anantharaman et al., 1988; Schaeffer, 1988). The Chess 4.5 paradigm (Slate & Atkin, 1977), on which many of them are based, advocates extensive forward search from the node in the game graph representing the current state of the game to a set of intermediate nodes, called *tip nodes*. These tip nodes are then

heuristically evaluated and their values are backed up (Berliner, 1979; Nilsson, 1980; Pearl, 1982; Rivest, 1987; McAllester, 1988) to estimate the best move from the current state. Iterative deepening, selective extension (Anantharaman et al., 1988), and the null move heuristic (Goetsch & Campbell, 1988) are refinements of this game graph search that have proved effective in chess. Such search assumes that the evaluation function is adequate, that the other participant will make no errors, and that a tip node is entirely reflective of the strength of the position, i.e., that search below the tip, when evaluated and backed up, will not substantially alter the tip's value. Such search may be supported by very large historical knowledge bases on openings, endgames, and other play experience (Schaeffer, 1988; Schultz &DeJong, 1988). Despite the machines' superior speed, accuracy, and retention during search, the outstanding human players at most of these games are still able to defeat the best of these one-game programs.

GAMES AND LEARNING

Although they may begin with well-developed perceptual skills, excellent memories, and fine reasoning ability, expert human game players do not spring forth as champions; they develop over hundreds, even thousands, of practice hours. People *learn* to play well. In contrast, many game playing machines have little ability to improve without programmer intervention. For example, Hitech's chess rating continues to climb because its software is continually improved. As they pore over its most recent experiences, it is the researchers who are learning from Hitech's experience, not the machine.

Recent research has attempted to add learning components to some game playing machines with limited success. Learning a preference between two states (Utgoff & Heitman, 1988) has not yet improved performance, and connectionist techniques (Tesauro & Sejnowski, 1988) to support generalization have not yet simulated expert play. Programs like Parry-Phoenix (Schaeffer, 1988) and GINA (Schultz & DeJong, 1988), maintain extensive historical knowledge-bases and supplement them with their own experience, learning by rote.

Only the first of the major game playing artifacts, Samuel's checker player, has learned to play very well. From treatises on how to succeed at checkers, Samuel preselected and hand-coded a list of features, measurements defined on a checkers state to evaluate its merit. Instead of a fixed evaluation function, his program repeatedly calculated weights for a linear sum of these features, and used that formula to distinguish among tip nodes during heuristic search. Over a series of contests, the program learned to play extremely well. The program could gradually decide, by letting a weight move toward zero, that some feature was irrelevant. It could not, however, identify new features that might be significant, generalize its knowledge, or play any game other than checkers.

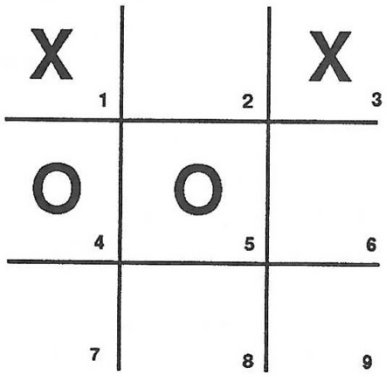
HOW HOYLE PLAYS

HOYLE begins with the ability to play games correctly. Like an experienced game player, HOYLE has predefined factual expectations about what games are and how they are played. This knowledge is represented in HOYLE's game frame and its game playing algorithm. The *game frame* represents all HOYLE's known declarative knowledge about a specific game. Initially, the game frame describes only the game's rules and material, like the inside of the box top on a commercial game. The *game playing algorithm* is a uniform procedure that takes any game frame as input and is able to play, and moderate, a contest in that game.

The game playing algorithm asks who goes first, sets up the initial state of the board, provides a running commentary on whose turn it is, accepts and verifies the legality of all moves, and announces the end of the contest and its results. The game playing algorithm is the same for every contest in every game; it represents all of HOYLE's procedural knowledge about how all the games in its domain are played. With valid data in the pertinent game frame, HOYLE plays any game correctly.

Rather than a single node evaluation method, HOYLE has a set of procedures called *Advisors*. Each Advisor represents some narrow, but quite rational, view of the move selection problem. When an Advisor evaluates a possible move, the Advisor either produces a *comment* on the move (recommending it or advising against it) or offers no opinion. At any state, an Advisor may make any number of comments, each with a strength from 0 (adamant opposition) to 10 (insistent support). When it is HOYLE's turn to move in a contest, it computes the legal moves it could make from the current state and offers them to its Advisors.

Given a game state and a list of legal moves, HOYLE's Advisors evaluate intermediate nodes and generate comments. Consider, for example, a tic-tac-toe game, where Player marks squares with X's and Opponent uses O's. In Figure 1, with Player as mover, one



A State in a Tic-tac-toe Contest
Figure 1

comment might be "move in square 6," to prevent a win by Opponent. This comment could cite as support the threat of the O's in squares 4 and 5, and would have maximum strength, since a win by a reasonably observant Opponent across the second row on the next turn is virtually certain. A second comment, of a similar nature, might be to "move in square 8," since Opponent threatens to win in the second column after two turns. Such a comment must be weaker than the first, because it projects a longer contest. A third comment might be "move in square 2," because Player will win immediately.

Each Advisor epitomizes a different perspective on game playing. The Advisor Panic, for example, looks to see whether the non-mover has a sure win on his next move. In Figure 1, it is Panic who insists that a move in square 6 is the only way to save the day. Victory is another Advisor; it looks for a sure win for itself on the current move. In Figure 1, it is Victory who insists that a move in square 2 is the best choice. With access to certain limited libraries, discussed in the next section, various Advisors

may warn against previously unsuccessful lines of play, and recommend those that were successful in the past. Advisors may also recommend simple plans and defend against primitive models of the opposing player. There are even comments recommending psychological gambits. At the moment there are 17 Advisors, in various stages of implementation. Their descriptive names are Victory, Panic, Sadder, Wiser, Don't-Lose, Enough-Rope, Shortsight, Pitchfork, Open, Candide, Worried, Knot, Sneak, Fewest, Anthropomorph, Spotter, and Wild.

Each Advisor is deliberately defined to take a fairly narrow view of the move selection problem, and is permitted only limited resources before it offers its recommendations. As in the example of Figure 1, the Advisors rarely agree. The *strategy frame* describes HOYLE's theories about how to win at a particular game, including how to mediate among disagreeing Advisors (Epstein, 1989b). To determine its next move, HOYLE applies the control information in its strategy frame for the game to the current contest and the Advisors' recommendations.

HOYLE, however, models an experienced game player. It may not always know whose advice to take at any moment, but it does know that all Advisors are not created equal, i.e., that in game playing some of them are always more important than others. When it is HOYLE's turn to move, it consults its Advisors in tiers, and continues on to the next tier only when unable to make a clearly-substantiated decision from the information at hand. This *hierarchical consultation system* is an important part of the metatheory; it encapsulates general knowledge about game playing that does not have to be relearned for each game. The tiers capture the following commonsense rules :

- If you can move to a win immediately, do it.
- If you cannot win now and are about to lose at the next turn, block the move that threatens you.
- If your opponent has a losing move, try to leave it for him.
- The more contests of the same game you play, the more you should rely on your previous experience.
- In long contests (longer, say, than the number of positions on the board) look for a safe cycle.
- When prospect for winning are dim, take an offensive risk.

WHAT HOYLE LEARNS

HOYLE learns from a kind of spiral curriculum: as the program becomes more proficient at a game against an expert, the individual contests usually force the opponent to marshal more skill. Subsequent contests then provide HOYLE with more advanced and detailed information. It is important to note that HOYLE does not require thousands, or even hundreds, of practice games from which to learn. HOYLE's theories about a game evolve after each contest, during a human-like post-mortem. No game is ever assumed to be learned perfectly; HOYLE is always willing to play another contest. HOYLE employs a variety of learning paradigms to learn to play better.

Selective Memory

Some of the Advisors consult limited, rather than exhaustive, libraries constructed during the post-mortem after a contest is played. HOYLE has a library of previous contests, a library

of openings, a library of human expert moves, a library of significant states, and a knowledge base about each individual opponent. After a contest, HOYLE may extract and cache a play-by-play description of the contest itself, the *opening* (the first 10% of the moves), any *significant states* (states certain to lead to a win or a loss), and data about its competition against each specific participant.

Don't-Lose and Wiser are examples of Advisors that reference these libraries. Don't-Lose comments, with strength 0, any move from the current state whose result is cached as a certain loss, i.e., Don't-Lose recalls previous failure due to such a move and insists that HOYLE try something else. Wiser comments, with strength 10, any move from the current state whose result is cached as a certain win, and lists the next move in the recalled successful endgame. Wiser attributes previous success to the move immediately following the current one, and demands that HOYLE replicate it.

The identification of significant states occurs during the post-mortem after a contest. Consider, for example, the description in Figure 2 of a contest that Player wins on the 25th move,

Mover	State before Move	Move
Player	S-1	M-1
.	.	.
.	.	.
.	.	.
Player	S-21	M-21
Opponent	S-22	M-22
Player	S-23	M-23
Opponent	S-24	M-24
Player	S-25	M-25
Opponent	S-26	none

Calculating Significant States
Figure 2

from state S-25 to state S-26. The second-to-last state, S-25, is cached as "win in 1" with the final move, M-25. If it ever encounters S-25 again, Wiser will recall that state and strongly recommend the winning move M-25 without any consideration of alternatives. The third-to-last state, S-24, is cached as "lose in 2" if and only if, under a limited resource allotment, exhaustive forward search from it is complete and fails to find a better alternative for the mover at that point. If the third-to-last state is so cached, then the fourth-to-last state, S-23, is cached as "win in 3" with the winning M-23 move. On the other hand, if Opponent is the winner in state S-26, HOYLE caches S-25 as "lose in 1" with move M-25, and may, based on a search from S-25 similar to the one described, cache S-24 as "win in 2" and S-23 as "lose in 3" with move M-23. Then, if it ever encounters S-25 again, Don't-Lose will recall the state and strongly recommend against the losing move M-25. In either case, if S-24 and S-23 are cached, HOYLE's post-mortem could continue to retrace the contest through S-22 and then S-21. Theoretically, this cycle of caching and searching might ultimately reach back in the

game graph as far as the first move. Such search is costly, however, and HOYLE, with bounded rationality, devotes only a finite amount of resources to it. This resource allocation is discussed in the next section.

An opening is cached if Player is a (presumably expert) human, if the opening has never been encountered before, and if Player does not lose (i.e., wins or ties) with that opening. (Because HOYLE currently is unable to attribute blame for the loss of a contest to any specific decision or move, HOYLE cannot determine where a participant went wrong. Thus, although the opening in a lost contest may have been correct, even strong, HOYLE will not retain it.) Across time, HOYLE calculates and stores in the strategy frame a *role winner*, the role in which a person wins most often at a specific game. A contest is cached if it is the origin of at least one significant state, if it contributes a new opening, or if the expected role winner lost. If, at the end of an untied contest, there is not yet an expected role winner, the contest is cached for examination after the role winner has been identified.

HOYLE also has the ability to simulate learning by observation, i.e., it accepts as input and caches "book games." These contests between two human experts are presumably ladened in both roles with expert wisdom and offer (unspecified) lessons to be learned. Some HOYLE Advisors accord such contests particular respect. If a human expert wins any contest, every move he made, along with his identity, is cached in the library of human expertise, for reference by the Advisors.

Selective Search

HOYLE allocates limited resources to its Advisors and to its post-mortem analysis. As its knowledge about a particular game graph expands, however, those resources can be used to expand upon what HOYLE already knows. Take, for example, the contest of Figure 2, and assume that allocated resources were exhausted after S-23 was identified as significant and cached. If some subsequent contest reproduces the last six states of the game, HOYLE will apply its post-mortem resources to an analysis of states S-22 and S-21, and perhaps even their predecessors. As HOYLE plays more contests in a given game, its knowledge about the game graph is gradually incremented.

Selective search of the game graph can prevent serious defensive errors during play. One of HOYLE's Advisors, Panic, uses the null move heuristic (as Goetsch and Campbell have done for chess) in a 2-ply lookahead. Panic tests whether, if the non-mover were the mover with the identical board configuration, he could win the contest in a single move. If so, Panic recommends all legal moves that would make such winning moves impossible. (This is how Panic recommended a move in square 6 of Figure 1.) Panic effectively skips one node level, putting itself in the non-mover's place.

Another selective defensive search technique is found in Pitchfork, an extremely successful Advisor on games with uninvertible moves. Informally, a fork is two or more overlapping plans to win or to achieve an advantage. Pitchfork constructs a bipartite graph to model all possible ways that the non-mover could win from the current state. In theory, Pitchfork knows all forks and could guard against them to produce an optimal defense. In practice, such protection consumes far too many resources, because it requires the detection of induced isomorphs of all forks of a given size in these graphs. Instead, Pitchfork tracks which forks

each of its human opponents has ever used, i.e., learns what each individual's tactics are likely to be. During play, Pitchfork always guards against the 7 simplest forks. If HOYLE recalls its human opponent, HOYLE also watches for any more complex forks the other participant has been known to use. If a player uses an unanticipated fork, HOYLE revises its expectations and will check for that fork in future contests against the same player. Because the more complex forks are difficult for people to recognize and execute, this psychological modeling is quite effective.

Strategy Learning

When HOYLE learns a game, HOYLE is really learning a strategy for the game, an effective way to mediate among its often disagreeing Advisors. As HOYLE plays contests in a given game, it learns which Advisors are relevant to the game, which are usually most helpful, and how to balance advice from one against advice from another. Much as Samuel's program learned how to use board features to predict success in play, HOYLE learns how to value advice.

HOYLE currently has two models for applying the strengths of its Advisors' comments: equal representation and strength-based representation. A comment's *vote* is defined to be +1 if its strength is greater than 5, -1 if its strength is less than 5, and 0 otherwise. *Equal representation* values all comments only by their intent, regardless of their strength. Under equal representation a move is valued as the sum of the votes on comments about it. Under *strength-based representation*, a move is valued as the sum of the strengths of the comments about it. HOYLE can also learn weights for either paradigm. For a given game, slots in the strategy frame record whether strength-based or equal representation has proved most successful, and any current weights. At this writing, HOYLE is using unweighted, strength-based representation on the games in its testbed.

HOYLE acquires and refines a strategy for a particular game from its experience in playing that game. Even before HOYLE plays its first contest in a new game, when the strategy frame slots are empty, the slot values become a focus of attention simply by their presence, i.e., HOYLE's metatheory identifies significant factors to be learned in playing any game. Slots indicate whether or not it is an advantage to go first, what the best opening moves are, whether it is possible to win as Player and as Opponent, any territory that is most important to control, and so on. Thus, like a person, HOYLE is aware of strategic possibilities before any contest in the new game ever begins.

Even after only a few contests at a particular game, HOYLE is able to extract valuable knowledge and generalize it to fill the slots. One of HOYLE's Advisors, Open, inductively learns good openings from observing how people open. Another Advisor, Anthropomorph, regularly suggests what it has seen a winning person do in the current state. Yet another Advisor, Spotter, computes the board positions most often controlled by the winner at the end of a contest and recommends moves that secure them.

For now, HOYLE learns in an unstructured environment, i.e., it learns in simple contests, making inductive assumptions from what it experiences. Eventually, however, the program is expected to develop and test new strategies on its own. HOYLE will explore original game-specific strategies under the guidance of its Theoretician, its Experimenter, and its Critic. The Theoretician will postulate likely theories about a game, the Experimenter will oper-

ationalize each theory and explore the goodness of the resultant strategy, and the Critic will revise the theories in light of the experimental results. This aspect is similar in spirit to LEX (Mitchell, 1983) and to (Falkenhainer, 1988).

WORK IN PROGRESS

HOYLE can now play a broad spectrum of culturally diverse games correctly under the direction of its metatheory. The games in the current testbed are tic-tac-toe, lose tic-tac-toe, two versions of three-dimensional tic-tac-toe, tsoro yematatu (Zaslavsky, 1982), pong hau k'i, and achi (Bell, 1969). They were chosen because they span a variety of cultures, and therefore probably capture some aspects of game playing that people find particularly intriguing. Their game graphs certainly do not have the complexity of chess or Go, but they do offer the challenge of cycles and stage transitions, and one of the games (three-dimensional tic-tac-toe on four 4-by-4 boards) has a game graph of billions of nodes.

From its very first contest, HOYLE plays tic-tac-toe well in either role. Against an (inexpert) opponent who does not open in the center, HOYLE learns to play expertly after it tries at most two alternatives to those openings. Lose tic-tac-toe is identical to tic-tac-toe, except that the objective is to avoid getting three markers in a row, column, or diagonal. Colbum (1972) has solved this game mathematically, i.e., clearly delineated and proved a strategy for both roles. Against this strategy, HOYLE plays flawlessly as either Player or Opponent after any sequence of contests that include HOYLE as Opponent against a human expert. (The winning opening is counterintuitive and HOYLE must learn it by observation of a human Player.) In the other games HOYLE rarely loses after two contests, and regularly defeats most human opponents. In particular, HOYLE wins every contest of three-dimensional tic-tac-toe on three 3-by-3 boards when it takes the role of Player, and defends expertly against what should be certain loss when it takes the role of Opponent. At three-dimensional tic-tac-toe on four 4-by-4 boards, HOYLE has not lost a game in several months. Other than the time required to post explanations for its decisions, HOYLE moves virtually instantaneously. (If resources are not limited, however, HOYLE may spendas long as three minutes searching for forks during its first move or two in three-dimensional tic-tac-toe on four 4-by-4 boards.) The Advisors, with selective memory and very little forward search, have proved surprisingly powerful.

Current efforts are focused on the implementation and relative significance of the Advisors. Since Pitchfork now dominates its tier and wins without forward search into the game tree in many games (Epstein, 1989a), the testbed must be expanded. From play HOYLE is expected to learn and incrementally refine a reliable method of mediating among and applying its Advisors' recommendations to the particular game in question. At the moment, HOYLE learns a single strategy for a particular game; plans exist to have it learn strategies for use against particular players, to have it learn strategies that take advice differently in different stages of a contest (e.g., openings, endgames), to have it draw analogies between games, and to have it learn its own metatheory.

HOYLE plays under the control of the game playing algorithm and the appropriate game frame. Move selection is based upon the Advisors' opinions and determined with the guidance of the relevant strategy frame. The game playing algorithm, much of the game frame

and the hierarchical consultation system are precoded; the rest of this knowledge is to be learned. HOYLE will continue to be developed as the games in its knowledge base become more difficult, and the power of its Advisors will continue to be evaluated from the prowess the program displays and the speed with which it learns. In the meantime, HOYLE's incremental learning has thus far provided a remarkably robust alternative to traditional, computationally expensive solution methods.

ACKNOWLEDGMENTS

This work was supported in part by NSF DCR-8514395 and PSC-CUNY 668287 and 666397.

REFERENCES

Anantharaman, T., Campbell, M. and Hsu, F. 1988. Singular Extensions: Adding Selectivity to Brute Force Searching. In *Proceedings on Computer Game Playing*, 8-13. *AAAI 1988 Spring Symposium Series.*

Bell, R. C. 1969. *Board and Table Games from Many Civilizations.* London: Oxford University Press.

Berlekamp, E. R., Conway, J. H. and Guy, R. K. 1982. *Winning Ways for Your Mathematical Plays.* 2 vols. London: Academic Press.

Berliner, H. J. 1979. The B*Tree Search Algorithm: A Best-First Proof Procedure. *Artificial Intelligence* 12 (1): 23-40.

Berliner, H. J. 1980. Backgammon Computer Program Beats World Champion. *Artificial Intelligence* 14 (2): 205-220.

Berliner, H. J. and Ebeling, C. 1989. Pattern Knowledge and Search: The SUPREM Architecture. *Artificial Intelligence* 38(2): 161-198.

Cohen, D. I. A. 1972. The Solution of a Simple Game. *Mathematics Magazine* 45 (4): 213-216.

Ebeling, C. 1986. All the Right Moves: A VLSI Architecture for Chess. Ph.D. diss., Department of Computer Science, Carnegie Mellon University.

Epstein, S. L. 1989a. Deep Forks in Strategic Maps. Technical Report, TRS-89-04, Hunter College of the City University of New York.

Epstein, S. L. 1989b. Mediation among Advisors. In *Proceedings on AI and Limited Rationality,* 35-39. *AAAI 1989 Spring Symposium Series.*

Falkenhainer, B. and Rajamoney, S. 1988. The Interdependencies of Theory Formation, Revision, and Experimentation. In *Proceedings of the Fifth International Conference on Machine Learning,* 353-366.

Goetsch, G. and Campbell, M. 1988. Experiments with the Null Move Heuristic in Chess. In *Proceedings on Computer Game Playing,* 14-18. *AAAI 1988 Spring Symposium Series.*

Lee, K. F. and Mahajan, S. 1988. A Pattern Classification Approach to Evaluation Function Learning. *Artificial Intelligence* 36 (1): 1-26.

McAllester, D. A. 1988. Conspiracy Numbers for Min-Max Search. *Artificial Intelligence* 35 (3): 287-310.

Mitchell, T. M., Utgoff, P. E. and Banerji, R. 1983. Learning by Experimentation: Acquiring and Refining Problem-Solving Heuristics. In *Machine Learning: An Artificial Intelligence Approach*, ed. R. S. Michalski, J. G. Carbonell and T. M. Mitchell. Palo Alto: Tioga Publishing.

Nilsson, N. J. 1980. *Principles of Artificial Intelligence.* Palo Alto: Tioga Publishing.

Palay, A. J. 1982. The B* Tree Search Algorithm - New Results. *Artificial Intelligence* 19 (2): 145-164.

Rivest, R. L. 1987. Game Tree Searching by Min/Max Approximation. *Artificial Intelligence* 34 (1): 77-96.

Rosenbloom, P. S. 1982. A World-Championship Level Othello Program. *Artificial Intelligence* 19 (3): 279-320.

Samuel, A. L. 1963. Some Studies in Machine Learning Using the Game of Checkers. In *Computers and Thought*, ed. E. A. Feigenbaum and J. Feldman. New York: McGraw-Hill.

Samuel, A. L. 1967. Some Studies in Machine Learning Using the Game of Checkers. II - Recent Progress. *IBM Journal of Research and Development* 11 (6): 601-617.

Schaeffer, J. 1988. Learning from (Other's) Experience. In *Proceedings on Computer Game Playing*, 51-53. *AAAI 1988 Spring Symposium Series.*

Schultz, A. C. and DeJong, K. A. 1988. Using Experience-Based Learning in Game Playing. In *Proceedings of the Fifth International Conference on Machine Learning*, 284-290.

Slate, D. J. and Atkin, L. R. 1977. CHESS 4.5 - The Northwestern University Chess Program. In *Chess Skill in Man and Machine*, ed. P. Frey. New York: Springer.

Tesauro, G. and Sejnowski, T. J. 1988. A Parallel Network that Learns to Play Backgammon: Recent Results. In *Proceedings on Computer Game Playing*, 41-45. *AAAI 1988 Spring Symposium Series.*

Utgoff, P. E. and Heitman, P. S. 1988. Learning and Generalizing Move Selection Preferences. In *Proceedings on Computer Game Playing*, 36-40. *AAAI 1988 Spring Symposium Series.*

Zaslavsky, C. 1982. *Tic Tac Toe and Other Three-in-a-Row Games, from Ancient Egypt to the Modern Computer.* New York: Crowell.