Pattern-Based Learning and Spatially-Oriented Concept Formation in a Multi-Agent, Decision-Making Expert

Susan L. Epstein
Department of Computer Science
Hunter College and The Graduate School of The City University of New York
695 Park Avenue, New York, NY 10021
212-772-5210
epstein@roz.hunter.cuny.edu

Jack Gelfand
Department of Psychology, Green Hall
Princeton University, Princeton, NJ 08544 USA
609-258-2930
jjg@phoenix.princeton.edu

Joanna Lesniak *
Department of Computer Science
Hunter College
695 Park Avenue, New York, NY 10021

* Present address: Department of Computer Science, Rutgers University, New Brunswick, NJ 08903, jlesniak@paul.rutgers.edu.

Keywords: concept formation, game playing, pattern recognition, machine learning, spatially-oriented reasoning, decision making, Hoyle, morris games.

To appear in Computational Intelligence, 1996

Pattern-Based Learning and Spatially-Oriented Concept Formation in a Multi-Agent, Decision-Making Expert

Abstract

As they gain expertise in problem solving, people increasingly rely on patterns and spatially-oriented reasoning. This paper describes an associative visual pattern classifier and the automated acquisition of new, spatially-oriented reasoning agents that simulate such behavior. They are incorporated into a multi-agent game-learning program whose architecture robustly combines agents with conflicting perspectives. When tested on three games, the visual pattern classifier learns meaningful patterns, and the pattern-based, spatially-oriented agents generalized from these patterns are generally correct. The accuracy of the contribution of each of the newly created agents to the decision-making process is measured against an expert opponent, and a perceptron-like algorithm is used to learn game-specific weights for these agents. Much of the knowledge encapsulated by the new agents was previously inexpressible in the program's representation and in some cases is not readily deducible from the rules.

1. Pattern learning in game playing

The thesis of this work is that an associative visual memory and spatially-oriented reasoning agents can make significant contributions to programs that do high-level reasoning. We tested this approach on two-person, perfect information, finite-board games using a multi-agent game playing program as a platform for our experiments. From a pattern-oriented language based on small collections of markers, patterns that were persistently associated with wins, losses, and draws were stored in a pattern cache. These patterns were used to guide the decision-making of a pattern-based agent in the multi-agent game playing program. We then generalized over sets of patterns in the pattern cache to create new pattern-based concepts and then proceduralized those generalizations as new heuristic agents in the program. Finally, we validated both the agent based on the individual patterns and those agents proceduralized from the concepts formed from those patterns. This was done with an algorithm that compares their advice with the decisions of an expert opponent and weights their contribution to the decision-making process accordingly.

This approach uses two kinds of pattern-oriented learning for game playing: the association of particular patterns with successful or unsuccessful play, and the construction of spatially-oriented heuristics from those patterns. Figure 1(a), where the empty locations are blanks and # denotes "don't care," is an example of the association of an individual pattern with a particular outcome of the game. It links a particular pattern from tic-tac-toe with success for X. In any symmetric orientation and whatever the # squares contain, a human expert associates such a configuration with a win for X.

Along with particular patterns, game playing experts use more general but equally salient heuristics as spatially-oriented "rules of thumb." Figure 1(b) is an example of such a pattern-based concept. It is the spatially-oriented heuristic "reflect O's move through the center," proved to be optimal play for X in the game of lose tic-tac-toe (Cohen, 1972). Advice from experts on how to analyze and play games is repeatedly couched in the language of such spatially-oriented patterns. Chess and checkers are discussed in terms of controlling the center of the board, while control of the edges is crucial in Othello (Fine, 1989; Gelfer, 1991; Lee and Mahajan, 1990; Samuel, 1963). Concepts such as shape and thickness are fundamental to the game of Go (Hideo, 1992; Iwamoto, 1976; Yoshio, 1991). As people improve their expertise in game playing, they increasingly employ spatially-oriented heuristics, and treat them as compiled knowledge, integrated but no longer reasoned about.

To learn pattern associations, programs use a feature language and inductive learning algorithms that operate on game states described in that language. There are several chess-playing programs that capitalize upon patterns. MACH integrates chunks, identified by human master chess players from grandmaster games, into the evaluation function of a chess program called Phoenix (George and Schaeffer, 1991). Morph learns threat-and-defense digraphs for chess with temporal difference learning and genetic algorithms, and generalizes them, where appropriate, as features for its evaluation function (Levinson and Snyder, 1991). HiTech uses elaborate hand-coded concepts as part of its evaluation function (Berliner, 1992). CHUNKER solves king and pawn endings with equivalence class sets of pawns (Campbell, 1988). Flann's program learns a decision tree of predicate calculus descriptions to classify lost-in-n-ply chess positions (Flann, 1992).

Applying learned patterns to game playing, however, has proved somewhat problematic. There are usually a great many of them and matching is non-trivial. T2 and Zenith, for example, learned predicate calculus expressions for tic-tac-toe and Othello, respectively (Fawcett and Utgoff, 1991; Yee, et al., 1990). On one run T2 learned 45 tic-tac-toe concepts with 52 exception

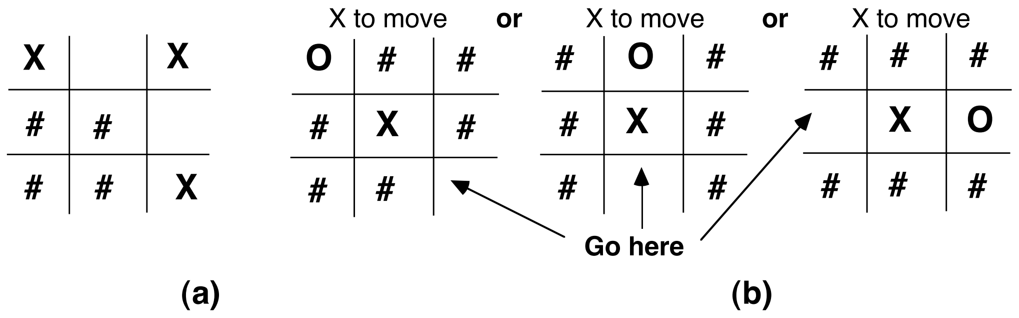

Figure 1: (a) A tic-tac-toe pattern that X associates with winning. # denotes "don't care." (b) "Reflect through the center," a spatially-oriented heuristic for lose tic-tac-toe.

clauses after 800 contests, a great many for so simple a game. To make its memory tractable, Morph limits the number of concepts retained at any moment to 5000. Phoenix made better moves with MACH, but used the pattern-based elements in the evaluation function only at the root and not at subsequent nodes because it was so time-consuming to search and evaluate the pattern-based terms.

In the work described here, learned pattern knowledge is used to construct higher-order, spatially-based reasoning agents. Programs that learn concepts from game-playing experience have in the past been hampered by a predicate calculus representation that lacks incisiveness, and by exhaustive explanation of inconsistencies for positions that may have no consequence in the strategic play of the game (Fawcett and Utgoff, 1991; Yee, et al., 1990). The process we describe, in contrast, is able to deal with inconsistencies robustly while it focuses attention on those situations containing important visual patterns. Pell deduces higher-order reasoning agents from the rules of a game, agents which may or may not be pattern-based(Pell, 1993). In contrast, our work induces them from playing experience .

The major contribution of this work is a system that limits the cost of creating strategically meaningful higher order concepts by remembering a limited group of patterns associated with wins, losses and draws as a basis for new spatial concepts. We used thresholding, aging and consistency mechanisms to filter the cache formation process. The process of creating a restricted set of patterns might leave some information out of the learning process. We found, however, that the concepts formed were those that are deemed important in published analyses of those games that we tested.

The long-range objective of this work is to create a heuristically-based decision maker that learns rapidly enough to participate in intelligent behavior while it is still acquiring knowledge. Within a hierarchical multi-agent system the presence of other, more general problem solving agents prevents incorrect actions, especially during early experience while learning. In this paper we show that the system functioned within this environment. We found that the validation process for newly-created agents performed properly, and that the system worked smoothly as knowledge was being refined during the learning process. We believe that this process of creating new agents and testing their correctness in a multiple-agent program is unique.

The multi-agent game playing program is detailed in Section 2, the pattern-based learning system in Section 3, and the results of our experiments in Section 4. Section 5 discusses the results and related work is presented in Section 6. For this initial test of the approach we used simple games and made some simplifications in the individual component parts of the program and their operation. We discuss these and methods for scaling the approach to more complex games in Section 7.

2. A game-learning program

There is evidence that humans integrate a variety of strategies to accomplish problem solving (Biswas, et al., 1995; Crowley and Siegler, 1993; Ratterman and Epstein, 1995). For example, the primate visual system has pathways for form, place, motion, and color (DeYoe and Van Essen, 1988; Ungerleider and Mishkin, 1982). Information from these streams is combined to form a perception of the visible world (Kandel, 1991). In addition, it has been found that different parts of the brain are activated when decisions are being made about different strategic aspects of chess (Nichelli, et al., 1994).

The mechanisms we describe below simulate these features. *Hoyle* is a program that learns to play two-person, perfect information, finite-board games. It is based on a learning and problem-solving architecture for skills called FORR, which employs multiple concurrent decision-making agents (Epstein, 1994a). Hoyle, as modified here, includes a separate stream for pattern learning.

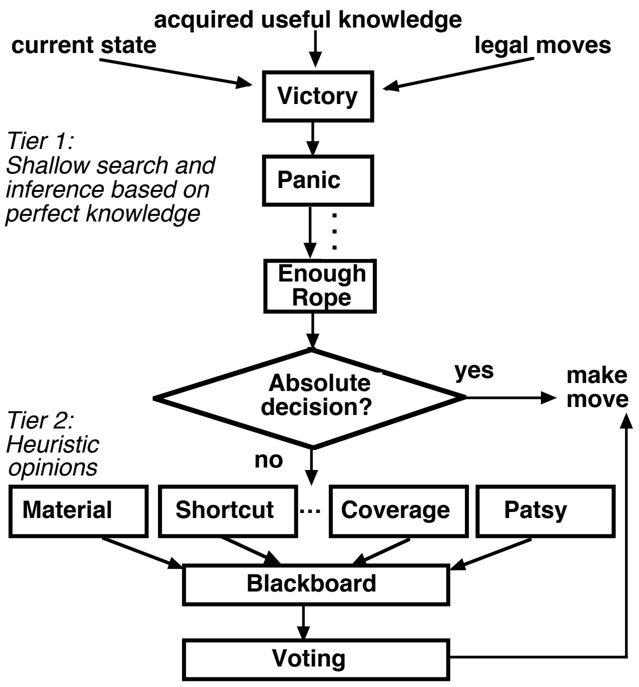

Figure 2: How Hoyle makes decisions.

Hoyle learns to play in competition against a hand-crafted, external expert program for each specific new game. As shown in the schematic of Figure 2, whenever it is Hoyle's turn to move, a hierarchy of resource-limited procedures called *Advisors* is provided with the current game state, the legal moves, and any useful knowledge (described below) already acquired about the game.

Table 1: Hoyle's Advisors for game playing.

Name	**Tier**	**Description**	**Useful knowledge**	**Learning Strategy**
Wiser	1	Makes the correct move if the current state is remembered as a certain win.	Significant states	Deduction
Sadder	1	Resigns if the current state is remembered as a certain loss.	Significant states	Deduction
Victory	1	Makes the winning move from the current state if there is one.	None	—
Don't Lose	1	Eliminates any move that results in an immediate loss.	Significant states	Deduction
Panic	1	Blocks a winning move the non-mover would have if it were his turn now.	Significant states	Deduction
Shortsight	1	Advises for or against moves based on a two-ply lookahead.	Significant states	Deduction
Enough Rope	1	Avoids blocking a losing move the non-mover would have if it were his turn now.	None	—
Anthropomorp h	2	Moves as a winning or drawing non-Hoyle expert did.	Expert moves	Abduction
Candide	2	Formulates and advances naive offensive plans.	None	—
Challenge	2	Moves to maximize its number of winning lines or minimize the non-mover's.	None	—
Coverage	2	Maximizes the mover's markers' influence on predrawn game board lines or minimizes the non-mover's.	None	—
Cyber	2	Moves as a winning or drawing Hoyle did.	Important contests	Abduction
Greedy	2	Moves to advance more than one winning line.	None	—
Leery	2	Avoids moves to a state from which a loss occurred, but where limited search proved no certain failure.	Play failure and proof failure	Abduction
Material	2	Moves to increase the number of its pieces or decrease those of the non-mover.	None	—
Freedom	2	Moves to maximize the number of its subsequent immediate moves or minimize those of the non-mover.	None	—
Not Again	2	Avoids moving as a losing Hoyle did.	Important contests	Abduction
Open	2	Recommends previously-observed expert openings.	Opening database	Induction
Patsy	2	Recreates visual patterns credited for positive outcomes in play; avoids those blamed for negative ones.	Visual patterns	Associative pattern classifier
Pitchfork	2	Advances offensive forks or destroys defensive ones.	Forks	EBL
Shortcut	2	Bisects the shortest paths between pairs of markers of the same contestant on predrawn lines.	None	—
Vulnerable	2	Reduces the non-mover's capture moves on two-ply lookahead.	None	—
Worried	2	Observes and destroys naive offensive plans of the non-mover.	None	—

As detailed in Table 1, Hoyle has 23 heuristic, game-independent Advisors in two tiers. (The newest, Patsy, is discussed extensively in Section 3.4.1.) The first tier sequentially attempts to compute a decision based upon correct knowledge, shallow search, and simple inference, such as Victory's "make a move that wins the contest immediately." If no single decision is forthcoming, then the second tier collectively makes many less reliable recommendations based upon narrow viewpoints, such as Material's "maximize the number of your markers and minimize the number of your opponent's." Although this may appear to be quite a few Advisors, they do a large job with remarkable efficiency. Hoyle learns to play one game with about 9 million states expertly, for example, during exposure to about .012% of the search space, and explicitly retains data on only about .006% of the states in the game graph. Based on the Advisors' recommendations, a simple arithmetic vote selects a move that is forwarded to the game-playing algorithm for execution. Hoyle plays without ever searching more than two ply (one move for each contestant) ahead in the game tree.

Hoyle learns from its experience to make better decisions based on acquired useful knowledge. *Useful knowledge* is expected to be relevant to future play and is probably correct in the full context of the game tree. Examples of useful knowledge include recommended openings and states from which a win is always achievable with perfect play on both sides. Each item of useful knowledge is associated with at least one learning algorithm. The learning methods for useful knowledge vary. Table 1 includes Hoyle's useful knowledge and its associated learning strategies. The learning algorithms are highly selective about what they retain; they may generalize and they may choose to discard previously acquired knowledge. An Advisor outputs its recommendations in the form of *comments*. A comment is of the form "(Advisor, action, strength)" where *strength* is an integer from 0 to 10 that measures the intensity and direction of opinion. Further details on Hoyle are available in (Epstein, 1992).

3. Learning to use and apply patterns

The crux of this paper is the addition to Hoyle of pattern learning and its application in new, game-dependent third-tier Advisors. With only 22 Advisors, the program had already learned to play 18 different games extremely well. The implementation of pattern learning and its application were inspired by repeated laboratory experiences with people, in the context of many different games. College students spoke about, reacted to, and relied upon familiar, sometimes symmetrically transposed, patterns while learning (Ratterman and Epstein, 1995). Later, they relied heavily upon these patterns as a kind of compiled expertise.

In this work, visually-perceived regularities are represented as *patterns*, small geometric arrangements of marker types (e.g., black, X) and unoccupied positions (*blanks*) in a particular geographical location. A new useful knowledge object, the *associative pattern store,* provides a heuristically-organized database that links patterns with contest *outcome* (win, loss, or draw). The associative pattern store includes a set of templates, a waiting list, a pattern cache, generated concepts, and uninformative patterns.

Figure 3 provides an overview of the refinement of the pattern matcher and the development of pattern-based Advisors from the game-specific associative pattern store. There are four processes detailed here: **associate**, **generalize**, **proceduralize**, and **validate**. Once patterns are identified, they are associated on the waiting list with winning, losing, or drawing. Patterns that persist over time and are identified with a single consistent outcome move from the waiting list to the pattern cache. Patterns in the cache are proceduralized via an associative pattern classifier, a new, game-independent Advisor called *Patsy*. Additional Advisors are created on periodic sweeps through the pattern cache to generalize sets of patterns into concepts. Concepts are proceduralized as individual, game-specific Advisors that are then validated during subsequent learning. In addition, the pattern matcher improves as Hoyle learns to constrain pattern generation by excluding uninformative patterns and templates.

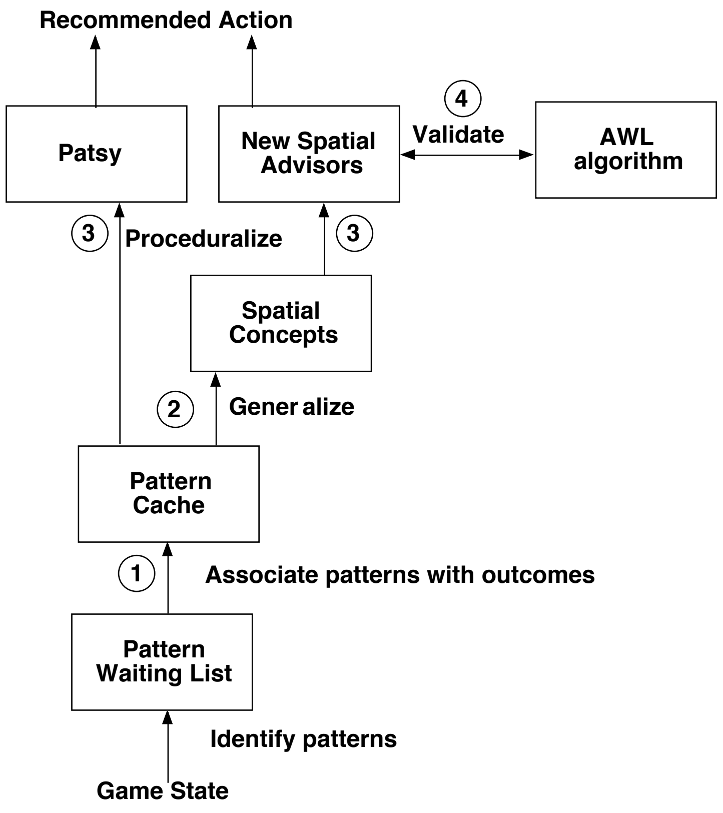

Figure 3: A schematic diagram of the pattern-oriented learning system, including refinement of the pattern matcher and development of pattern-based Advisors from the game-specific associative pattern store.

Much of this paper references morris games, played on boards like those in Figure 4. A morris game has two contestants, black and white, each with an equal number of markers. A morris contest has two stages: a *placing stage*, where initially the board is empty, and the contestants alternate placing one of their markers on any empty position, and a *sliding stage,* where a turn consists of sliding one's marker along any line drawn on the game board to an immediately adjacent empty position. A marker may not jump over another marker or be lifted from the board during a slide. Three markers of the same color on immediately adjacent positions on a line form a *mill*. Each time a contestant constructs a mill, she *captures* (removes) one of the other contestant's markers that is not in a mill. Only if the other contestant's markers are all in mills, does she capture one from a mill. The first contestant reduced to two markers, or unable to move, loses. Morris games offer substantial challenges: five men's morris has about nine million states in its search space, nine men's about 7.7 billion (Gasser, In press).

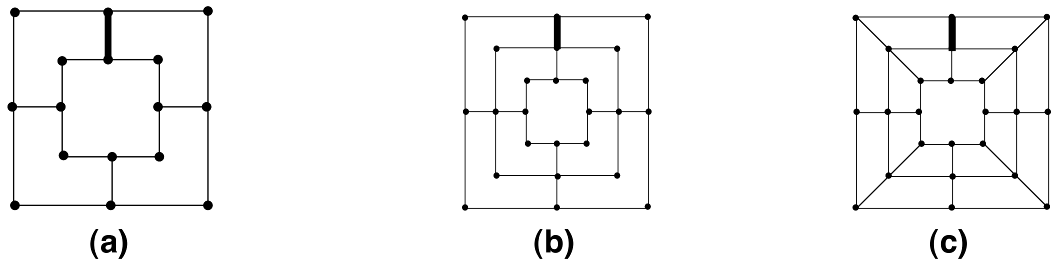

Figure 4: Some morris boards with (a) 16 positions for five or six men's morris, and 24 positions for (b) nine men's morris and for (c) 11 men's morris. Dots appear at the positions where markers may be placed. The darkened line segments represent the metric unit used in the Bounded Pattern Language described in Section 3.1.

Throughout the implementation, patterns were distinguished by game and by stage, e.g., there were separate caches for placing stage moves and sliding stage moves in each game. (Stages constitute virtually separate games with their own rules; hence the separate segments of the cache for each of them.) The waiting list, pattern cache, concepts, and uninformative patterns were represented as hash tables of unlimited size.

3.1 Constructing a Pattern Based Language

For purposes of this investigation of simple games, we begin with a set of prespecified, game-independent, perceptually-biased templates. For larger, more complex games we would expect to use a more a sophisticated pattern classifier. One must choose between a complete language capable of expressing any pattern along with induction rules that generate many, overly complex instances, and an explicitly-biased language that filters potential concepts at the risk of not expressing everything. We have chosen the latter for two reasons: because people are known to have visual perceptual biases for small geometric groupings (Hendee, 1993; McBurney and Collings, 1977) and the implementation of pattern learning and its application were inspired by repeated laboratory experiences with people, in the context of many different games (Ratterman and Epstein, 1995). While T2's bias emphasizes moves later in a contest, and Morph's bias emphasizes threat and defense, our bias is toward visual patterns. We do not claim that our bias is better, but we present evidence here that with it we can learn certain kinds of concepts inexpressible, or unlikely to be learned, with the other biases.

BPL (Bounded Pattern Language) is a set of expressions, each of which describes a shape delineated by some number of required points. There are five valid expressions in BPL: straight lines, squares, diagonals, L's, and triangles without right angles. Definitions of these expressions appear in Table 2. A BPL expression has ?'s in its required positions and #'s in its irrelevant ("don't care") ones. Each of them, except for diagonals, is constructed only from predrawn straight line segments on the game board. For example, a BPL square has ?'s in each of its four required corners, and has sides that are already drawn as lines on the game board. The size and location of the square are deliberately unspecified.

Table 2: The valid BPL expressions

Template type	Required positions	Optional positions	Predrawn lines	Field of view delimits
Line	2 endpoints and midpoint, if any	yes	only	endpoints
Square	4 vertices	no	only	diagonal
Diagonal	2 endpoints	yes	not permitted	endpoints
L	3 vertices (forming a right angle)	yes	only	hypotenuse
Triangle	3 vertices (forming no right angle)	yes	only	longest side

To scale such shapes to a game board, a metric is necessary. Let the version of the game board drawn for output, as in Figure 4, be called the *picture*. The *metric unit* for the game board is the smallest Euclidean distance in the picture between any two positions where markers can be placed. An example of the board-specific metric unit is darkened for each game board in Figure 4.

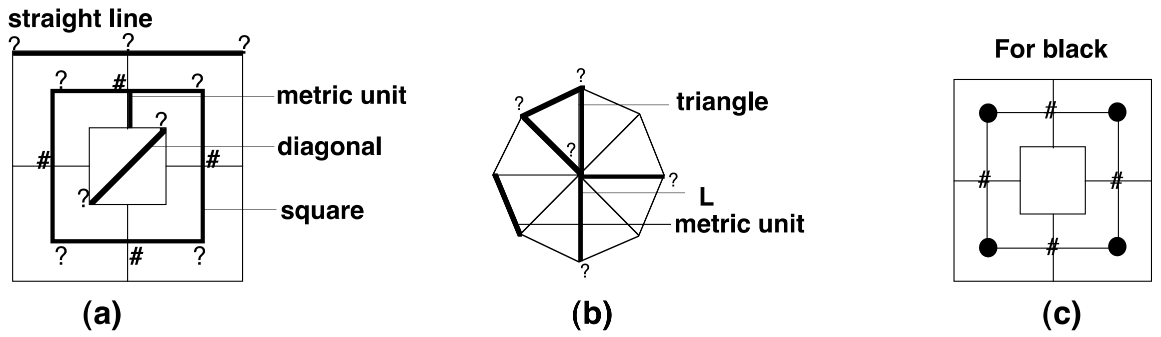

Figure 5: Templates for (a) nine men's morris and (b) shisima (Zaslavsky 1982). (c) A pattern in nine men's morris. # denotes "don't care."

A board-specific *template* is an elaboration on a BPL description that specifies both the location of the shape on the board and the size of the shape in metric units. Thus a template highlights certain specific positions on the board with ?'s. (Any other positions it is superimposed upon are labeled with the don't care symbol #.) The BPL square, for example, can give rise to three different square templates for the board in Figure 4(b). The template that is concentric with the board and whose sides measure 4 metric units is show in Figure 5(a), with four #'s on its irrelevant positions.

Board-specific templates are a function of a game board's topology, and are calculated only once, when Hoyle first encounters a new game. Rather than capture every possible board-specific template, a single *field of view parameter* limits a template's maximum breadth as an integer multiple of the game board's metric unit. Several other board-specific templates are shown in Figures 5(a) and 5(b). Field of view must be set to at least 2 for the triangle and the L shown in Figure 5(b) to be detected, and at least 3 for the diagonal, and 6 for the square and the straight line in Figure 5(a). Templates are unique up to symmetry, so that, for example, there is only a single triangle template in Figure 5(b). As a result, there should be relatively few relevant templates, even for a game board with a fair number of marker positions. With field of view 4, Hoyle generates only 18 templates for the nine men's morris board in Figure 5(a).

A *pattern* specifies a mover and instantiates the ?'s in a board-specific template with blanks and the markers of one or both contestants. For example, Figure 5(c) is an example of a pattern formed from the nine men's morris square template in Figure 5(a). It specifies that black is the mover and already occupies all the corners of the size-4 square. Each template is matched to a game board using any of the eight symmetries of the two-dimensional plane (the identity mapping, three rotations, and four reflections).

3.2 Associate: learn responses for patterns

The first of the four processes in Figure 3 is the association of responses with the identified patterns. The *pattern classifier* is an algorithm that processes the patterns identified by the board-specific templates of the preceding section. It associates each pattern with a *response*, defined as a sequence of three integers that count number of contests won by the first contestant, number won by the second contestant, and number drawn in which this pattern appeared.

Most states match one or more templates and therefore make multiple contributions to the associative pattern store. For the purposes of this study, we limited pattern learning to at most four distinct, crucial states, the two from each stage resulting from each contestant's last non-forced move. (Such a state offers the mover more than one legal move and has no immediate block to a win the other contestant would have if it were her turn instead.) Our premise is that a state that achieves a goal or subgoal for either player is one that contains key patterns. Such states result from significant attention, and, to the extent that the opponent is an expert whose move may have been pattern-based, they can offer a particular benefit. Of course, the non-forced move states, while not necessarily endgame states, are closely related to them. One could extend our procedures to subgoals to provide patterns less oriented to states at the end of each stage. The algorithm matches the selected states against the board-specific templates, adjusting for all 8 symmetries.

Table 3: Algorithm to learn to associate patterns with outcomes.

```
Learn-pattern (pattern, outcome, contest-number)
Case 1: pattern is in uninformative-patterns
  exit
Case 2: pattern is in waiting-list
  Update pattern's data, adjusting for frequency and consistency of its
  association with outcome
  If pattern appears on waiting list more often than threshold times and has
  exactly one non-zero response
      then transfer pattern from waiting list to the pattern cache
Case 3: pattern is in the cache
  Update pattern's data, adjusting for frequency and consistency of its
  association with outcome
Otherwise: insert pattern on waiting-list
```

The patterns detected at the end of each contest are processed one at a time by the pattern-learning algorithm sketched in Table 3. When it is first identified, a pattern not among the uninformative pattern section of the store (see Section 3.3) is relegated to the *waiting list,* labeled with its mover, its contest outcome, and the number of the contest in which it has appeared. For example, the first time that the pattern shown in Figure 5(c) is encountered in the placing stage of nine men's morris and associated with a win, the pattern would enter the placing stage waiting list for that game with response 1-0-0 (number of times the first contestant has won, lost, and drawn with this pattern, respectively), and the contest number in which it appeared. When a pattern already on the waiting list is re-encountered, its response values are updated and the new contest number is recorded. Thus if the same pattern is encountered again in the placing stage and as a win, the pattern's response is updated to 2-0-0 with that contest number.

We also age response values by multiplying them by an *aging* parameter at the end of every contest but before any new patterns are processed. A pattern association which does not reappear will eventually be forgotten as its response value eventually reaches effective zero. Thus, one-time experiences are not retained. There are, of course, one-time experiences worthy of retention. These, we argue, are cases, not patterns, and are more appropriately learned by other components of the system.

3.2.1 Managing inconsistency

Because a novice cannot always capitalize appropriately on its own good patterns or exploit the opposition's poor ones, the learner may initially make incorrect associations, only to find them contradicted later when it plays better (Epstein, 1994c). Our learning algorithm therefore employs a *confidence parameter* to revalue responses in the face of disagreeing evidence. If a pattern is already in a cache or waiting list, but now arrives with a different non-zero response, the previous responses are multiplied by 1- confidence. For example, if a pattern is recorded with response 12-0-3 and that pattern (with the same stage and mover) is now processed after a contest in which the first contestant lost when the confidence parameter is 0.4, the new response would be 7.2, 1, 1.8. At any point in time, confidence is the same for all patterns.

Initially the confidence parameter *c* is zero, but it dynamically reflects how well the program has played across time. After *k* contests generating some sequence of wins, losses, and draws, the program's raw confidence in its ability to play well was measured by

[1] $$c_{raw} = \sum_{i=1}^{k} \frac{\text{outcome}_i}{k - i + 1} \text{ where outcome}_i = \begin{cases} +2 \text{ for a win in contest i} \\ -2 \text{ for a loss in contest i} \\ +1 \text{ for a draw in contest i} \end{cases}$$

If there were α wins, β losses, and γ draws in the sequence, maximum confidence c_{max} would result from c_{raw} computed on the sequence $L_1L_2 \ldots L_\beta D_1D_2 \ldots D_\gamma W_1W_2 \ldots W_\alpha$ and minimum confidence c_{min} would result from the sequence $W_1W_2 \ldots W_\alpha D_1D_2 \ldots D_\gamma L_1L_2 \ldots L_\beta$. Thus normalized confidence in [0, 1] is

[2] $$c = \frac{c_{raw} - c_{min}}{c_{max} - c_{min}}$$

3.2.2 Managing consistency

When a pattern has consistently appeared with the same association, it shifts from the waiting list to the pattern cache. There are two criteria for this shift: the pattern must have appeared a certain number of times (the *threshold* parameter) and exactly two of the pattern's response values must effectively be zero. Admittedly, if two patterns appear the same number of times with the same associations, the one detected earlier will age its inconsistencies sooner and could thereby migrate to the cache sooner. Because pattern learning is presumed to be an ongoing process, we do not consider this unreasonable.

Response values can eventually reach effective zero because they too are multiplied by an *aging* parameter in the same way as the waiting list. Aging for the cache is slower than for the waiting list because it is important to retain salient patterns that are only seen occasionally. If a pattern has not occurred for a long time, however, its value diminishes; this is why consistently-associated patterns must be promoted from the cache to the status of concept. Concepts are not aged; corrections to their relevance are based only on new evidence as described in Section 3.5.

We note that there is refinement of the contents of both the waiting list and pattern cache in terms of a threshold to get into the waiting list, aging in both the waiting list and pattern cache, and the management of both consistent and inconsistent entries. Although we did not perform a quantitative study of this cache refinement process, we did find that without it performance degraded. These processes are ongoing and constantly refine the storage of important patterns with experience.

3.3 Generalize: formulate concepts from the associative pattern store

Cached patterns are a rich source of information about the marker clusters to be seen during a particular game. Some of them ought to be forgotten; others are worthy of elevation to concepts that drive game-dependent Advisors. The identification of both kinds of patterns is done during a periodic sweep of the cache. Currently, the first sweep of the pattern cache to form concepts is after 15 contests, and then the frequency is recomputed as a function of the confidence parameter after each sweep.

Some small patterns are identified in every, or almost every, contest of a particular game, regardless of its outcome. For example, the pattern consisting of an X at either end of the top lose tic-tac-toe row and a blank in the center often occurs when it is O's turn to move. Almost every lose tic-tac-toe contest produces this "X-blank-X in the top row, O to move" pattern, regardless of the contest's outcome. We found such patterns offered no meaningful associations yet were costly to process, and have therefore developed a method that learns to avoid their repeated consideration. A pattern that appears in almost every contest of a particular game, regardless of its outcome, is learned as uninformative. When a pattern is first extracted with a template, it is checked against the uninformative patterns first, to see if it warrants further processing. Furthermore, if every possible instantiation of a template becomes an uninformative pattern, then the template itself is discarded, so that no future pattern observation uses it.

Generalization summarizes a set of detailed experiences into a more useful and efficient representation. Hoyle has two generalization rules to form concepts. Patterns in a cache are said to *agree* when they originate from the same template and pertain to the same stage.

- Given distinct agreeing patterns P1, P2, and P3 with q ?'s that have the same mover and single, non-zero response, and are identical, except that in the *i*th position P1 has a black, P2 a white, and P3 a nil value, construct a new pattern P on the q-1 ?'s other than the *i*th. An example appears in Figure 6(a).

• Given distinct agreeing patterns P1 and P2 such that interchanging the contestants' markers and changing the mover in P1 results in P2 with the opposite single non-zero response, construct a new pattern P with variable place holders α for black and β for white. An example appears in Figure 6(b).

The cache is organized to support fast detection of agreeing patterns.

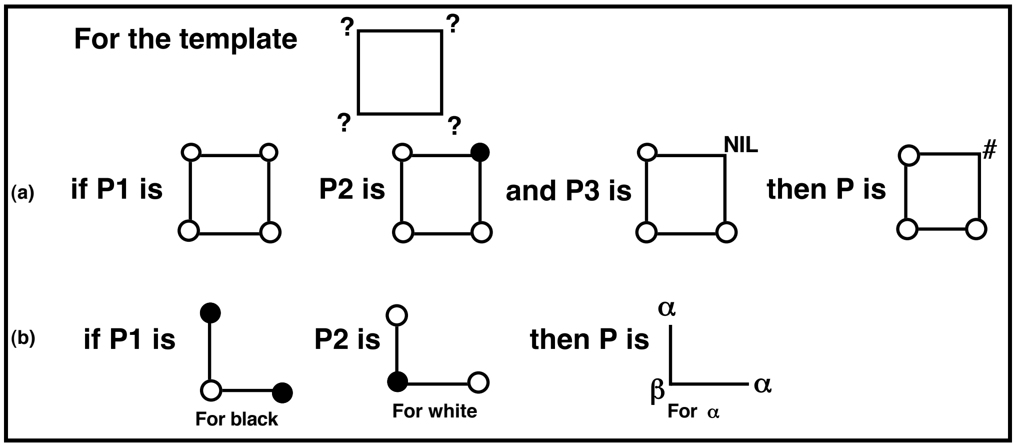

Figure 6: Two generalization rules that are applied to patterns to formulate concepts.

3.4 Proceduralize: convert knowledge into advice

Proceduralization is the transformation of expert knowledge into expert behavior. This is a non-trivial task in AI (Mostow, 1983). When there is much data or it conflicts in its potential application, as with pattern knowledge, interesting challenges arise. Each segment of the associative pattern store therefore relates differently to decision making. Patterns on the waiting list have no impact on decision making at all. Patterns in the cache serve as input to the associative pattern classifier, Patsy. Pattern-based concepts become game-specific Advisors.

3.4.1 Patsy, the game-independent, pattern-based Advisor

The new, game-independent, second-tier Advisor Patsy ranks legal next moves based on the way the states they engender match patterns in the cache. Patsy considers the set of possible next states resulting from the current legal moves. Each next state is compared with the patterns in the appropriate cache. (No new patterns are cached during this process.) Each pattern is assigned a value computed by

$$[3] \qquad \frac{2\left[w_p - \min(w_p, \ell_p, d_p)\right]}{W} - \frac{2\left[l_p - \min(w_p, \ell_p, d_p)\right]}{L} + \frac{\left[w_p - \min(w_p, \ell_p, d_p)\right]}{D}$$

where the response to the pattern p is w_p-l_p-d_p and the total number of won, lost, and drawn contests since the pattern was first seen are W, L, and D, respectively. (If any denominator is 0, that fraction is omitted from the sum.)

The strength of Patsy's comment on each legal next move is a function of the values of the patterns in the state to which it leads. (Matching analyzes the most specific version of a detected pattern.) A move that results in a state all of whose cached patterns are wins for the mover (no draws or losses) is recommended with strength 10. A move that results in a state all of whose cached patterns are losses for the mover (no wins or losses) is recommended with strength 0. Otherwise, each move is scored as the sum of the response values computed by (3) for newly-introduced patterns. Moves with negative pattern scores are recommended with strength 2, 3, or 4, and moves with positive pattern scores are recommended with strength 6, 7, or 8, depending upon their relative ranking. Thus Patsy actively encourages moves that lead to states introducing new patterns associated with a win or a draw, while it discourages moves that lead to states introducing patterns associated with a loss. As the strength of the associations changes with time and experience, Patsy adapts its advice appropriately.

3.4.2 A set of game-dependent, pattern-based Advisors

Like most game-playing programs, Hoyle gets into difficulty in the middlegame. It learns openings by copying them from its opponents. It learns endgame play by selective retrograde analysis, reasoning backward from some of the states experienced during play and storing the correct moves along with the significant states (Epstein, 1992). Frequently, however, the middlegame gets murky. There may be several dozen legal moves, among which the second-tier Advisors see as many as a third as viable alternatives. A traditional game-playing program faces a similar situation when it has searched interesting lines to some depth and its evaluation function detects no strong preference. A new, pattern-based third tier of game-dependent Advisors is designed to resolve middlegame dilemmas.

Each concept is proceduralized as a new, third-tier, game-specific Advisor. If the perfectly-correct, game-independent first-tier Advisors can select a move with their game-specific useful knowledge, they do so and the second tier is never consulted. If the heuristic but generally correct, game-independent second-tier Advisors can agree upon a move with their game-specific useful knowledge, they do so. Otherwise the moves judged equally good by the second tier are forwarded to the newly-created third tier of game-dependent, pattern-based Advisors. For concept C in game G the new G_C-Advisor comments only in game G. The G_C-Advisor advocates any move to a state where a new instance of C is introduced, and opposes any move to a state where an instance of C is eliminated, in a comment whose strength is a function of C. In the formulation of its comment, G_C does not consider the presence or absence of any other pattern concepts.

3.5 Validate: confirm the accuracy of new Advisors

As new, pattern-based Advisors are introduced and Hoyle's skill develops further, some of them may prove irrelevant, self-contradictory, or untrustworthy, despite prior empirical evidence of their validity. Credit/blame assignment in a domain such as this is extremely difficult. At the end of a contest, it is difficult, even for human experts, to pinpoint the move that won or lost. The significant decision may have been early in play, or may have been a set of moves rather than an individual one. Rather than credit or blame a particular move, we have chosen to credit or blame the Advisors that support expert-like behavior.

Consider, for example, a hypothetical game state in which Hoyle has only second-tier comments (Advisor-1, move-1, strength-1) and (Advisor-2, move-2, strength-2). Until now, if strength-1 and strength-2 were equal, the vote would be a tie, and one of the moves would have been chosen at random. If Advisor-2 were more trustworthy in this particular game, however, its comment should have more influence. This approach holds the rationale behind actions accountable, rather than the actions themselves. Irrelevant and self-contradictory Advisors in a particular game should have weight 0, and more trustworthy Advisors should have higher weights than less trustworthy ones. Empirical experience with Hoyle indicates that these weights are problem-class specific, i.e., a new item of useful knowledge to be learned.

With an external model of expertise as its performance criterion, we use a perceptron-like model called *AWL* to learn problem-class-specific weights for the decision-making procedure (Epstein, 1994b). Rather than tally each comment of the same strength as equivalent, AWL learns game-dependent, stage-dependent *weights* for all second-tier and third-tier Advisors, so that, for example, two comments with the same strength would not necessarily be treated equally.

AWL runs at the end of every contest Hoyle plays against an external (human or computer) expert. The algorithm considers, one at a time, only those states in which it was the expert's turn to move and Hoyle's first tier would not have made a decision. For each such state, AWL distinguishes among support and opposition for the expert's recorded move and for other moves. Essentially, Hoyle learns to what extent each of its Advisors simulates expertise, as exemplified by the expert's moves. AWL cumulatively adjusts the weights of second-tier and third-tier Advisors at the end of each contest (whether or not the third tier would actually have voted during play), and uses those weights to make decisions throughout the subsequent contest. The weights are a modification of Littlestone's perceptron-like algorithm (Littlestone, 1988). Updating during play would slow Hoyle down considerably; we massively update the weights at the end of each contest instead.

4. Results

In all of the experiments described here, Hoyle alternately moved first in one contest and second in the next. Such a *trial* continued until Hoyle was said to have *learned to play a game* because it could draw *n* consecutive contests in this environment. Once it met this *behavioral standard,* learning was turned off and the program was tested against a variety of challengers that simulated perfect, expert (10% random move selection, 90% perfect), novice (70% random move selection, 30% perfect), and random contestants. The threshold parameter, effective zero, the aging parameter for the waiting list, and the aging parameter for the cache, all described in Section 3, were 5, .01, .9, and .999 respectively. Without contradiction or further reinforcement, a single outcome will remain on a waiting list that way for 459 contests, and in the cache for 4603 contests. All trials included the AWL algorithm of the preceding section (Epstein, 1994b). Data for each game is averaged across 10 trials, and examples of consistently learned, pattern-based concepts for the three games appear in Figures 7, 8, and 9.

We have used pattern-based learning with Hoyle in tic-tac-toe, lose tic-tac-toe (played exactly like tic-tac-toe except that the first contestant to achieve three of the same playing piece along a row, column, or diagonal *loses*), and five men's morris. Since Hoyle had already learned to play all the games studied here expertly after relatively few contests, these experiments were intended to demonstrate that game-dependent visual patterns exist and persist, despite the non-determinism of the learning experience. We found that the potential computational overhead for concept formation is avoided because very few of the possible patterns ever appear on the waiting list or in the cache. In tic-tac-toe, despite the potentially large number of patterns, after learning there were 58 patterns in the waiting list, 22.2 patterns in the cache, 4.2 uninformative patterns, and 6.4 concepts, all for draws. In lose tic-tac-toe, with the same potential number of patterns, after learning there were 58.8 patterns in the waiting list, 57.2 patterns in the cache, 1.4 uninformative patterns, and 19 concepts, some for draws and others for losses.

Hoyle also learned the same pattern-based concepts on every run of a fixed game. This is particularly significant because the program is non-deterministic, i.e., its playing experience on every run is different. For example, all but the last concept in Figure 7 were learned and preserved on all tic-tac-toe runs; the last was learned four times. There are, as one would expect, slightly varying numeric responses from one run to the next. In lose tic-tac-toe the top four concepts were learned on every run and always appear with the highest weights in the third tier.

In addition, the Advisor Patsy was highly weighted by the AWL validation algorithm, indicating that game-specific pattern-based reasoning performed more like the external expert opponent that most of the other game-independent heuristics in the second tier. After learning tic-tac-toe, Patsy's average rank by weight among the Advisors in the second tier was 3 out of 17; after learning lose tic-tac-toe Patsy's average rank was 6.5 out of 17. AWL assesses Patsy to be a valuable Advisor. The growth in the weight of Patsy and in the weights of the pattern-based Advisors simulates the transition from high-level reasoning to skill learning.

With sufficient experience, Hoyle learns concepts based upon the patterns in the cache, concepts found to be correct by our own analysis or from previously published analyses of those games. The concept in Figure 7a, for example, describes control of the center. Although it appears to be a simple pattern, it is actually a generalization over a set of persistent patterns. The concepts in Figure 7e and 7f advocate blocking an incipient win in the center of a row or on one end.

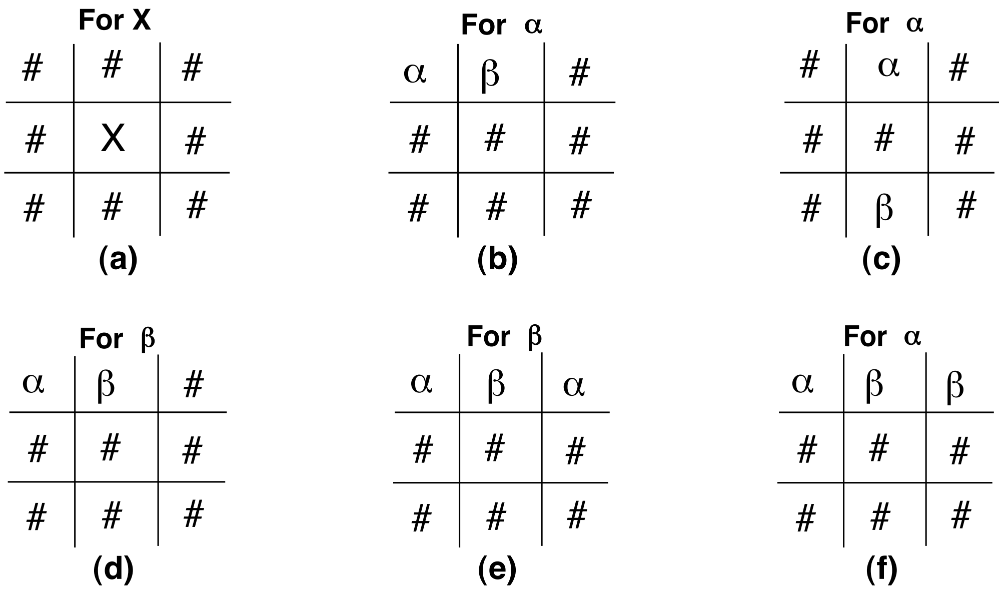

Figure 7: Some learned concepts for tic-tac-toe. α and β denote either X or O (or black or white) consistently; NIL denotes an empty position. Note that the mover for a concept is in the current state, but the pattern is matched for in a subsequent state.

The AWL algorithm functioned properly and was able to unlearn irrelevant or incorrect pattern-based concepts. These were created during the period when Hoyle was learning incorrect pattern associations based upon novice play. The Advisors proceduralized from the incorrect concepts are gradually ignored as their weights decrease below 1. Figures 8c and 8d are examples of incorrect pattern-based concepts that Hoyle learned and then gradually rejected because they consistently disagreed with the moves of an expert opponent. The irrelevant ones are discarded because they fail to comment in any contest after they are created. Figures 8e and 8f are examples of pattern-based concepts that Hoyle learned and then discarded.

Furthermore, important concepts are learned that were previously inexpressible in Hoyle's representation. An example of this appears in lose tic-tac-toe where, to play the role of X perfectly, one must move in the location that is the reflection, through the center, of O’s last move. Such reflection was not previously expressible in Hoyle’s useful knowledge, but is now learned as the concepts in Figure 8a and 8b. (Note that, with symmetry, all reflections are captured.)

New heuristics are learned which were previously obscured by the manner in which the rules were accessed. The program experiences the rules of a game only as a set of “black boxes” that return the current state, the legal moves from it, and whether or not a state results in a win, a loss, or a draw. Consider, for example, what we term here *confinement,* the concept of restricting a five men's morris marker to a corner so that it can no longer slide. (Recall that a morris contestant unable to slide loses.) Confinement, the rightmost concept in Figure 9, is learned by Hoyle on every run. The concept of a mill (three markers of the same color on immediately adjacent positions on a line) was also previously outside the program's knowledge. (Hoyle only knows that certain moves permit it to capture, but not why.) Now on every run of five men's morris, Hoyle learns the first two concepts in Figure 9 as a pair of Advisors that subgoal on mills.

Correct reflection concept

For β

α	#	#
#	#	#
#	#	β

(a)

For β

#	α	#
#	#	#
#	β	#

(b)

Found to be incorrect

For α

NIL	α	#
#	#	#
#	#	#

(c)

For α

α	NIL	#
#	#	#
#	#	#

(d)

Found to be irrelevant

For β

α	α	#
#	#	#
#	#	#

(e)

For β

α	#	α
#	#	#
#	#	#

(f)

Figure 8: Some learned concepts for lose tic-tac-toe. α and β denote either X or O (or black or white) consistently; NIL denotes an empty position. Note that the mover for a concept is in the current state, but the pattern is matched for in a subsequent state.

Hoyle learned tic-tac-toe with a behavioral standard of 10 against an external, expert, game-specific program that played perfectly. It learned lose tic-tac-toe and five men's morris, however, with a behavioral standard of 20 and lesson and practice training (Epstein, 1994c). In this environment (unnecessary for the easier of game tic-tac-toe), the program cycles between *lessons* (a set of two contests against the expert) and *practice* (a set of seven contests against itself). Without lesson and practice training, Hoyle had learned the correct patterns and pattern-based concepts for competition against an expert, but lacked the same knowledge for competition against less expert players. With lesson and practice training, pattern learning continued on every contest, but AWL was applied only to the lessons, so that Hoyle learned to imitate only the expert. The reflection Advisors for lose tic-tac-toe and the mill Advisors for five men's morris have weights that remain among the top few in the third tier during learning with AWL. Although the reflection Advisors tend to emerge only after 80 or so contests, they typically achieve weights higher than 10 of the 17 second-tier Advisors.

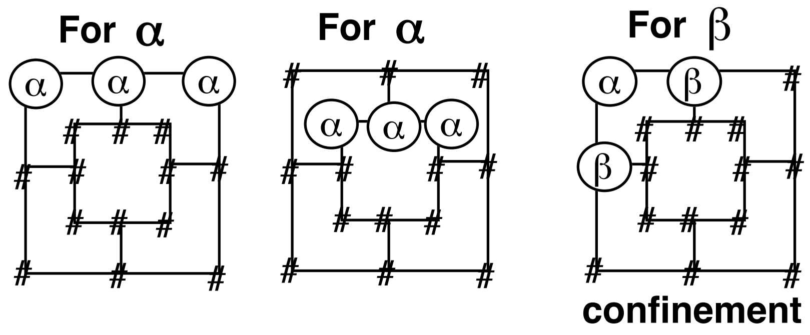

Figure 9: Some learned concepts for five men's morris. α and β denote either X or O (or black or white) consistently; NIL denotes an empty position. Note that the mover for a concept is in the current state, but the pattern is matched for in a subsequent state.

5. Discussion

Our work not only integrates pattern learning with high-level reasoning, it also suggests how the former gradually comes to support and enhance the latter. We do *not* advocate reliance on pattern-learning alone. That would ignore the other higher-level processes quite evident in humans. Indeed, Hoyle learns many other kinds of useful knowledge detailed elsewhere (Epstein, 1992). Pattern learning is, however, an important component in skill development, one that those interested in the simulation of human intelligence or the design of adaptive game-playing programs cannot afford to ignore.

Each of the patterns Hoyle now learns is a generalization over a class of states that occurs with some frequency and contains a simple configuration of spatially-related markers. These patterns occur in the context of a particular stage of the game and are consistently associated with a single outcome. An associative pattern classifier provides learning whose possibly premature guidance is tempered by the higher-level reasoning of the other Advisors. More experienced, concept-based Advisors gradually emerge to emphasize broader generalities, and are expected to advocate expert play to retain their status. Finally, the identification and exclusion of uninformative patterns constrains the pattern generator and thereby focuses the entire process more intelligently.

In any attempt to replicate this learning model, one must allow enough time for patterns to migrate from the waiting list into the cache and to validate the game-dependent Advisors. There is an intricate relationship among the number of contests played, the threshold that keeps patterns on the waiting list, and the aging parameters for the waiting list and the cache. For example, Hoyle learns tic-tac-toe so quickly with a visual threshold of 2 that it has no opportunity to create concepts at all. Although we tested other combinations, the parameter values used here were the most successful.

The choice of these five particular BPL shapes is not central to the theme of this research, but was used as a starting point to illustrate the operation of the system. When students in our laboratory learn these games, they repeatedly cite small geometric arrangements of pieces as salient patterns, much like Simon's chunks (Chase and Simon, 1973). More complex games will require a more complex BPL, for example, one that would include thickness and shape for Go. An important problem, however, is to define and test for chunks in a way that minimizes the potential combinatoric explosion. In a game with n possible board locations and only t types of markers (including blanks), there are t^n possible patterns.

Not all visual patterns, of course, are worth detecting or remembering. If patterns are overly specific (e.g., an exact board description) there will be too many of them. If patterns are overly general (e.g., a marker in a corner) they may provide little reliable information in the context of the game tree. Even when a visual pattern is at the "right" level of specificity, it may not be worth noticing (e.g., the patterns in Fig. 8e and 8f) because nothing important is denoted by its presence. Statistical pattern recognition also requires adequate experience to render it reliable; in an extremely large space that may be impossible. We have employed a variety of devices to limit the number of identified patterns: BPL for game-independent template generation with a visual bias, normalization for symmetry, and the field of view parameter. Each of them is empirically observable in humans and captures certain kinds of regularity detected in their visual system, such as the generalization on symmetry and biases toward learning regular and more compact patterns (Darley, et al., 1981).

6. Related Work

6.1 Expert Game Players, Human and Machine

Psychologists have established, contrary to popular belief, that game-playing experts do not have distinctive mental abilities (like exceptional powers of concentration, enormous memories, or high IQ's), that they do not do extensive forward search into the game tree, and that they do not rely on statistical measures of typicality or concrete visual images during play (Binet, 1894; Charness, 1981; Djakow, et al., 1927; Holding, 1985). What grandmasters do have are perceptual focus of attention, carefully organized knowledge, and procedures to manipulate that knowledge. They summarize some of their knowledge in concepts, both as verbal memories and as chunks.

A grandmaster's recall is better than an ordinary person's on chess positions, but only for chess positions that are meaningful, i.e., ones that would arise during the play of a contest (Chase and Simon, 1973).

In general, empirical evidence indicates that skilled people in many domains have better memories, but only for meaningful patterns, and that, given the same knowledge, an unskilled person remains unskilled, i.e., cannot apply it appropriately. (See, for example, (Allard, et al., 1980; Egan and Schwartz, 1979; Engle and Bukstel, 1978; Shneiderman, 1976).) An expert's memory is organized contextually, so that a single board configuration that could arise in either of two different games is chunked and recalled differently, depending upon the game in which it is perceived (Eisenstadt and Kareev, 1975). There is also some evidence that a limited amount of memory organization may be around prototypes, i.e., that certain situations may be reminiscent of others (Goldin, 1978; Watkins, et al., 1984).

The typical game-playing program, in contrast, plays only one game and does not learn, plan, or retain its experience. Instead it uses fast, deep, sophisticated search, and a feature-based evaluation function that estimates the strength of any possible game state as a real number (Rich and Knight, 1991). It also relies on an opening book of early move sequences favored by human experts. Often this is implemented with special-purpose hardware. The programmer selects static features that human experts hold in high regard. The tradeoff between the number of nodes searched and the time devoted to computation at each node is critical for real-time performance, and is well documented in the game-playing literature (Kierulf, 1989; Lee and Mahajan, 1990). Large-scale memory for endgames has resulted in strong play for checkers and nine-men's morris; statistics on the success of each previously encountered state have fared less well (DeJong and Schultz, 1988; Schaeffer, et al., 1991; Schultz and De Jong, 1988).

Cognitive scientists, however, consider this brute force approach not at all reflective of the processes they observe in skilled experts. Practical problem-solving and decision-making domains offer challenges in scale well beyond most games. Brute-force memory approaches in game playing include full-blown retrograde analysis and statistics on how often an encountered state is associated with a win. The former overlooks context, and the latter can only be helpful when the same states are repeatedly encountered, that is in games where there are few enough "typical" states. And both of these mass-memory approaches overlook the AI objective; the intent of a game-playing program is to make decisions like an expert, not to know or remember exhaustively. Our work moves instead toward Clancey's "dialectic coupling of sensorimotor systems in an ongoing sequence of coordinations" (Clancey, 1993).

6.2 Pattern-oriented Play

In AI, visual cues have previously demonstrated their power as explicit search control directives and as hand-selected terms in an evaluation function (Gelernter, 1963; Samuel, 1963). There is also evidence that perceptually-based schemas for geometry theorem proving support planning at a more abstract level and thereby limit search (Koedinger and Anderson, 1990). Visual imagery, as an alternative representation has been addressed in part by Glasgow's computational imagery, but without a procedural component (Glasgow and Papadias, 1992). "Pure" pattern-oriented programs learn slowly and tend to be unreliable (Boyan, 1992; Michie, 1974; Painter, 1993). Nonetheless, many important game-playing programs have had strong, visually-oriented features in their evaluation functions (Kierulf, 1989; Lee and Mahajan, 1990; Samuel, 1963; Samuel, 1967).

There is more than one kind of pattern that expert programs detect. Programs that learn predicate calculus plans or EBL explanations are slowed by extensive matching time (Fawcett and Utgoff, 1991; Freed, 1991; Morales, 1991; Wilkins, 1980; Yee, et al., 1990). The use of chunks to focus attention and abstract the chess board has met with some limited success (Campbell, 1988; Flann, 1992; George and Schaeffer, 1991). Morph's tactical patterns are based on threat and defense(Levinson and Snyder, 1991). Our work takes a different approach but overlaps somewhat with Morph. Inspection of the 12 published patterns learned by a Morph-like variant playing tic-tac-toe (Levinson, et al., 1992) shows several that are covered by Hoyle Advisors (e.g., Victory, Panic, Open, and Pitchfork). A mill in the morris games is not a Morph-like pattern, however, nor is symmetry through the center (although the components are). There is nothing in the architecture of this system that precludes Morph-like patterns from this framework.

There has been much work on generalization methods for data represented in predicate calculus, in rule-based formulations, and numerically, as well as some work on matrix-oriented visual perception (Anderson, 1986; DeJong, 1986; Dietterich and Michalski, 1983; Glasgow and Papadias, 1992; Kodratoff and Ganascia, 1986; Langley, et al., 1986; Sammut and Banerji, 1986). Future work will develop additional generalization rules based on these methods.

7. Limitations and Future Work

For the initial test of this overall approach we used simple games and made a number of simplifications in the individual components of the system. Patsy, the Advisor based on individual patterns, was placed in Hoyle's second tier. The correct tier assignment for the new Advisors created from pattern-based concepts is another subject of current research. They were placed in the third tier for the experiments described here, to avoid interference with a preexisting second tier that already worked quite well. To improve computational efficiency, however, and to model the transition to automaticity, the pattern-based Advisors should either reside in the second tier or gradually migrate to it. Then, if they competed in parallel with the other second-tier Advisors, the pattern-based Advisors should comment much faster in situations where they are applicable, and thereby supplant the others.

Thus far, the AWL algorithm that learns problem-class-specific weights for decision-making procedures is interactive; a human is asked "permission" before an Advisor is removed because it is irrelevant, self-contradictory, or harmful. We intend to automate this process fully, so that Hoyle will once again be a system without human intervention. We are also testing a variety of loss bounds for use with AWL (Haussler, et al., 1994; Kivinen and Warmuth, 1994).

For the purposes of this study, we limited pattern learning to four states in each contest, two from each stage resulting from each contestant's last non-forced move. These game states near the end of the game, or the end of a game stage in the case of the morris games, proved to capture significant patterns in these shallow games. Our current research includes more sophisticated methods of learning pattern associations. It might be possible to use states preceding subgoals to provide patterns in the middle game. We are also experimenting with temporal difference learning to consider every pattern as it arises in states throughout a contest (Sutton, 1988).

Future work includes more difficult games and other kinds of templates for spatial relations (such as center, edge, perimeter, bounded regions, length, and area), and causally-based pattern generation where one or more patterns that give rise to concepts are combined to create new, larger, somewhat less regular patterns. We intend to experiment with other learning algorithms to determine which is best for our application, and to develop and test a suite of generalization rules and meta-rules to construct concepts from patterns, so that, for example, the two reflection concepts for lose tic-tac-toe become a single one, as do the two mill concepts for five men's morris.

Acknowledgments

We acknowledge helpful discussions with Ron Kinchla, Philip Johnson-Laird and Nick Littlestone, and the detailed comments of an anonymous referee. This work was supported in part by a grant from the James. S. McDonnell Foundation to the Human Information Processing Group at Princeton University, NSF grants #9001936 and #9423085, ONR grant #00014-93-1-0510, and PSC-CUNY grant #665292.

References

Allard, F., Graham, S. and Paarsalu, M. E. 1980. Perception in Sport: Basketball. Journal of Sport Psychology **2**: 14-21.

Anderson, J. R. 1986. Knowledge Compilation: The General Learning Mechanism. *In* Machine Learning: An Artificial Intelligence Approach - Volume II, *Edited by* R. S. Michalski, J. G.

Carbonell and T. M. Mitchell. Tioga Publishing, Palo Alto. 289-310.

Berliner, H. 1992. Pattern Recognition Interacting with Search. Technical Report, Pittsburgh, PA, CMU-CS-92-211, Carnegie Mellon University.

Binet, A. 1894. Psychologie des Grands Calculateurs et Joueurs D'échecs. Hachette, Paris.

Biswas, G., Goldman, S., Fisher, D., Bhuva, B. and Glewwe, G. 1995. Assessing Design Activity in Complex CMOS Circuit Design. *In* Cognitively Diagnostic Assessment, *Edited by* P. Nichols, S. Chipman and R. Brennan. Lawrence Erlbaum, Hillsdale, NJ.

Boyan, J. A. 1992. Modular Neural Networks for Learning Context-Dependent Game Strategies. Master's thesis diss., University of Cambridge, Cambridge.

Campbell, M. S. 1988. Chunking as an Abstraction Mechanism. Ph.D. thesis diss., Carnegie Mellon University, Pittsburgh, PA.

Charness, N. 1981. Search in Chess: Age and Skill Differences. Journal of Experimental Psychology: Human Perception and Performance **7**: 467-476.

Chase, W. G. and Simon, H. A. 1973. The Mind's Eye in Chess. *In* Visual Information Processing, *Edited by* W. G. Chase. Academic Press, New York. 215-281.

Clancey, W. J. 1993. Situated Action: A Neuropsychological Interpretation (Response to Vera and Simon). Cognitive Science **17** (1): 87-116.

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

Crowley, K. and Siegler, R. S. 1993. Flexible Strategy Use in Young Children's Tic-Tac-Toe. Cognitive Science **17** (4): 531-561.

Darley, J., Glucksberg, S. and Kinchla, R. 1981. Chapter 4: Perception. *In* Psychology, Prentice Hall, New York. 118-153.

DeJong, G. 1986. An Approach to Learning from Observation. *In* Machine Learning: An Artificial Intelligence Approach - Volume II, *Edited by* R. S. Michalski, J. G. Carbonell and T. M. Mitchell. Tioga Publishing, Palo Alto. 571-590.

DeJong, K. A. and Schultz, A. C. 1988. Using Experience-Based Learning in Game Playing. In Proceedings of the Fifth International Machine Learning Conference, 284-290. Ed. J. Laird. Morgan Kaufmann, San Mateo.

DeYoe, E. and Van Essen, D. 1988. Concurrent Processing Streams in the Monkey Visual Cortex. Trends in Neuroscience **11**: 219-226.

Dietterich, T. G. and Michalski, R. S. 1983. A Comparative Review of Selected Methods for Learning from Examples. *In* Machine Learning: An Artificial Intelligence Approach, *Edited by* R. S. Michalski, J. G. Carbonell and T. M. Mitchell. Tioga Publishing, Palo Alto. 41-81.

Djakow, I. N., Petrowski, N. W. and Rudik, P. A. 1927. Psychologie des Schachspiels. de Gruyter, Berlin.

Egan, D. E. and Schwartz, B. J. 1979. Chunking in Recall of Symbolic Drawings. Memory and

Cognition **7** (2): 149-158.

Eisenstadt, M. and Kareev, Y. 1975. Aspects of Human Problem Solving: The Use of Internal Representations. *In* Explorations in Cognition, *Edited by* D. A. Norman and D. E. Rumelhart. Freeman, San Francisco. 308-346.

Engle, R. W. and Bukstel, L. 1978. Memory Processes among Bridge Players of Differing Expertise. American Journal of Psychology **91**: 673-689.

Epstein, S. L. 1992. Prior Knowledge Strengthens Learning to Control Search in Weak Theory Domains. International Journal of Intelligent Systems **7**: 547-586.

Epstein, S. L. 1994a. For the Right Reasons: The FORR Architecture for Learning in a Skill Domain. Cognitive Science **18** (3): 479-511.

Epstein, S. L. 1994b. Identifying the Right Reasons: Learning to Filter Decision Makers. In Proceedings of the AAAI 1994 Fall Symposium on Relevance., 68-71. AAAI, Palo Alto.

Epstein, S. L. 1994c. Toward an Ideal Trainer. Machine Learning **15** (3): 251-277.

Fawcett, T. E. and Utgoff, P. E. 1991. A Hybrid Method for Feature Generation. In Proceedings of the Eighth International Workshop on Machine Learning, 137-141. Ed. L. A. Birnbaum and G. C. Collins. Morgan Kaufmann, San Mateo.

Fine, R. 1989. The Ideas behind the Chess Openings. Random House, New York. 182 pp.

Flann, N. S. 1992. Correct Abstraction in Counter-Planning: A Knowledge Compilation Approach. Ph.D. thesis diss., Oregon State University.

Freed, M. 1991. Learning Strategic Concepts from Experience: A Seven-Stage Process. In Proceedings of the 13th Annual Conference of the Cognitive Science Society, 132-136. Morgan Kaufmann, San Mateo.

Gasser, R. In press. Solving Nine Men’s Morris. Computational Intelligence :

Gelernter, H. 1963. Realization of a Geometry-Theorem Proving Machine. *In* Computers and Thought, *Edited by* E. A. Feigenbaum and J. Feldman. McGraw-Hill, New York. 134-152.

Gelfer, I. 1991. Positional Chess Handbook. Macmillan, New York. 212 pp.

George and Schaeffer. 1991. Chunking for Experience. *In* Advances in Computer Chess VI, *Edited by* D. F. Beal. Ellis Horwood, London. 133-147.

Glasgow, J. and Papadias, D. 1992. Computational Imagery. Cognitive Science **16** (3): 355-394.

Goldin, S. E. 1978. Memory for the Ordinary: Typicality Effects in Chess Memory: Human Learning and Memory. Journal of Experimental Psychology: Human Learning and Memory **4**: 605-611.

Haussler, D., Kivinen, J. and Warmuth, M. K. 1994. Tight Worst-Case Loss Bounds for Predicting with Expert Advice. Technical Report, Santa Cruz, CA, UCSC-CRL-94-36, University of California at Santa Cruz.

Hendee, W. R. 1993. Cognitive Interpretation of Visual Signals. *In* The Perception of Visual

Information, *Edited by* W. R. Hendee and P. N. T. Wells. Springer-Verlag, Berlin. 134-159.

Hideo, O. 1992. Good Shape. *In* Opening Theory Made Easy, *Edited* Ishi Press, San Jose, CA. 62-111.

Holding, D. 1985. The Psychology of Chess Skill. Lawrence Erlbaum, Hillsdale, NJ.

Iwamoto, K. 1976. Go for Beginners. Random House, New York. 148 pp.

Kandel, E. 1991. Chapter 30: Perception of Motion, Depth, and Form. *In* Principles of Neural Science, *Edited by* E. Kandel, J. Schwartz and T. Jessel. Elsevier, Amsterdam. 440-466.

Kierulf, A. 1989. New Concepts in Computer Othello: Corner Value, Edge Avoidance, Access, and Parity. *In* Heuristic Programming in Artificial Intelligence - The First Computer Olympiad, *Edited by* D. N. L. Levy and D. F. Beal. John Wiley, New York.

Kivinen, J. and Warmuth, M. K. 1994. Exponentiated Gradient Versus Gradient Descent for Linear Predictors. Technical Report, University of California at Santa Cruz.

Kodratoff, Y. and Ganascia, J.-G. 1986. Improving the Generalization Step in Learning. *In* Machine Learning: An Artificial Intelligence Approach - Volume II, *Edited by* R. S. Michalski, J. G. Carbonell and T. M. Mitchell. Tioga Publishing, Palo Alto. 215-244.

Koedinger, K. R. and Anderson, J. R. 1990. Abstract Planning and Perceptual Chunks: Elements of Expertise in Geometry. Cognitive Science **14**: 511-550.

Langley, P., Zytkow, J. M., Simon, H. A. and Bradshaw, G. L. 1986. The Search for Regularity: Four Aspects of Scientific Discovery. *In* Machine Learning: An Artificial Intelligence Approach - Volume II, *Edited by* R. S. Michalski, J. G. Carbonell and T. M. Mitchell. Tioga Publishing, Palo Alto. 425-469.

Lee, K. F. and Mahajan, S. 1990. The Development of a World Class Othello Program. Artificial Intelligence **43** (1): 21-36.

Levinson, R. and Snyder, R. 1991. Adaptive Pattern-Oriented Chess. In Proceedings of the Eighth International Machine Learning Workshop, 85-89. Morgan Kaufmann, San Mateo.

Levinson, R. A., Beach, B., Snyder, R., Dayan, T. and Sohn, K. 1992. Adaptive-Predictive Game-Playing Programs. Journal of Experimental and Theoretical Artificial Intelligence **4**: 315-337.

Littlestone, N. 1988. Learning Quickly when Irrelevant Attributes Abound: A New Linear-threshold Algorithm. Machine Learning **2**: 285-318.

McBurney, D. H. and Collings, V. B. 1977. Chapter 10 - Perception of Objects in Space. *In* Introduction to Sensation/Perception, *Edited* Prentice-Hall, Englewood Cliffs, NJ. 178-215.

Michie, D. 1974. On Machine Intelligence, second edition. second ed. Edinburgh University Press, Edinburgh. 199 pp.

Morales, E. 1991. Learning Features by Experimentation in Chess. In Proceedings of the European Workshop on Learning ‘91, 494-511.

Mostow, D. J. 1983. Machine Transformation of Advice into a Heuristic Search Procedure. *In* Machine Learning: An Artificial Intelligence Approach, *Edited by* R. S. Michalski, J. G. Carbonell and T. M. Mitchell. Tioga Publishing, Palo Alto. 367-403.

Nichelli, P., Grafman, J., Pietrini, P., Alway, D., Carton, J., et al. 1994. Brain Activity in Chess Playing. Nature **369**: 191.

Painter, J. 1993. Pattern Recognition for Decision Making in a Competitive Environment. Master's thesis diss., Hunter College of the City University of New York.

Pell, B. D. 1993. Strategy Generation and Evaluation for Meta-Game Playing. Ph.D. thesis diss., University of Cambridge.

Ratterman, M. J. and Epstein, S. L. 1995. Skilled like a Person: A Comparison of Human and Computer Game Playing. In Proceedings of the Seventeenth Annual Conference of the Cognitive Science Society, Lawrence Erlbaum Associates.

Rich, E. and Knight, K. 1991. Artificial Intelligence. McGraw-Hill, New York. 621 pp.

Sammut, C. and Banerji, R. B. 1986. Learning Concepts by Asking Questions. *In* Machine Learning: An Artificial Intelligence Approach - Volume II, *Edited by* R. S. Michalski, J. G. Carbonell and T. M. Mitchell. Tioga Publishing, Palo Alto. 167-191.

Samuel, A. L. 1963. Some Studies in Machine Learning Using the Game of Checkers. *In* Computers and Thought, *Edited by* E. A. Feigenbaum and J. Feldman. McGraw-Hill, New York. 71-105.

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., Culberson, J., Treloar, N., Knight, B., Lu, P., et al. 1991. Reviving the Game of Checkers. *In* Heuristic Programming in Artificial Intelligence 2 - The Second Computer Olympiad, *Edited by* D. N. L. Levy and D. F. Beal. Ellis Horwood, Chichester, England. 119-136.

Schultz, A. C. and De Jong, K. A. 1988. An Adaptive Othello Player: Experience-Based Learning Applied to Game Playing. In Proceedings of the AAAI Spring Symposium on Game Playing, AAAI, Palo Alto.

Shneiderman, B. 1976. Exploratory Experiments in Programmer Behavior. International Journal of Computer and Information Sciences **5**: 123-143.

Sutton, R. S. 1988. Learning to Predict by the Methods of Temporal Differences. Machine Learning **3**: 9-44.

Ungerleider, L. and Mishkin, M. 1982. Two Cortical Visual Systems. *In* Analysis of Visual Behavior, *Edited by* D. Ingle, M. Goodale and R. Mansfield. MIT Press, Cambridge. 548-586.

Watkins, M. J., Schwartz, D. R. and Lane, D. M. 1984. Does Part-set Cueing Test for Memory Organization? Evidence from Reconstruction of Chess Positions. Canadian Journal of

Psychology **38**: 498-503.

Wilkins, D. 1980. Using Patterns and Plans in Chess. Artificial Intelligence **14**: 165-203.

Yee, R. C., Saxena, S., Utgoff, P. E. and Barto, A. G. 1990. Explaining Temporal Differences to Create Useful Concepts for Evaluating States. In Proceedings of the Eighth National Conference on Artificial Intelligence, 882-888. AAAI Press, Palo Alto, CA.

Yoshio. 1991. All about Thickness. Ishi Press, Mountain View, CA. 194 pp.