Learning Game-Specific Spatially-Oriented Heuristics

Susan L. Epstein
Department of Computer Science
Hunter College and The Graduate School
The City University of New York
New York, NY 10021

Jack Gelfand
Department of Psychology
Princeton University
Princeton, NJ 08544

Esther Twersky Lock
Department of Computer Science
The Graduate School of
The City University of New York
New York, NY 10036

Abstract

This paper describes an architecture that begins with enough general knowledge to play any board game as a novice, and then shifts its decision-making emphasis to learned, game-specific, spatially-oriented heuristics. From its playing experience, it acquires game-specific knowledge about both patterns and spatial concepts. The latter are proceduralized as learned, spatially-oriented heuristics. These heuristics represent a new level of feature aggregation that effectively focuses the program's attention. While training against an external expert, the program integrates these heuristics robustly. After training it exhibits both a new emphasis on spatially-oriented play and the ability to respond to novel situations in a spatially-oriented manner. This significantly improves performance against a variety of opponents. In addition, we address the issue of context on pattern learning. The procedures described here move toward learning spatially-oriented heuristics for autonomous programs in other spatial domains.

Learning Game-Specific Spatially-Oriented Heuristics

Abstract

This paper describes an architecture that begins with enough general knowledge to play any board game as a novice, and then shifts its decision-making emphasis to learned, game-specific, spatially-oriented heuristics. From its playing experience, it acquires game-specific knowledge about both patterns and spatial concepts. The latter are proceduralized as learned, spatially-oriented heuristics. These heuristics represent a new level of feature aggregation that effectively focuses the program's attention. While training against an external expert, the program integrates these heuristics robustly. After training it exhibits both a new emphasis on spatially-oriented play and the ability to respond to novel situations in a spatially-oriented manner. This significantly improves performance against a variety of opponents. In addition, we address the issue of context on pattern learning. The procedures described here move toward learning spatially-oriented heuristics for autonomous programs in other spatial domains.

Introduction

There are many human cognitive and motor skills where representation and sensory processing are optimized through experience with the task (Saitta, Neri, Bajo, Canas, Chaiklin, Esposito, et al. 1995). Often neither the initial statement of the problem nor early experience provides people with an effective and efficient representation. Rather, people acquire these representations gradually, from repeated exposure to them. In this paper we describe an autonomous decision-making program that gradually acquires representations from experience as people do, integrates them with high-level reasoning, and then improves its performance as it shifts its reliance to these representations.

Spatially-oriented representations are useful for two-dimensional board games. Initial directions and rules given to novices, however, are typically statements of relations among a few playing pieces. These rules may be spatial in nature, but are usually of the lowest order. With experience, experts rely on higher order, spatially-oriented strategies to direct play. Indeed, much advice from experts on how to analyze and play board games is conveyed through spatially-oriented concepts. 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; Samuel 1963). Concepts such as shape and thickness are fundamental to the game of Go (Ishida 1991; Otake 1992).

Our approach to learning spatially-oriented heuristics for game playing is based upon visual features which people appear to perceive most easily. The human visual system is hierarchically organized and has detectors for features of various orders (Kandel 1991; Zeki 1993). At the lowest end there are detectors for lines and angles (Kandel 1991). It appears, in addition, based on cognitive psychological measurements, that people also have detectors for combinations of these features (Goldstein 1989;

Treisman 1986). Rather than consider all possible shapes and patterns, our work therefore begins with a set of spatial arrangements of pieces that it processes: straight lines, L's, triangles, squares, and diagonals. This particular set, represented in Figure 1(a), was chosen with grouping principles similar to those used to construct Gestalt visual primitives (Goldstein 1989). Any one of these arrangements can be fitted to a game board in several ways, as the L-shape is to the tic-tac-toe board in Figure 1(b). Playing pieces instantiate those fitted shapes to form patterns, as Figure 1(c) instantiates the small L-shape in Figure 1(b). The central point of this paper is that an algorithm uses those simple shapes to filter through a quantity of data, and manages to extract the salient strategic features in the two games reported here.

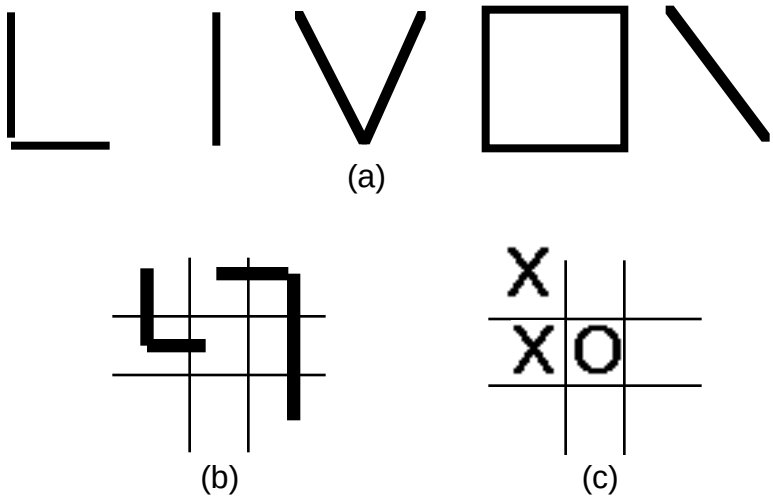


Figure 1: (a) Spatial arrangements of game pieces processed in the algorithms described here. (b) The L-shaped arrangement fitted to a game board two different ways. (c) A fitted L-shape instantiated to produce a tic-tac-toe pattern.

The applications of spatially-oriented representations are by no means limited to game playing. Spatial reasoning has been applied to manufacturing (Penev and Requicha 1997; Vandenbrande and Requicha 1993; Waco and Kim 1994), robot navigation (Kuipers and Levitt 1990), and geographical information systems (Egenhofer and Mark 1995). Often, the salient spatial features in many applications such as these are not known in advance, and the ability to learn them is essential to a high-performance system. One could, in principle, reason spatially about a particular task and use axioms of spatial reasoning to infer higher-order spatial concepts about the domain. The inherent complexity of these domains, however, and the complexity of spatial inference itself, make it difficult to deduce higher-order spatial knowledge from first principles. The purpose of this research is to provide a mechanism to use experience as a way to focus attention on an application's salient spatial knowledge. For example, sophisticated manufacturing design tasks are initially stated in terms of the features of the individual

parts or pairs of parts, but heuristics about the spatial relationships among groupings of components can be discovered by performing the task. Similarly, when scheduling transportation resources, protocols are initially formulated as utilization rules and tradeoffs among costs, but experience dictates that optimal or nearly optimal solutions have particular spatial characteristics.

Any architecture that autonomously acquires skill in a specific problem class should be able to perform tasks there while learning. For a system to learn through experience, it must be able to perform at some low level of competence that supports the kind of experience required to achieve a higher level of performance through practice. These systems, however, typically have a complete set of knowledge for the particular problem class, as well as a knowledge representation initially optimized for it. In contrast, the architecture described here retains its capacity for general baseline performance while it introduces an additional representation that specializes its behavior to a particular problem class, in this case an individual game.

This paper represents a substantial extension of prior work, where learned spatial knowledge was isolated from other decision elements in the program (Epstein, Gelfand, and Lesniak 1996). Here we directly integrate new spatial knowledge into decision making with an improved version of weight learning. Previously, patterns were extracted only from final contest states; now they are extracted on every move. As a result of these changes, we are able to demonstrate, for the first time, significantly improved performance and a shift with experience to spatially-oriented play. The principal contribution of this work is not a sophisticated spatial reasoning algorithm but rather the ability to learn and integrate new spatial knowledge with other knowledge sources.

The Architecture

FORR (FOr the Right Reasons) is a general architecture for learning and problem solving in a broad domain of related problem classes (Epstein 1994a). FORR solves problems with a mixture of *Advisors*, procedural implementations of particular decision-making rationales. This approach is supported by evidence that people integrate a variety of strategies to accomplish problem solving (Biswas, Goldman, Fisher, Bhuva, and Glewwe 1995; Crowley and Siegler 1993; Ratterman and Epstein 1995). Because all Advisors are not equally important or equally trustworthy, FORR organizes them into the two-tier hierarchy of Figure 2. Advisors in tier 1 are guaranteed to be correct; they perform shallow (if any) search and infer from perfect knowledge. Advisors in tier 2 are heuristic; they too are restricted in their search depth, and any information they may rely upon is inductively generated.

Hoyle is the instantiation of FORR for game playing (Epstein 1992). As in Figure 2, the input to a FORR-based program at any decision point is the current state of the world, a list of the legal actions possible, and a collection of *useful knowledge*, information that is potentially applicable and probably correct for a specific problem class. Useful knowledge is heuristic, learned from experience, and expected to enhance performance. Good openings are an example of useful knowledge for game playing.

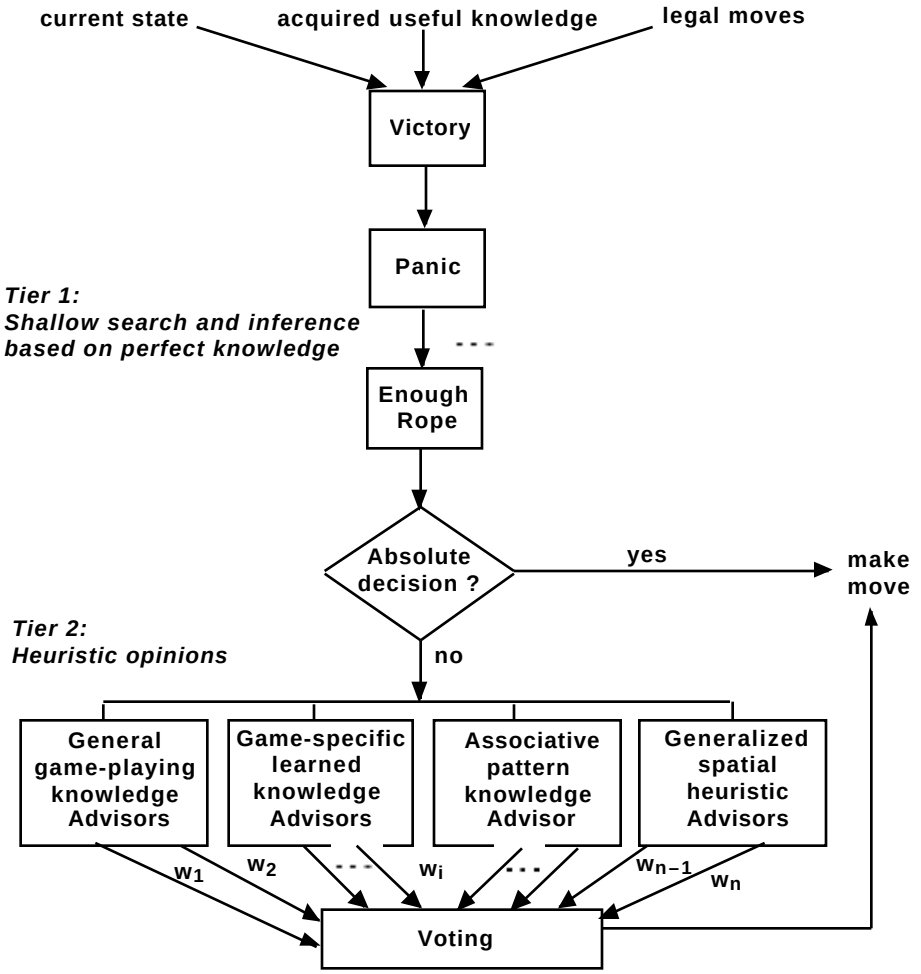


Figure 2: A schematic of decision making in Hoyle.

The other kinds of useful knowledge for game playing include significant states (ones certain to result in a win for a particular contestant assuming perfect play on both sides), forks (two or more overlapping simple plans belonging to the same participant, directed toward different desired states, and requiring one or more common moves), expert moves (made by the other contestant in particular situations), Hoyle's moves (successfully made by the program in particular situations), dangerous states (ones that could not be proved significant during limited learning time, but are suspect nonetheless), and the pattern database described in the next section. Each kind of useful knowledge has its own learning algorithm, triggered at the end of a contest or after a move. (See (Epstein 1992) for further details.)

Each of Hoyle's Advisors is an individual procedure that receives as input the current state, a list of legal moves to consider, and whatever useful knowledge Hoyle has acquired about a particular game thus far. The output of an Advisor is any number of *comments*, each of the form

<Advisor, action, strength>.

Strength is an integer in [0, 10] that indicates the attitude of the Advisor to the action, from firm opposition (0) to strong support (10). Other than these I/O specifications and a fixed time limit within which each must terminate, there is no uniformity imposed upon the Advisors' structure. Advisors need not reference useful knowledge and they do not learn; they simply generate opinions. Consider, for example, the tier-2 Advisor Leery. Given the current state and its legal move options, it checks the useful knowledge and comments against any move that would lead to a dangerous state. The more reliable the dangerous state cache, the more reliable Leery's comments will be.

Decision making in Hoyle considers each tier in turn. The first tier sequentially attempts to compute a decision based upon current useful knowledge, no more than 2-ply search, and simple inference. For example, Victory recommends a move that wins the contest immediately. If the perfectly-correct, game-independent tier-1 Advisors can select a move, they do so, and the second tier is never consulted. Otherwise the Advisors in tier 2 collectively make less reliable comments from individual, narrow heuristic viewpoints, such as Material's "maximize the number of your playing pieces and minimize the number of your opponent's."

Table 1 categorizes Hoyle's Advisors with respect to their generality and the useful knowledge on which they rely. Each Advisor's name is descriptive of its general game-playing perspective. (Anthropomorph, for example, attempts to emulate the human opposition.) In the first category are general game-playing decision makers that do not reference learned useful knowledge. In the second category, Advisors are general game-playing decision-makers that reference learned useful knowledge represented as explicit boards and moves. Advisors in the first two categories are located in both tiers, that is, some are absolutely correct and others are heuristics that vote. Only one Advisor, *Patsy*, applies general game-playing principles to learned, game-specific pattern associations based on visual perception. Finally, some number (determined during execution) of learned, game-specific Advisors (*learned spatial Advisors*) proceduralize spatial concepts based on visual perception. This organization allows the system to function at an acceptable level of performance as it learns to play a particular game.

In tier 2 a move may be supported or opposed by many Advisors. Each tier-2 Advisor has a non-negative weight (e.g., w_1 in Figure 2) that reflects its relevance to and relative significance in a particular game. When a decision must be made in tier 2, Hoyle selects the move with maximal support, summing the product of the strength from each comment about the move with the weight of the commenting Advisor.

Empirical experience with Hoyle indicates that these weights are game specific and should therefore be learned. Initially, the weight of each general game-playing Advisor is set to 1. After every contest Hoyle plays against an expert, *AWL* (Algorithm for Weight Learning) considers, one at a time, only those states in which it was the expert's turn to move. For each such state, AWL distinguishes among support and opposition for the expert's recorded move and for other moves. AWL cumulatively adjusts

Table 1: Hoyle's Advisors for Game Playing.

Name	Tier	Description	Useful Knowledge
General game-playing Advisors that do not rely on learned, game-specific knowledge			
Victory	1	Makes winning move from current state if there is one.	—
Enough Rope	1	Avoids blocking losing move non-mover would have if it were its turn.	—
Candide	2	Formulates and advances naive offensive plans.	—
Challenge	2	Moves to maximize its number of winning lines or minimize non-mover's.	—
Coverage	2	Maximizes mover's influence on predrawn board lines or minimizes non-mover's.	—
Freedom	2	Moves to maximize number of its immediate next moves or minimize non-mover's.	—
Greedy	2	Moves to advance more than one winning line.	—
Material	2	Moves to increase number of its pieces or decrease those of non-mover.	—
Vulnerable	2	Reduces non-mover's capture moves on two-ply lookahead.	—
Worried	2	Observes and destroys naive offensive plans of non-mover.	—
General game-playing Advisors that rely on learned, game-specific knowledge			
Wiser	1	Makes correct move if current state is remembered as certain win.	Significant states
Sadder	1	Resigns if current state is remembered as certain loss.	Significant states
Don't Lose	1	Eliminates any move that results in immediate loss.	Significant states
Panic	1	Blocks winning move non-mover would have if it were its turn now.	Significant states
Shortsight	1	Advises for or against moves based on two-ply lookahead.	Significant states
Anthropomorph	2	Moves as winning or drawing non-Hoyle expert did.	Expert moves
Cyber	2	Moves as winning or drawing Hoyle did.	Hoyle moves
Leery	2	Avoids moves to state from which loss occurred, but where limited search proved no certain failure.	Dangerous states
Not Again	2	Avoids moving as losing Hoyle did.	Hoyle moves
Open	2	Recommends previously-observed expert openings.	Opening database
Pitchfork	2	Advances offensive forks or destroys defensive ones.	Forks
General game-playing Advisor that relies on learned, game-specific spatial heuristics			
Patsy	2	Supports or opposes moves based on their patterns' associated outcomes	Pattern cache
Learned game-specific Advisors that rely on learned, game-specific spatial concepts			
Learned spatial Advisors	2	Supports or opposes moves based on their creation or destruction of a single pattern.	—

the weights of tier-2 Advisors at the end of each contest, and uses those updated weights to make decisions throughout the subsequent contest. Essentially, Hoyle learns to what extent each of its tier-2 Advisors simulates expertise, as exemplified by the expert's moves. Eventually, 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. AWL is based upon Winnow, a fast, perceptron-like learning algorithm (Blum 1997; Littlestone 1988).

In some ways, Hoyle is similar to a mixture of linear, independent, weighted experts, where each Advisor is an expert with a particular decision-making point of view (Chatterjee and Chatterjee 1987; Jacobs 1995). What distinguishes Hoyle from a simple mixture of experts is the hierarchical organization of the underlying FORR architecture. Tier-1 makes decisions based on perfect information and prevents the system from making obvious errors. A decision made in tier 1 also avoids any additional computation that the experts in tier 2 would have required.

Learning and Applying Patterns

For Hoyle, a *pattern* is a visually-perceived regularity, represented as a small geometric arrangements of playing pieces (e.g., black or X) and *blanks* (unoccupied positions) in a particular geographic location. A move can create a pattern by providing some missing piece or blank, or, in games where pieces are not permanently placed, destroy a pattern by removing one. When it first learns a new game, Hoyle constructs a set of board-dependent *templates* as a filter for its perceived patterns: straight lines, L's, triangles, squares, and diagonals of a limited size composed of legal piece positions. The use of these patterns was inspired by repeated laboratory experiences with people who relied upon familiar, sometimes symmetrically transposed patterns while learning (Ratterman and Epstein 1995). When a template is instantiated with some combination of pieces, blanks, and don't care (#) symbols, it becomes a pattern. (Further details on this process are available in (Epstein, Gelfand, and Lesniak 1996).)

The *associative pattern store* (a pattern queue, a pattern cache, and generated spatial concepts) is a heuristically-organized database that links patterns with *contest outcome* (win, loss, or draw). Figure 3 is an overview of the development of Hoyle's spatial orientation from the game-specific associative pattern store. After each contest, the patterns newly created by each move are extracted with the templates. Next, patterns are associated with winning, losing, or drawing, and stored in a pattern queue. Patterns that persist on the pattern queue over time and are identified with a single consistent outcome enter the pattern cache.

Proceduralization is the transformation of expert knowledge into expert behavior. Patterns in the cache are proceduralized with Patsy. Periodic sweeps through the pattern cache also attempt to generalize sets of patterns into spatial concepts. Each concept is proceduralized as an individual, game-specific, learned spatial Advisor, a heuristic that is then validated during subsequent learning. Because pattern knowledge is extensive and contradictory, each segment of the associative pattern store relates

differently to decision making. Queued patterns have no impact at all, cached patterns serve as input to Patsy, and pattern-based concepts become game-specific, learned spatial Advisors. Both patterns and learned Advisors are efficiently represented. A hierarchy from less to more specific speeds pattern matching and supports subsumption testing. In addition, Hoyle normalizes with respect to the symmetries of the plane, thereby reducing matching costs.

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, game-specific cache. (No new patterns are cached during this process.) Each pattern is assigned a value computed from the total number of won, lost and drawn contests since the pattern was first seen. The strength of Patsy's comment on each legal next move is a function of the values of the patterns created by the move in the state to which it leads. Thus Patsy encourages moves that lead to states introducing patterns associated with a win or a draw for the mover, while it discourages moves that lead to states introducing patterns associated with a loss.

Generalization summarizes a set of detailed experiences into a more useful and efficient representation. Each of Hoyle's three generalization methods is represented as a procedure that

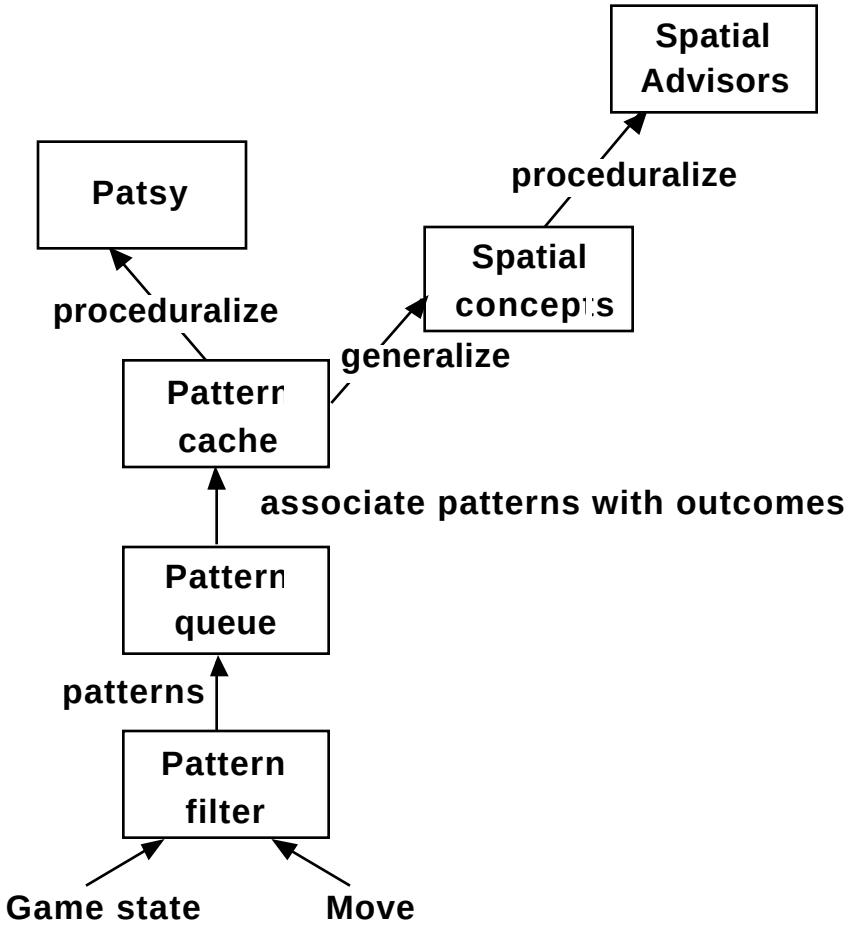


Figure 3: Hoyle's model for spatial learning.

processes existing patterns. After every 10 learning contests, these three generalization processes sweep the pattern cache to form spatial concepts. Patterns *agree* when they originate from the same template. As shown in the example of Figure 4, one generalization drops a position, a second variabilizes the mover and all pieces, and a third variabilizes the mover and a single piece. During generalization, only patterns with compatible associations may be combined. For example, a pattern must be associated with both a win for X and with a win for O to be generalized as a win for the mover α. In addition, if a new concept subsumes one that already exists, the more specific is eliminated. Each concept is proceduralized as a tier-2, game-specific, learned spatial Advisor.

Learning is sometimes subject to by erroneous early experience that creates learned spatial Advisors that do not provide correct information. As Hoyle's skill develops further and the learned spatial Advisors are introduced into tier 2, some of them may prove irrelevant, self-contradictory, or untrustworthy, despite prior empirical evidence of their validity. To support their smooth integration into tier 2, the weights of learned spatial Advisors are initially discounted by an additional multiplier. This factor begins at 0.1 and reaches 1.0 after the learned spatial Advisor comments appropriately 10 times.

Results

Hoyle now learns pattern associations and game-specific spatial Advisors while it plays. The games we used were tic-tac-toe and *lose tic-tac-toe* (played like tic-tac-toe but whoever gets three in a row, column, or diagonal first loses). Both are *draw games,* that is, play between two perfect contestants must, by the nature of the game graph, end in a draw. Because tic-tac-toe and lose tic-tac-toe have the same board, they begin with the same templates. Tic-tac-toe is extremely easy for Hoyle to learn well,

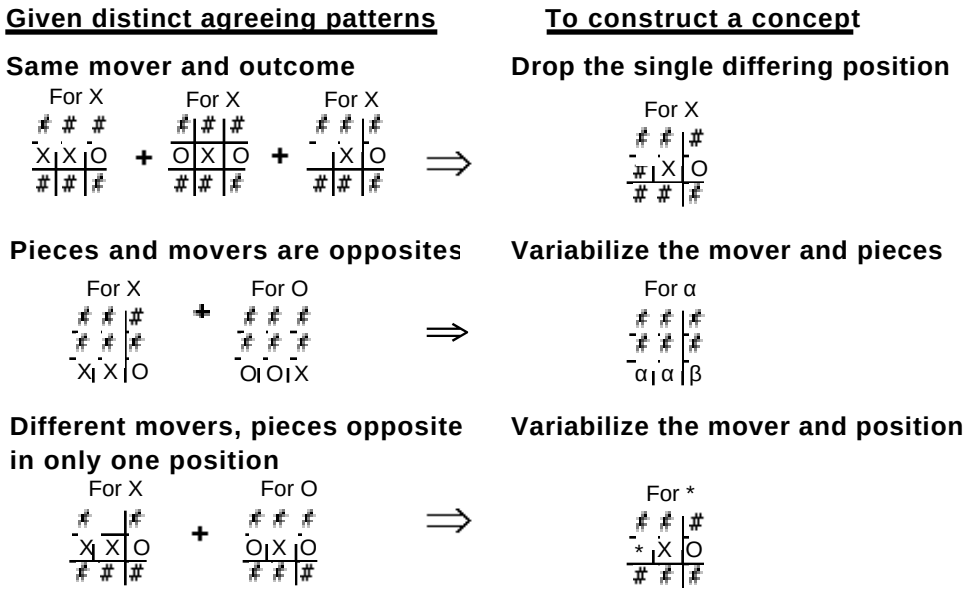


Figure 4: Generalizing patterns into spatial concepts.

and we expected no improvement; it was present only to demonstrate that weights, patterns, and learned spatial Advisors were game-board independent. Lose tic-tac-toe is a far more difficult game to learn to play well, both for people and for machines (Ratterman and Epstein 1995). It has been solved mathematically (Cohen 1972) and the correct strategies for the two contestants are different. Thus it forces the program to distinguish between patterns and concepts good for one contestant from those good for both.

A *run* here is learning followed by testing. On each run, Hoyle alternately moved first in one contest and second in the next. During learning, the other contestant was a hand-coded *perfect player* for the game being learned, one that always moved to secure the best possible outcome despite subsequent error-free play by the opposition, and chose from among equally good moves at random. After learning, Hoyle was tested against three *challengers*: the same perfect player, a 10% random and 90% perfect player, and a 70% random and 30% perfect player. Because Hoyle is non-deterministic (it breaks ties in tier-2 voting by selecting a top-ranked move at random), data is averaged over 10 runs to form a single *experiment*.. To begin, each experiment was run twice: once with a version of Hoyle that had AWL but neither patterns nor learned spatial Advisors, and a second time with Hoyle including Patsy and any learned spatial Advisors that emerge from the process in Figure 3. Improvements cited here are statistically significant at the 95% confidence level.

Hoyle learns to value *pattern-oriented* play (i.e., Patsy and the learned spatial Advisors) highly. After learning in 200 tic-tac-toe contests, 32.6% of all weight is assigned to Advisors that are pattern-oriented, and Patsy and the best learned spatial Advisor always have the top two weights. In lose tic-tac-toe, 29.3% of all weight is assigned to Advisors that are pattern-oriented, and Patsy ranks second on all but one run, where it ranks third. On 80% of the runs, Hoyle learned at least one spatial Advisor for lose tic-tac-toe with weight at least one, and that Advisor ranked fifth on average. Learning new spatially-oriented heuristics also reduces the number of contests Hoyle requires to develop consistent expert performance. With patterns and the learned spatial Advisors, the program never lost at lose tic-tac-toe during learning after contest 29.0, versus contest 56.0 without patterns and the learned spatial Advisors.

Because learned spatial Advisors are produced by induction, not all of them are correct. This is especially true during early learning experience, when Hoyle is not yet playing well enough to exploit good pattern knowledge. Figure 5 shows three learned spatial Advisors for lose tic-tac-toe, and how AWL adjusted their weights based on their performance in one run of 200 contests. Advisor 1 is the horizontal and (through symmetry) vertical portion of the heuristic “reflect through the center,” proved optimal play for most situations in the game (Cohen 1972). The weight of Advisor 1 increases rapidly after its creation. Advisor 2 advocates playing in a row that does not go through the center, where each contestant already has one piece and the non-mover has a corner. Advisor 2 recommends a correct but infrequently applicable action, and its weight increases moderately. Advisor 3 recommends a move into

a side row between an O and a blank. The weight of Advisor 3 initially increases but then falls off rapidly as Hoyle finds it misleading and discounts it on the basis of further experience.

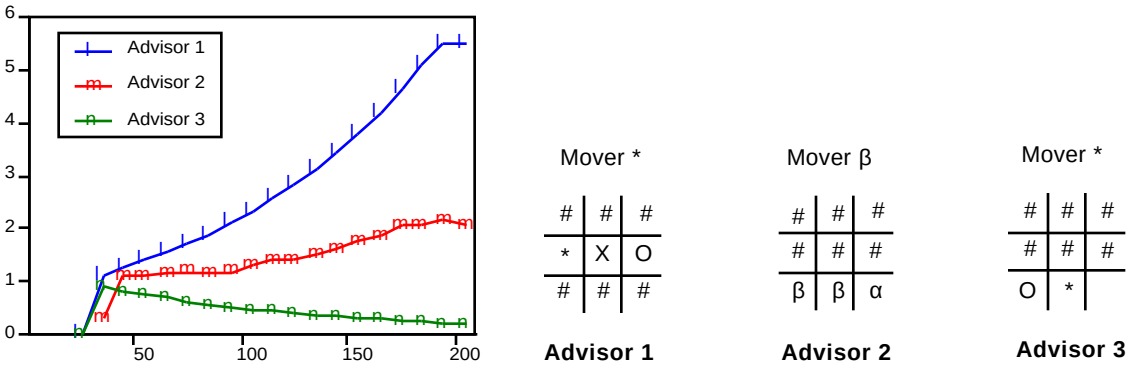
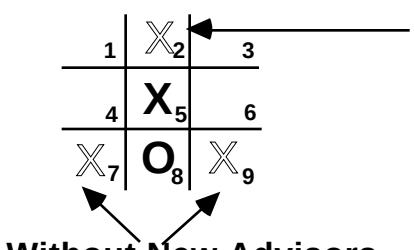


Figure 5: Three learned spatial Advisors for lose tic-tac-toe, and their weights during 200 consecutive contests. The mover for each Advisor is in the current state; the pattern is matched for in the subsequent state In an α−β Advisor, either α = X and β = O or α = O and β = X. In an * Advisor, * is either X or O consistently.

Two manifestations of Hoyle's learned spatial orientation are its performance against different opponents and its response in novel situations. Recall that Hoyle trains against a perfect player, which should result in stereotyped movement in the game tree (Epstein 1994b). During testing, however, the challengers with varying degrees of randomness in their responses introduce states in the game tree provably never experienced by Hoyle during learning. Table 2 shows the performance improvements pattern-oriented play offers. Note how Hoyle with patterns is better able to win and draw at lose tic-tac-toe in most of the categories, despite the differences between its trainer and its challengers.

Table 2: Average and standard deviation of performance with and without spatial orientation against three challengers. Boldface is an improvement over play without patterns at the 95% confidence level. Estimated optima are in italics.

Challenger	**Perfect Player**		**90% Perfect**		**30% Perfect**	
	Wins+Draws	**Wins**	**Wins+Draws**	**Wins**	**Wins+Draws**	**Wins**
Tic-tac-toe	*100.0*	—	*100.0*	*16.4*	*100.0*	*80.7*
Without patterns	100.0 (0.0)	—	98.0 (4.0)	18.0 (7.5)	97.0 (6.4)	83.0 (11.9)
Pattern-oriented	100.0 (0.0)	—	97.0 (6.4)	13.0 (12.8)	94.0 (4.9)	77.0 (13.5)
Context and weight ≥ 1 only	100.0 (0.0)	—	**100.0** (0.0)	**22.0** (16.6)	**99.0** (3.0)	85.0 (11.2)
Lose tic-tac-toe	*100.0*	—	*100.0*	*18.5*	*100.0*	*66.4*
Without patterns	100.0 (0.0)	—	96.0 (4.9)	18.0 (7.5)	73.0 (7.8)	54.0 (9.2)
Pattern-oriented	100.0 (0.0)	—	**98.0** (6.0)	18.0 (8.7)	**92.0** (6.0)	49.0 (11.4)
Weight > 2 only	100.0 (0.0)	—	**99.0** (3.0)	18.0 (11.7)	**96.0** (6.6)	**68.0** (8.7)



With New Advisors

move	*vote*
1 and 3	35.7
2	59.2
4 and 6	15.4
7 and 9	43.8

Without New Advisors

move	*vote*
1 and 3	34.8
2	15.3
4 and 6	12.8
7 and 9	43.0

Figure 6: A learned spatial Advisor affects decision making.

Figure 6 demonstrates how spatially-oriented play improves behavior in novel situations. The position shown from lose tic-tac-toe never occurs during learning against the perfect player when Hoyle is X, but arises often in testing against the other three challengers. Without the learned spatial Advisors, Hoyle typically votes to move into a corner adjacent to O's move. Advisor 1 of Figure 5, however, though it has never experienced the specific instance of reflection through the center required here, swings the vote to do so because of the generalization used in its concept formation algorithm. Thus Hoyle can make correct decisions against the imperfect challengers in situations it has never seen during training. It parlays inductive spatial generalizations into game playing expertise.

There is, however, a danger in learning heuristics reflected in Table 2 for tic-tac-toe. In the last five categories, performance actually degrades with pattern-oriented play. Inspection revealed that Hoyle overgeneralized on tic-tac-toe. On every run Hoyle learned only a single new spatial Advisor that ever commented: "move into the corner of an empty row or column." Let us call this Advisor *Corner*. On every run Patsy had the highest weight, and Corner had the second highest. Corner reduced Hoyle's reliability and power against the imperfect challengers. Inspection revealed that Hoyle's new errors all stemmed from Corner's incorrect response when playing O to a side-square opening by X, shown in Figure 7. Because the perfect player always opens in the center or a corner, Hoyle had never encountered this opening during learning. Without Patsy and Corner, Hoyle correctly plays the center. Patsy, however, learned a specific pattern in the context of competition against a perfect player, a pattern that does not always apply when the competition is not perfectly expert. Corner suffers from a similar lack of context. Together, every time an imperfect challenger opens in a side square, Patsy and Corner swing the vote in tier 2 to a corner not adjacent to X, a fatal error. (X will play the corner between its first move and O. A sequence of forced moves then guarantees a win for X.)

The use of context should correct such problems with learned heuristics. In this case, the strength of a

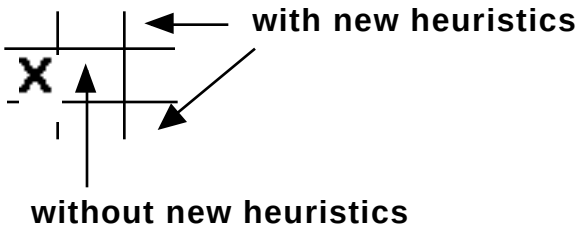


Figure 7: Voting for O's first tic-tac-toe move, Patsy and a learned spatial Advisor incorrectly select two corners.

pattern is in large part determined by the environment in which it is experienced. Hoyle's pattern induction did detect an important tic-tac-toe pattern, *but only for play against a perfect player.* In other words, the induction process overgeneralized to say "corners in empty rows or columns are good," whereas the more accurate statement would be "if someone already holds the center, corners in empty rows or columns are good." Thus heuristic learning is demonstrably context-dependent. When the context changes, the automatic response to a heuristic (in this case playing the corner of an open column) may be wrong.

Nonetheless, the Advisor Corner gives generally good advice. Rather than keep a list of caveats, our preference is to broaden the context in which the pattern is learned, that is, to learn against all the challengers. We therefore also ran two *high-weight* experiments. In the first, for tic-tac-toe, Hoyle learned new spatial Advisors as before but during testing only Advisors with weight at least 1 were permitted to comment. This time the error in Figure 7 never occurred, and Hoyle's performance actually improved in three categories as Table 2 indicates. Hoyle simply needed additional knowledge to prevent the misuse of the new spatial Advisor; knowledge it acquired with broader training. In the second high-weight experiment, for lose tic-tac-toe, the weight minimum was 2 but we did not change the learning environment, and performance was even stronger. Emphasis on its most reliable Advisors, of any kind, clearly helps Hoyle play better.

Discussion

The use of heuristic, rather than absolutely correct, rationales in decision making is supported by evidence that people *satisfice*, that is, they make decisions that are good enough (Simon 1981). Although satisficing solutions are not always optimal, they may achieve a high level of expertise.

In people, perceptual salience appears to cue functional significance, directing human attention (Tversky 1989). Psychologists hypothesize, therefore, that attention to salient parts could cue learning about function (Tversky 1989). Without the don't cares, Hoyle's learned spatial Advisors are salient parts of an image of the board state, and Hoyle learns how to use them. Indeed, these spatial features

simulate some aspects of human visual cognition. As game players become more expert, they value higher order, spatially-oriented representations as explanations and as strategies. Prior to this work, one of our students, who built a perfect player for a game board with positions at the center and on the circumference of a circle, eventually described "the death triangle," a pattern she avoided at all costs. Another student, building a perfect player for a grid-based game, developed a theorem that explained the "power of the black L," his key to understanding the problem. Neither game is included here, but the features we describe would detect both patterns. Finally, in a separate study on five- and six-year old children playing games, at least one subject gave a clear explanation of the significance (to him) of a "Y" in tic-tac-toe (Ratterman and Epstein in preparation).

The language in which to represent spatial concepts is an important issue. One cannot predict in advance precisely *which* spatial features are crucial, and therefore an ample potential vocabulary for them is needed. Complete representations, however, are extraordinarily costly and not particularly incisive. Patsy's templates include only five possible shapes. Therein lie both their power and their fallibility: power in that they focus attention on some salient shapes, and fallibility in that the language is incomplete. A complete pattern language, say a boolean one on all possible pieces on all possible locations such as ELF (Utgoff and Precup 1997), is unwieldy, and the patterns are not strategically significant. As a result, it is unlikely to detect important concepts quickly. Current work involves bootstrapping from these initial patterns to other shapes that prove important.

We do not claim that this program is a full cognitive model, but it does contain many appropriate elements which enhance its performance as an autonomous learning program. For example, the templates that filter perceived patterns and curtail a potential combinatoric explosion are not an ad hoc device, but are inherent in the human perceptual system (Goldstein 1989). Future work includes more difficult games and spatial concept formation based upon higher-order features such as center, edge, perimeter, bounded regions, length and area. We are also aware that games like chess, in which each contestant has several kinds of pieces, would require modifications to the pattern identification and generalization techniques. To control costs, focus upon the more powerful pieces would probably be a good approach.

There are other kinds of patterns that are not visually cued, such as threat and defense descriptions (Levinson and Snyder 1991). We do not claim that people use only visual patterns, or only the patterns representable in this study, but we do consider them an essential component of a robust game player. As described here, Hoyle now attends to patterns when they are significant, and relies on other decision-making factors when that is more appropriate.

This paper reports results for tic-tac-toe and lose tic-tac-toe. Although the latter is quite difficult for people to learn (Ratterman and Epstein 1995) and has many possible patterns, our method found its salient features using a highly restricted set of templates generated from shapes known to be perceived

readily by people. It is our expectation that these shapes will continue to serve well as we scale up to morris games, because a key element in strategic play there is the establishment of a mill, three identical pieces in a row. One would therefore expect that the strategically important spatial features of morris games would be accessible through a set of simple templates such as those used here. Indeed, even in chess, all but one of the pieces (the knight) can only threaten squares that are physically accessible directly from them. This results in strategic spatial relationships composed from relatively simple spatial features. In larger problems, search-based approaches must be responsive to increases in branch factor and state space size. Our method, however, relies on a family of heuristics instead of deep search. So long as the number of learned spatial heuristics is well controlled, scaling up should remain manageable.

Acknowledgments

We acknowledge helpful discussions with Phil Johnson-Laird, Ron Kinchla, and Anne Treisman. This work was supported in part by NSF grant #9423085, PSC-CUNY #666318, The New Jersey Center for Multi-Media Research, and the NSF Center for Digital Video and Media.

References

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

Blum, A. 1997. Empirical Support for Winnow and Weighted-Majority Based Algorithms: Results on a Calendar Scheduling Domain. *Machine Learning*, 26: 5-23.

Chatterjee, S. and Chatterjee, S. 1987. On Combining Expert Opinions. *American Journal of Mathematical and Management Sciences*, 7(3-4): 271-295.

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.

Egenhofer, M. J. and Mark, D. M. 1995. Naive Geography, 95-8, National Center for Geographic Information and Analysis.

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. Toward an Ideal Trainer. *Machine Learning*, 15(3): 251-277.

Epstein, S. L., Gelfand, J. and Lesniak, J. 1996. Pattern-Based Learning and Spatially-Oriented Concept Formation with a Multi-Agent, Decision-Making Expert. *Computational Intelligence*, 12(1): 199-221.

Fine, R. 1989. *The Ideas behind the Chess Openings*. New York: Random House.

Gelfer, I. 1991. *Positional Chess Handbook*. New York: Macmillan.

Goldstein, E. 1989. *Sensation and Perception*(third ed.). Belmont, CA: Wadsworth.

Ishida, Y. 1991. *All about Thickness*. San Jose, CA: Ishi Press.

Jacobs, R. A. 1995. Methods for Combining Experts' Probability Assessments. *Neural Computation*, 7: 867-888.

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

Kuipers, B. and Levitt, T. 1990. Navigation and Mapping in Large-Scale Space. In S. Chen (Ed.), *Advances in Spatial Reasoning*, 207-251. Norwood, NJ: Ablex.

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.

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

Otake, H. 1992. Good Shape. In *Opening Theory Made Easy*, 62-111. San Jose, CA: Ishi Press.

Penev, K. and Requicha, A. A. G. 1997. Automatic Fixture Synthesis in 3D. In *Proceedings of the IEEE International Conference on Robotics & Automation*, 1713-1718. Albuquerque, NM: IEEE Press.

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*, 709-714. Pittsburgh: Lawrence Erlbaum Associates.

Ratterman, M. J. and Epstein, S. L. in preparation. Game-Playing Expertise in Young Children. :

Saitta, L., Neri, F., Bajo, M., Canas, J., Chaiklin, S., Esposito, F., Kayser, D., Nedellec, C., Sabah, G., Tiberghien, A., et al. 1995. Knowledge Representation Changes in Humans and Machines. In P. Reimenn, & H. Spada (Ed.), *Learning in Humans and Machines*, Oxford: Elsevier Science.

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

Simon, H. A. 1981. *The Sciences of the Artificial*(second ed.). Cambridge, MA: MIT Press.

Treisman, A. 1986. Features and Objects in Visual Processing. *Scientific American*, 255: 114-125.

Tversky, B. 1989. Parts, Partonomies, and Taxonomies. *Developmental Psychology*, 25(6): 983-995.

Utgoff, P. and Precup, D. 1997. Constructive Function Approximation, 97-04, Department of Computer Science, University of Massachusetts.

Vandenbrande, J. and Requicha, A. 1993. Spatial Reasoning for the Automatic Recognition of Machinable Features in Solid Models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(10): 1269-1285.

Waco, D. and Kim, Y. 1994. Geometric Reasoning for Machining Features Using Convex Decomposition. *Computer-Aided Design*, 26(6):

Zeki, S. 1993. *A Vision of the Brain*. Oxford: Blackwell Scientific.