This paper appeared in the *Proceedings of the AAAI Workshop on Computational Cognitive Modeling: Source of the Power*, 1996. Portland, OR, AAAI.

The Creation of New Problem Solving Agents from Experience with Visual Features

Susan L. Epstein
Department of Computer Science
Hunter College and the Graduate School of
The City University of New York
New York, NY 10021
epstein@roz.hunter.cuny.edu

Jack Gelfand
Department of Psychology
Princeton University
Princeton, NJ 08544
jjg@princeton.edu

Abstract

A significant source of power in a cognitive system is the ability to represent task-based experience appropriately. As people acquire skill in a cognitive task, they rely in part on spatially-oriented reasoning. This paper describes a multi-agent decision-making architecture that addresses a domain of related problem classes. The architecture has a general capacity to perform tasks in the domain, so that it can gain experience within a specific problem class. It also has the ability to generate new, problem-class-specific, spatially-oriented reasoning agents from experience. Much of the knowledge encapsulated by the correct new agents was previously inexpressible in the program's representation and, in some cases, not readily deducible from the structure of the problem class.

Introduction

There are many examples in human cognitive and motor processing in which representation and sensory processing are optimized through experience with the task (Gelfand, Flax, Endres, Lane and Handelman 1992b; Gelfand, Flax, Endres, Lane and Handelman 1992a; Gelfand, Handelman, Lane and Epstein 1994). Often people are not aware of the optimum representation from the initial statement of the problem, or from early experience. Rather, humans acquire these representations gradually, from repeated exposure to them. One interesting domain where the acquisition of spatially-oriented reasoning has substantial impact is game playing.

Our work on 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. Even six-year-olds describe their observation of expert play and their own decisions in terms of the same kinds of patterns (Ratterman and Epstein 1996).

Initial directions and rules given to novices are usually statements of relations among a small number of pieces. These rules may be spatial in nature, but are usually of the lowest order. As game players become more expert, they rely on higher-order, spatially derived strategies to direct play. Advice from experts on how to analyze and play games is repeatedly 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 (Samuel 1963; Lee and Mahajan 1988; Fine 1989; Gelfer 1991). Concepts such as shape and thickness are fundamental to the game of Go (Iwamoto 1976; Yoshio 1991; Hideo 1992).

An important feature of any architecture that acquires problem-class-specific skill is the ability to perform tasks 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. Many cognitive models have such ability, and can perform tasks in a serviceable fashion while mechanisms such as chunking operate to provide a higher level of performance with experience (Newell 1990). These systems, however, typically have a complete set of knowledge for a particular problem, and a knowledge representation initially optimized for the problem class. This paper describes a modular, multi-agent architecture with the capacity for general performance, as well as the ability to generate agents with problem-class-specific knowledge. The experimental domain is game playing, and the problem-class-specific agents it learns are for individual games.

Program Architecture

The architecture we discuss here is based on *FORR*, a general architecture for learning and problem solving in a broad domain of related problem classes (Epstein 1994a). The instantiation of FORR for game playing is called *Hoyle* (Epstein 1992). The use of a multi-agent architecture is supported by evidence that humans integrate a variety of strategies in order to accomplish problem solving (Crowley and Siegler 1993; Biswas, Goldman, Fisher, Bhuva and Glewwe 1995; Ratterman and Epstein 1995). The brain appears to use a modular architecture to accomplish integration of information (Ungerleider and Mishkin 1982; DeYoe and Van Essen 1988). Additionally there is

evidence that different parts of the brain are activated when decisions are being made about different strategic aspects of chess (Nichelli, Grafman, Pietrini, Alway, Carton and Miletich 1994).

A schematic diagram of this system is shown in Figure 1. A hierarchy of resource-limited procedures called *Advisors* is provided with the current game state and legal moves. Hoyle's Advisor hierarchy has two tiers. The first tier sequentially attempts to compute a decision based upon current knowledge, shallow (no more than 2-ply) search, and simple inference. An example of this is Victory which 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. If no single decision is forthcoming, then the second tier of heuristic but generally correct Advisors collectively makes less reliable decisions based on a set of individual, narrow heuristic viewpoints, such as Material's "maximize the number of your playing pieces and minimize your opponent's." Based upon the strengths of the Advisors' responses, a simple arithmetic vote selects the move.

A complete list of Advisors and their characteristics appears in Table 1. As can be seen from Table 1, Advisors can be categorized with respect to their generality and learning capabilities. This organization allows the system to function at an acceptable level of performance as it learns to play a particular game. The first category is game-playing agents that do not learn problem-class-specific knowledge but have decision-making abilities in the general game-playing domain. These Advisors are located in the both the first and second tier, that is, some base their decision on perfect knowledge, and some are general game playing heuristics that participate in the voting process. In the second category, Advisors can learn problem-class-specific knowledge, represented as direct comparisons of board states or moves.

In the third category, Advisors base their recommendations on spatial configurations of pieces. There are two levels of representation in the spatially-oriented Advisors. On the first level is *Patsy*, an Advisor that remembers and directly associates specific patterns with wins, losses, or draws. On the second level is a set of spatially-oriented agents, Advisors proceduralized from spatial concepts. Each spatial concept is a generalization over individual patterns, and each produces an individual agent. The generation of these spatial Advisors is described in the next section.

Learning to Use and Apply Patterns

In our work, a *pattern* is a visually-perceived regularity, represented as a small geometric arrangement of *marker types* (e.g., black, X) and unoccupied positions (*blanks*) in a particular geographical location. An 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 and generated spatial concepts.

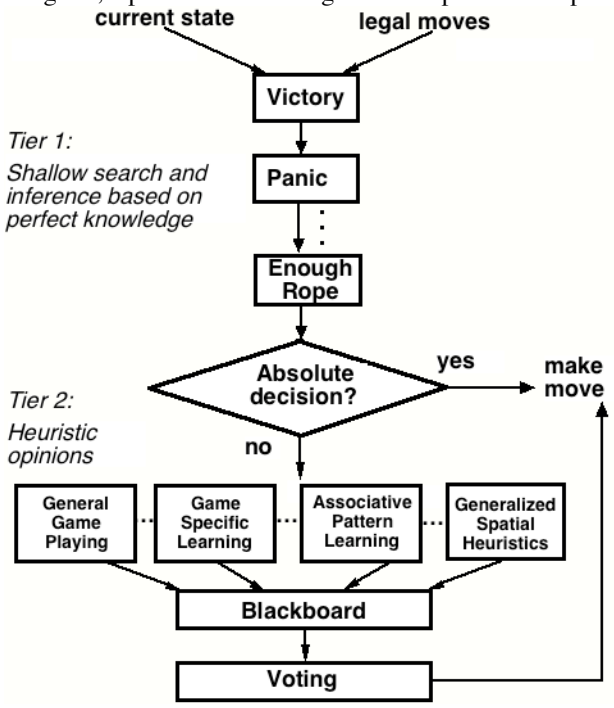

Figure 1: A schematic diagram of decision making in Hoyle.

Figure 2 provides an overview of the system and the development of pattern-based Advisors from the game-specific associative pattern store. There are four stages detailed here: associate, generalize, proceduralize, and validate. Once patterns are identified, they are associated with winning, losing, or drawing and stored in a pattern queue. Patterns that persist over time and are identified with a single consistent outcome move from the pattern queue to the pattern cache. Patterns in the cache are proceduralized via an associative pattern classifier, a new, game-independent Advisor called Patsy. Periodic sweeps through the pattern cache also attempt to generalize sets of patterns into concepts. Each concept is proceduralized as an individual, game-specific Advisor that is then validated during subsequent learning.

Formulating Concepts

Generalization summarizes a set of detailed experiences into a more useful and efficient representation. Currently, the pattern cache is swept once every 10 contests during learning, and 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 of the game. There are two generalization processes. Given distinct agreeing patterns that have the same mover and single, non-zero response, and are identical except in a

single position, Hoyle constructs a new pattern from the remaining positions. Given distinct agreeing patterns such

Table 1: Hoyle's Advisors for Game Playing.

Name	Tier	Description	Useful Knowledge	Learning Strategy
General Game Playing Advisors - No Learning				
Victory	1	Makes the winning move from the current state if there is one.	None	-
Enough Rope	1	Avoids blocking a losing move the non-mover would have if it were its turn now.	None	-
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	-
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	-
Shortcut	2	Bisects the shortest paths between pairs of markers of the same contestant on game board lines.	None	-
Vulnerable	2	Reduces the non-mover's capture moves on two-ply lookahead.	None	-
Worries	2	Observes and destroys naive offensive plans of the non-mover.	None	-
Greedy	2	Moves to advance more than one winning line.	None	-
Advisors that learn domain specific knowledge				
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
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 its turn now.	Significant states	Deduction
Shortsight	1	Advises for or against moves based on a two-ply lookahead.	Significant states	Deduction
Anthropomorph	2	Moves as a winning or drawing non-Hoyle expert did.	Expert moves	Abduction
Cyber	2	Moves as a winning or drawing Hoyle did.	Important contests	Abduction
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
Not Again	2	Avoids moving as a losing Hoyle did.	Important contests	Abduction
Open	2	Recommends previously-observed expert openings.	Opening database	Induction
Pitchfork	2	Advances offensive forks or destroys defensive ones.	Forks	EBL
Advisors that learn domain specific patterns and spatial heuristics				
Patsy	2	Moves to recreate visual patterns credited for positive outcomes in play, and to avoid those blamed for negative ones.	Visual patterns	Associative pattern classifier
Spatial Heuristic Advisors	2	Generalizes spatial heuristics from sets of patterns in Patsy's pattern cache.	Spatial concepts	Generalization

that interchanging the contestants' markers and changing the mover transforms one to the other, Hoyle constructs a new pattern with variable place holders.

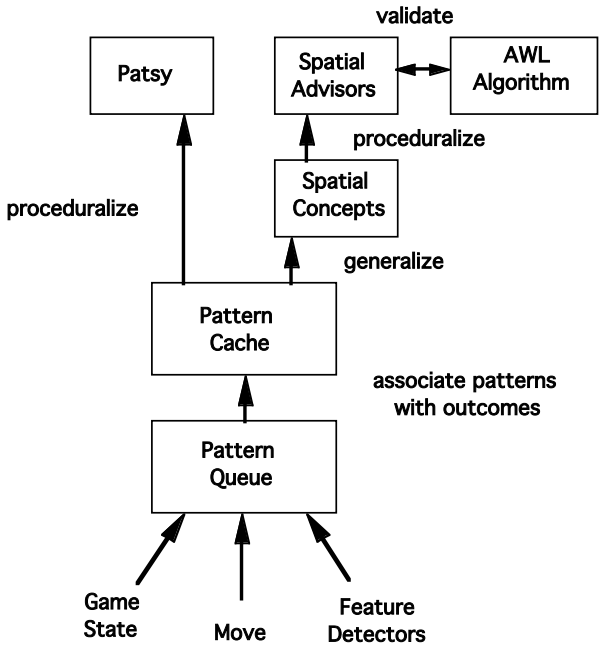

Figure 2: A schematic diagram of associative pattern learning and spatial concept formation in our model.

Proceduralization

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.

The game-independent, tier-2 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, 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 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's marker type, while it discourages moves that lead to states introducing patterns associated with a loss.

Validation of New Advisors

Each concept is proceduralized as a new, tier-2, game-specific Advisor. As these 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 complex. 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. 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 and should therefore be learned.

With an external model of expertise as its performance criterion, we use AWL, a modified version of the perceptron-like Winnow (Littlestone 1988), to learn problem-class-specific weights for tier-2 Advisors (Epstein 1994b). 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 tier-2 Advisors at the end of each contest, and uses those weights to make decisions throughout the subsequent contest.

Results

We have successfully implemented the integration of spatially-oriented Advisors into Hoyle's second tier while learning to play tic-tac-toe and lose tic-tac-toe. Lose tic-tac-toe is played exactly like tic-tac-toe except the first person to get three markers along any row, column or diagonal loses. Tic-tac-toe is easy for Hoyle to learn, but lose tic-tac-toe is considerably more difficult, both for

people (Ratterman and Epstein 1995) and for Hoyle. We report primarily on the results for lose-tic-tac-toe in this paper.

In all 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 (met a behavioral standard of *n*). Since Hoyle had already learned to play lose tic-tac-toe without visual features, these experiments were intended to demonstrate that game-dependent visual patterns exist and persist, despite the non-determinism of the learning experience. They also show that such patterns can be gathered without a combinatoric explosion, and that the transition from waiting list to pattern cache to concept and Advisor is warranted. Furthermore, we show that new, game-specific Advisors can be learned and managed appropriately, all without reducing the program's ability to play well.

The process of learning is sometimes influenced by erroneous early experience and creates Advisors that do not provide correct information. To support the smooth integration of new Advisors into tier 2, the comment strengths of the new Advisors spawned by the system are discounted by an additional multiplier. This factor begins at 0.1 and reaches 1.0 after the Advisor comments appropriately 10 times. Meanwhile, the comments from new Advisors are compared with the behavior of an expert opponent, and their weights are adjusted by AWL.

Figure 3 shows three of the spatial concepts learned in one run of lose tic-tac-toe which was continued until a behavioral standard of 20 draws in a row was reached. If an Advisor is described with α and β, the symbols are interpreted as either α = X and β = O or α = O and β = X.

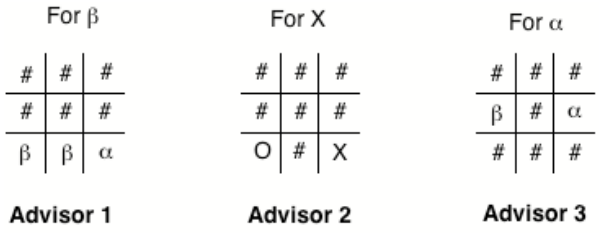

Figure 3: Some pattern-based Advisors learned for lose tic-tac-toe. The mover for each Advisor is in the current state; the pattern is matched for in the subsequent state.

Advisor 1 advocates playing in an *outside row* (one that does not include the center) where each contestant already has one marker. Advisor 2 recommends that X play a corner in a row where O already holds a corner. Advisor 3 is the horizontal and (through symmetry) vertical portion of the heuristic "reflect through center" which has been proven to be optimal play for X in lose tic-tac-toe (Cohen, 1972). Figure 4 shows how AWL adjusts the weights of each of the three new spatial Advisors in Figure 3 on the basis of their performance. The weight of Advisor 3 increases rapidly after its creation. Advisor 1 recommends a correct but not frequently applicable action, and its weight increases moderately. The weight of Advisor 2 on the other hand begins to increase but then falls off

This paper appeared in the *Proceedings of the AAAI Workshop on Computational Cognitive Modeling: Source of the Power*, 1996. Portland, OR, AAAI.

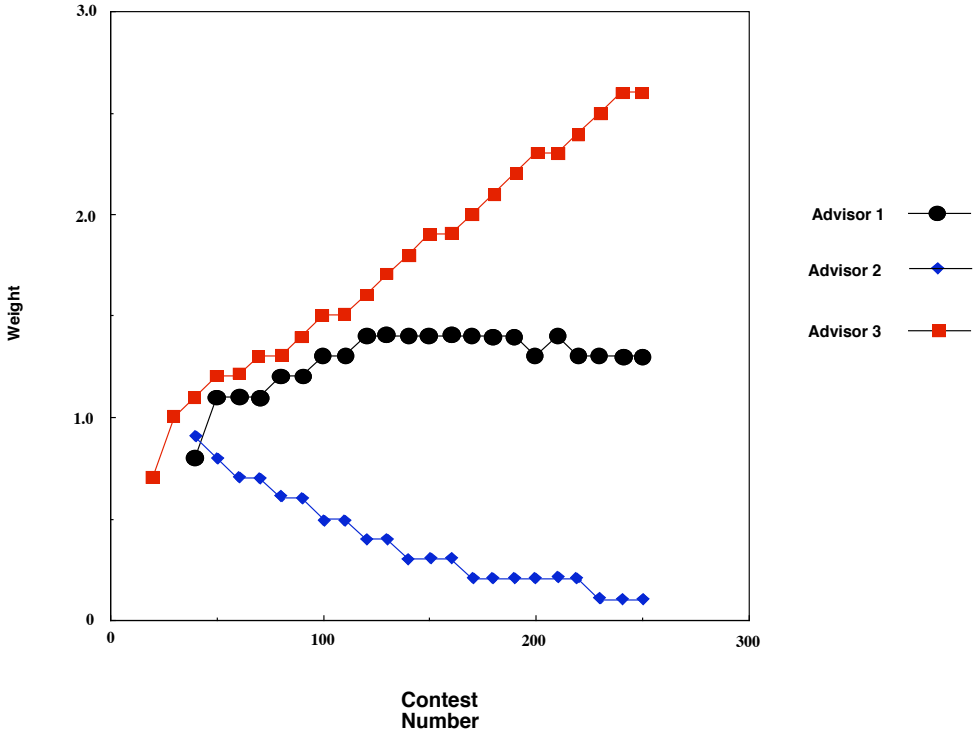

Figure 4: The weights attributed to three learned, spatially-oriented Advisors during 250 consecutive learning contests

rapidly as Hoyle finds it misleading and discounts it on the basis of further experience.

Discussion

As a consequence of the overall process described in this paper, Hoyle plays in a more spatially-oriented fashion with experience. For the initial test of the operation of the system as a whole we used simple games and made a number of simplifications in the individual components of the system. 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. In addition, other spatial representations such as threat and defense relationships (Levinson 1989). should be explored in the context of this system.

Acknowledgements

We acknowledge helpful discussions with Ron Kinchla, Philip Johnson-Laird and Nick Littlestone. This work was supported by a grant from The James S. McDonnell Foundation, NSF grant # 9423085 and PSC-Cuny grant # 666318.

References

Biswas, G., S. Goldman, D. Fisher, B. Bhuva and G. Glewwe. (1995). Assessing Design Activity in Complex CMOS Circuit Design. In P. Nichols, S. Chipman et al (Ed.), *Cognitively Diagnostic Assessment* (pp. 167-185). Hillsdale, NJ: Lawrence Erlbaum.

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

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

Epstein, S. L. (1992). Prior Knowledge Strengthens Learning to Control Search in Weak Theory Domains. 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. New Orleans: AAAI.

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

Gelfand, J., M. Flax, R. Endres, S. Lane and D. Handelman (1992a). Acquisition of Automatic Activity through Practice: Changes in Sensory Input. In *Proceedings of the Tenth National Conference on Artificial Intelligence*, 189-193. San Jose: AAAI Press.

Gelfand, J., M. Flax, R. Endres, S. Lane and D. Handelman. (1992b). Senses, Skills, Reactions and Reflexes: Learning Automatic Behaviors in Multi-Sensory Robotic Systems. In G. Bekey and K. Goldberg (Ed.), *Neural Networks in Robotics* (pp. 319-330). Boston: Kluwer Academic.

Gelfand, J., D. Handelman, S. Lane and S. Epstein. (1994). Adapting Human Functional Architectures and Behaviors for Intelligent Machines. In J. Hendler (Ed.), *Handbook of Neuropsychology* (pp. 361-376). Amsterdam: Elsevier.

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

Hideo, O. (1992). Good Shape. In (Ed.), *Opening Theory Made Easy* (pp. 62-111). San Jose, CA: Ishi Press.

Iwamoto, K. (1976). *Go for Beginners*. (Ed.). New York: Random House.

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

Levinson, R. A. (1989). A Self-Learning, Pattern-Oriented Chess Program. In *Proceedings of the Conference on New Directions in Game-Tree Search*, 2-11. Edmonton, Canada:

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

Mostow, D. J. (1983). Machine Transformation of Advice into a Heuristic Search Procedure. In R. S. Michalski, J. G. Carbonell et al (Ed.), *Machine Learning: An Artificial Intelligence Approach* (pp. 367-403). Palo Alto: Tioga Publishing.

Newell, A. (1990). *Unified Theories of Cognition*. (Ed.). Cambridge, MA: Harvard University Press.

Nichelli, P., J. Grafman, P. Pietrini, D. Alway, J. Carton and R. Miletich (1994). Brain Activity in Chess Playing. *Nature*, 369: 191.

Ratterman, M. and S. L. Epstein (1995). Skilled like a Person: A Comparison of Human and Computational Game Playing. In *Proceedings of the 17th Annual Cognitive Science Conference*, Pittsburgh: Lawrence Erlbaum.

Ratterman, M. and S. L. Epstein (1996). The development of game playing expertise in young children, in preparation.

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

Ungerleider, L. and M. Mishkin. (1982). Two Cortical Visual Systems. In D. Ingle, M. Goodaleet al (Ed.), *Analysis of Visual Behavior* (pp. 548-586). Cambridge: MIT Press.

Yoshio. (1991). *All about Thickness*. (Ed.). Mountain View, CA: Ishi Press.