# The Interaction of Memory and Explicit Concepts in Learning*

**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
sehhc@cunyvm.cuny.edu

## Abstract

The extent to which concepts, memory, and planning are necessary to the simulation of intelligent behavior is a fundamental philosophical issue in AI. An active and productive segment of the research community has taken the position that multiple low-level agents, properly organized, can account for high-level behavior. The empirical research relevant to this debate with fully operational systems has thus far been primarily on mobile robots that do simple tasks. This paper recounts experiments with Hoyle, a system in a cerebral, rather than a physical, domain. The program learns to perform well and quickly, often outpacing its human creators at two-person, perfect information board games. Hoyle demonstrates that a surprising amount of intelligent behavior can be treated as if it were situation-determined, that often planning is unnecessary, and that the memory required to support this learning is minimal. The contribution of this paper is its demonstration of how explicit, rather than implicit, concept representation strengthens a reactive system that learns, and reduces its reliance on memory.

## Introduction

This paper is about the interaction among explicit concept representation, memory requirements, and the ability to learn. *Learning*, in this context, is defined as the transformation of subsequent behavior by previous experience. Learning during problem solving may manifest itself as a change in the speed with which one solves a problem, as a change in the path one takes to a solution, or as a change in the solution at which one arrives.

Although there is general agreement that an intelligent artifact learns, there is less certainty about what is required to learn. Clearly, by the definition of learning, experience is necessary. Most would argue that memory is also necessary for learning, although a machine that rewired itself to incorporate new knowledge, rather than recorded it in some "softer" manner, would meet the criterion. The necessity for concepts, reasoning, and planning in learning, however, has recently come under careful scrutiny by proponents of reactive systems.

The thesis of this paper is that reactive, hierarchical systems can minimize deliberation, but that both memory and explicitly represented concepts are necessary if a program is to learn to perform intelligently. The discussion focuses on a domain previously cited as inhospitable for a reactive system: two-person, perfect information board games (Kirsh, 1991). The paper demonstrates how reactive systems that learn to play games can have unreasonable memory requirements, and discusses concepts and their role in cerebral tasks. Empirical evidence shows how explicit concept representation can reduce memory requirements and improve performance while preserving the essential features of a reactive system: refusal to plan, reluctance to search, and reliance on low-level responses to achieve high-level goals.

## The Control-Concept Controversy

One of the lessons of empirical AI is that general state space search heuristics are weak methods, and that power requires domain specialization. A program may have the appropriate knowledge prespecified or may learn it (Laird, Rosenbloom, & Newell, 1987; Minton, 1988; Mitchell et al., 1989). When search and learning are not enough, many systems *plan*, i.e., reason about possible actions and their outcomes before committing to them.

Biology, however, offers many examples of seemingly intelligent and planned behavior that can be explained as prespecified, i.e., "hard-wired." Ants transporting food cooperatively or young birds avoiding precipices, it is said, do not reason about hunger or danger, although their behavior simulates a creature that does. Some researchers have extrapolated from this to suggest that, in the simulation of intelligence, planning, goals, and representation are unnecessary; that when behavior is cast as reaction to environmental stimulus, only appropriate control is required. Such

* This work was supported in part by NSF 9001936 and PSC-CUNY 668287.

programs are called *reactive systems*.

Brooks has provided the following "representation-free" description of a reactive system: "Low-level simple activities can instill the Creature with reactions to dangerous or important changes in its environment.... By having multiple parallel activities, and by removing the idea of a central representation, there is less chance that any given change in the class of properties enjoyed by the world can cause total collapse of the system.... Each layer of control can be thought of as having its own implicit purpose.... The purpose of the Creature is implicit in its higher-level purposes, goals, or layers." (Brooks, 1991)

Reactive systems are built from small components called *agents*. Each agent has a simple task to accomplish, for example, looking, feeling a force, or moving forward. Each agent "decides" what to do by processing input sensory data. The agent's reaction is its output. The entire program performs as a collection of competing behaviors to which an observer may impute motives and goals where none are ever explicitly represented, i.e., reactive systems do not *deliberate* (plan from concepts).

The coordination of these agents to effect such control is non-trivial. A *layer* is a subsystem of agents that produces an activity, i.e., pursues some implicit purpose. Experiments indicate that a hierarchical *subsumption architecture* that coordinates its agents in layers is the key to proper control for a reactive system (Brooks, 1991; Connell, 1990). One Brooks robot, for example, has a layer to avoid obstacles, another to wander, and one to explore.

The simulation of intelligence in reactive systems is purely a control issue, their proponents claim, without any concern for representation or focus of attention. A few robotic reactive systems have been able to learn their own control strategy (Maes & Brooks, 1990; Mahadevan & Connell, 1991).

Preliminary successes with robots have been predicted to scale up to any task because "there need be no explicit representation of either the world or the intentions of the system to generate intelligent behaviors for a Creature" (Brooks, 1991). Kirsh, however, claims that Brooks has worked only on *situation-determined behavior*, i.e., problems where an egocentric perception of the "indicators that matter" is sufficient to determine the appropriate course of action. (Kirsh, 1991). He characterizes *cerebral tasks*, the kinds of tasks on which he believes a reactive system would fail: tasks that involve other independent agents, that require planning, that require an objective viewpoint, that require problem solving. Between them they pose the *control-concept controversy*: Should a program learn explicit concepts that generalize experience, as in the traditional AI paradigms, or should it learn control for a reactive system? The remainder of this paper explores that issue in a domain Kirsh predicts as too difficult for a reactive system: game-playing.

## Reactive Playing

An obvious reactive system to play a specific game perfectly would construct one agent for each possible game state, an agent that would output the perfect move whenever it sensed a match with its state description. Challenging games, however, would require far too many such agents. Thus, this ideal reactive system must somehow be supplemented with knowledge. The four reactive programs described below are goal-free; all they do is sense patterns and respond to them.

Henri demonstrates how pure pattern recognition can be insufficient for learning even a simple game in a noise-free environment (Painter, 1992). For several different games on a three-by-three board, Henri learns values for three-symbol (X's, O's, and blanks) patterns and applies those values to each of the eight possible three-position lines on the board. Henri learns, for example, that in tic-tac-toe the pattern "X-X-blank" is more valuable than the pattern "blank-X-blank." Values are calculated by a primitive kind of reinforcement learning based on contest outcome. On its turn, Henri evaluates each possible legal move, and selects one with the highest pattern score. Against a programmed expert, after training in 200 tic-tac-toe contests, Henri still loses 15% of the time, because of inaccuracies in the pattern values. It is unclear how long Henri would take to learn to play perfect tic-tac-toe, or if it ever would.

N-N/Tree shows how pattern recognition plus search can still fail to learn a simple game in a realistic environment. This program uses temporal differences to learn weights for a neural net that accepts nine-position pattern input for games on a three-by-three board (Flax et al., 1990). N-N/Tree is also permitted a 3-ply search. It plays against a programmed expert that may err as often as 5% of the time. After 1000 tic-tac-toe training contests, approximately 9000 training examples, N-N/Tree still loses 8% of its contests.

Dooze suggests that learning only control, while adequate, may require more memory than a machine can offer. Dooze is a classifier system that learns to play games on a three-by-three board (Esfahany, 1992). Learning is the introduction and deletion of decision-making rules, called *classifiers*, at the end of each contest. Each classifier has the form "when the board matches the following pattern, move to position i." A pattern describes each of the nine positions as an X, an O, a blank, or a "don't care" symbol. After 63 contests, on average, Dooze learns to play apparently perfect tic-tac-toe. Its better performance may be attributable, however, to its larger memory requirements. There are $9 \cdot 4^8$ possible Dooze classifiers for tic-tac-toe. The program must maintain a set of 150 of them, about 15%, to learn to play expertly. Many of the learned classifiers are quite restrictive, i.e., entail patterns that would apply to very few game states. For five men's morris, a relatively simple game

with 10 markers and 16 positions, 15% of the possible classifiers would be about $2^{31}$ rules.

Morph highlights a possible learning tradeoff between memory size and number of training experiences required to learn. It learns patterns while playing chess against a competent commercial program (Levinson & Snyder, 1991). Morph is characterized as a search-free and purely syntactic game player, i.e., one that reacts only to patterns, without planning or reasoning. A Morph pattern is a labeled graph that describes how selected markers and positions on the board relate to each other. Such a pattern is more sophisticated and less specific than the ones used by Dooze and N-N/Tree, and often applicable to more game states. Given an appropriate, hand-crafted pattern language, Morph's methods can be applied to any game. On tic-tac-toe, a Morph-like program learned to play perfectly after approximately 250 contests and learned approximately 50 patterns (Levinson, 1991). The difference in learning rate and storage requirements between this program and Dooze suggests a trade off between memory size and number of training experiences required to learn.

Careful analysis reveals that each of these "representation-free" programs actually incorporates concepts, generalizations about game playing understood by the programmer and incorporated into the code. Henri only uses knowledge about lines and how positions lie on them in two-dimensional space; its performance is also the weakest. N-N/Tree uses knowledge about the minimax algorithm for search control and how to apply it three-ply deep. Dooze's learning algorithms value the winning move highly, value every move the expert model makes, and recognize that good positions for X are good for O when the markers are interchanged. This is a hefty dose of primitive game-playing commonsense. Dooze's don't-care symbols also support abstractions, such as "If X holds the center, ...." Morph's pattern language embeds ideas like threat and defense in both the pattern learner and in memory. In summary, although reactive game players are possible, they rely on hidden knowledge to achieve acceptable performance, and probably have some tradeoff between memory size and learning speed.

## Concepts and their Representation

A *concept* is defined here as some recognized set of regularities detected in some observed world. *Regularity* means repeated occurrence and/or consistency of use. In this context, a concept includes not only the necessary and sufficient descriptions called definitions, but also defaults, associations, and expectations. Thus a concept may incorporate error, bias, and inconsistency (Wierzbicka, 1985). From an AI perspective, a concept is generalized domain knowledge, a description of what has been encountered. Although specific examples may be remembered, a concept is not a set of instances but a summary of experience.

If a machine is constructed to meet a goal, either implicit or explicit concept representation is necessary. A cherry pitter, for example, implicitly references the concept of a cherry as a small, round object containing an even smaller object which can be extracted when pressure is appropriately applied. Although the architecture of a sufficiently elaborate machine, like a robot, may obscure its concepts, they are present implicitly, in circuitry and mechanical devices. Any program claimed "representation-free" is characterized here as a program with implicit concept representation. In contrast, explicit concept representation offers several benefits to a machine that learns: organization of knowledge, focus of attention, and ability to discard experience. Thus explicit concept representation reduces the need for induction and deduction, as it flexibly makes regularities immediately accessible.

People find it convenient to expect regularity in the world, and they have many devices to represent the regularities they detect. Four kinds of *concepts*, ways that people generalize about regularities in their experience, are identified here. *Compiled* regularities, like how to ride a bicycle, abbreviate a reliable response to specific situations. Compiled knowledge is experienced as reactive behavior; it has lost the detail, rationale, and instructions that once accompanied it. When the lost information is needed, reconstruction often requires observation from fresh experience. *Categories* are sets of objects with common features. For example, a chair is a category and every chair, physical or hypothetical, is an *instance* of the category, with specifically noted values for some of its features. *Scripts* are regularities about what is expected of an experience and those who participate in it (Schank & Abelson, 1977). For example, any visit to a restaurant is a walk through a script, a partially ordered set of expectations for everyone's behavior there. *Meta-principles* are regularities applicable to many different kinds of experience, ones to fall back upon when more detailed knowledge fails. Examples of meta-principles include efficiency, safety, and propriety. The instantiation of a meta-principle for a particular domain results in a *principle*, behavioral guidance that may curtail search. The application of efficiency, safety, and propriety to driving a car, for example, would result in directives to drive rapidly, to drive carefully, and to obey the driving laws, respectively. Note the evident conflict among these principles.

People behave appropriately and learn quickly in part because they retrieve and apply these regularities, or concepts, continually and effectively. There is ample psychological and anthropological evidence that concepts are both learned and culturally determined, and that people prefer them to logical reasoning for any but the simplest examples (D'Andrade, 1991). In any culture, those judged experts are those who give more

modal responses, i.e., agree most with commonly held regularities (D'Andrade, 1990). Thus *an expert learns compiled knowledge, categories, scripts, and principles, and knows when and how to apply them.* Given those regularities, learning and problem solving with them may not be trivial, but it should be easier and require less memory.

## The Power of Concepts and Memory

Hoyle is a learning program that now equals or outperforms its human mentors at more than a dozen two-person, perfect information board games. The complexity of the program prevents a full technical description in this abbreviated space; interested readers are referred to (Epstein, 1992a, 1992b) for additional detail. Hoyle explicitly represents, integrates, and exploits each of the four kinds of concepts in its memory, learning, and behavior. There is a script for game playing that provides predefined, uniform, procedural direction, so that the system performs as if it were accustomed to playing games. There is a category representation for games and another for *useful knowledge* (knowledge that is possibly relevant and probably correct) that may be acquired during play. Hoyle's compiled knowledge resides in its *Learner*, as pre-specified, uniform, game-independent heuristic procedures to compute and selectively store useful knowledge. Finally, Hoyle's *Advisors* are principles, implemented as heuristic agents and layered in a subsumption architecture. They accept current knowledge and make comments on moves they favor or oppose.

Given a game, the Learner initiates a series of tournaments against an *expert model* (Kirsh's "other agent") that is only observed, never queried. Whenever it is Hoyle's turn to move, the Advisors comment based upon the current state of their cerebral reality: the game state, the legal moves, and any useful knowledge about the game already acquired. Move selection is a simple arithmetic calculation, part ordering and part voting, that mediates among the Advisors' disagreeing comments. After contests and after tournaments, the Learner's algorithms compute and record useful knowledge.

Hoyle is a reactive system for a cerebral task. The Hoyle cycle is pause-sense-react, where "pause" cedes control to the expert model, "sense" is the collection of current information, and "react" is the collective response of the agents to their input. Each Advisor is an agent, a low-level intelligence that does not plan, that merely senses the input data and responds to it with output signals. The control mechanism is based upon a hierarchical subsumption architecture. *Each move choice is a rapid and simplistic mathematical computation, a reaction without search or deliberation.*

Hoyle meets the postulated reactive system criteria as follows. The low-level simple activities are its quick reactions to short-term possibilities of success and failure. The multiple parallel activities are its Advisors, each of which processes sensory data independently. When Hoyle's world changes, with a new game to play or new opposition to play against, the program is robust and degrades gracefully. The implicit purpose of each Advisor is to forward, in its own particular way, the meta-principle it instantiates. Hoyle can play legally with any subset of its Advisors. The purpose implicit in its higher-level layers is to learn to play perfectly, but there is no explicit representation, in Hoyle's game-playing algorithm or in its control mechanism, of intention, belief, plan, goal, subgoal, win, loss, or draw.

Figure 1 shows performance curves for several 20-contest tic-tac-toe tournaments. The bottom line (#1) is a reasonable lower bound for performance; it shows how a program lost all but one contest in 20 when it made random legal moves against a programmed expert. The top line in Figure 1, for absolute expertise, is a reasonable upper bound; it shows how a program that made perfect moves achieved repeated draws against a programmed expert. Once a program learns to play perfectly, its performance curve should parallel that for absolute expertise indefinitely.

Hoyle's useful knowledge is a compendium of the regularities expert game players look for and exploit. A *significant state* is an inevitable win or loss when both participants play expertly. Such a state is a deduced, compiled regularity computed by the Learner at the end of a contest and stored in memory. The regularity captured by a significant state is that *every* time it occurs the outcome when two experts play is inevitable, not that it is an abstraction of a game state. A significant state may either be treated as a concept (used in computation) or treated as a reflex action (turned toward or avoided). Retrieval of significant states is from a hash table, and is assumed to require no search. Besides significant states, useful knowledge includes selected contest histories, moves experts have made that may have served them well, whether or not it is an advantage to go first, the length of the average contest, data gathered on the relevance and reliability of individual Advisors, and relevant *forks*, game-independent concepts whose instantiation with the current game state can provide powerful offensive and defensive advice (Epstein, 1990). The memory requirement

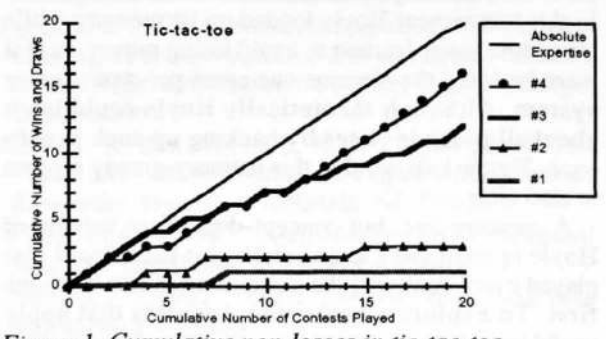


*Figure 1. Cumulative non-losses in tic-tac-toe.*

for learning a game is essentially a function of the number of significant states and expert moves.

At present Hoyle plays correctly (has the rules for) 24 games gathered from almost as many cultures (Bell, 1969; Zaslavsky, 1982). Hoyle's task is to learn to play each game expertly. Although none of the games is as difficult as checkers or chess, they incorporate a variety of challenges: boards of varying shapes and sizes, stages where the rules change, cycles, and very large search spaces. One of the games it learns to play expertly is Qubic, which has more than a billion game states and is generally acknowledged to lie on the border between simple games and the difficult ones.

The performance of the full version of Hoyle at tic-tac-toe against a programmed expert is shown as curve #4 in Figure 1. In 11 contests, Hoyle learned to play perfectly, and stored an average of 3 significant states and 4 expert moves. Compare this with Dooze's 63 contests and 150 rules, and the Morph-like 250 contests and 50 patterns.

Are Hoyle's concepts or its memory responsible for its ability? Elimination of all concepts from Hoyle would deprive it of its game-playing algorithm, and make it unable to play at all. Less radically, if all the Advisors and all learning were removed, Hoyle would make random moves and would play no better than curve #1 in Figure 1. An interesting reactive version of Hoyle with severe concept restrictions learns responses to game states but uses them only in one way. This *concept-poor* version of Hoyle is a flawless imitator; the Advisors react to the input knowledge but do not perform simple calculations from past experience. This version restricts learned useful knowledge to detailed recollection of the contests it has played and of significant states as reflexes, not as concepts; that way it is permitted only performance repetition, rather than simple computation, with its knowledge. The concept-poor version recognizes previously encountered certain wins and losses, imitates moves the expert made in the identical situation, and tries to avoid reproducing its own failing moves in an identical situation. The performance of this concept-poor version against a programmed expert is shown as curve #2 in Figure 1. The partially-disabled Advisors immediately learn and recommend the successful openings of the human participant, but find the play later in a contest more difficult. In this tournament Hoyle loaded up its memory while it very gradually learned to avoid losing moves, as if it were building the obvious one-agent-per-state reactive system. Although theoretically Hoyle could learn about all possible states by backing up such experience, Figure 1 shows that this memory-greedy process is also slow.

A *memory-free* but concept-dependent version of Hoyle is analogous to an intelligent participant that played every contest at the same game as if it were the first. To explore whether the Advisors that apply useful knowledge really need memory to learn to play expertly, Hoyle played a tournament against a human expert (#3 in Figure 1), this time only with those Advisors omitted from the concept-poor version and without memory. Now the program could rely only on its concepts. After a few contests, a human opponent unaware that Hoyle lacked memory tried a simple strategy for X that defeated the program, one the memory-free version could not learn to avoid. On a hunch the person repeated the same strategy and immediately observed that Hoyle did not learn from its mistakes. (This accounts for the step-like pattern of #3; Hoyle played perfectly in the alternate contests.) Thus the program without memory plays reasonably intelligent contests, but performs unintelligently in a tournament situation. Learning requires memory, and without memory Hoyle never would develop expertise. In all of the games, except the very easiest, it has been repeatedly observed that *Hoyle's power derives from this synergy between memory and concepts.*

Thus far, Hoyle has learned to play as well or better than each of its 14 game-specific external experts, without planning and with only minimal search. When Hoyle has had difficulty learning a new game, its useful knowledge has been very gradually extended to include new concepts and the low-level agents to apply them. This gradual debugging process is much like Brooks' robot-control layering: "so far, so good" (Brooks, 1991).

## Conclusions

In a domain that is not situation-determined, Hoyle is a successful, reactive, hierarchical system that retains only a small fraction of what it experiences. The program pays an interesting price for its reactivity, however: it must rely on concepts to learn to perform intelligently. Hoyle offers evidence that learning cerebral tasks demands more explicit concepts than Brooks would like, and far fewer than Kirsh would assume. Hoyle may not resolve the control-concept controversy, but it should certainly influence our attitude on the significance of low-level agents in high-level tasks.

Four other reactive game playing programs have been shown here to employ concepts implicitly. Their pattern generalizations, however, are tailored to a single set of board-specific algorithms, and their memory requirements grow dramatically with the number of positions on the board. Hoyle outperforms these programs, this paper has argued, because it explicitly represents and exploits its concepts.

Hoyle's concepts organize the way it remembers experience, focus its attention on what is important to learn, force it to apply its experience, and permit it to discard experience that is judged unlikely to be useful. As a result it learns with smaller memory requirements and applies its compact useful knowledge more flexibly. Although Hoyle is reactive, the full version

of the program incorporates and remembers concepts: knowledge about the regularities that people learn, prefer, and exploit when playing games, and how people use those regularities. When the program is partially disabled and the results observed, it is clear that the synergy between memory and concept application provides the program with its power.

Hoyle's ability to learn with only 15 relatively simple Advisors suggests that *more high-level behavior is available through low-level reactive processes than one might initially suspect.* As the games become more difficult, new concepts are necessary to support performance. *Learning high-level behavior efficiently with a limited memory requires concepts.* After a recent improvement that provided symmetry discovery, Hoyle learned faster and required less memory. *Low-level sensory data can offer an immediate improvement in high-level processing.*

For the time being, several tasks have been relegated to the human system designer: the framework of the categories for game definition and useful knowledge, the correct identification of the culturally determined meta-principles (characterized as "commonsense" but by no means trivial), the instantiation of the meta-principles to construct low-level agents, the assignment of Advisors to tiers based upon knowledge about relations among meta-principles, the specification of which Advisors access which concepts, and the description of how they apply that knowledge. This author believes that all of these can eventually be automated. Work continues on the specified sequence of games; for the moment search during play is limited to two-ply and there is no planning.

Hoyle's results demonstrate for at least one broad cerebral task, game playing, that *a reactive system without memory is impractical, and that reliance only on extensive, detailed memory is brittle and often impossible.* This paper has shown how concepts can structure resource-efficient memory, provide flexibility, and regularize knowledge to support performance. Will a reactive program ever, then, have to search and plan and believe? Hoyle's answer is not yet, perhaps not explicitly, and far less than we ever expected.

**Acknowledgments**

The author thanks Jack Gelfand, Alice Greenwood, Cullen Shaeffer, and Rick Shweder for their insightful comments and suggestions.

## References

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

Brooks, R.A. 1991. Intelligence without Representation. *Artificial Intelligence* 47: 139-160.

Connell, J. 1990. *Minimalist Mobile Robotics.* New York: Academic Press.

D'Andrade, R.G. 1990. Some Propositions about the Relations between Culture and Human Cognition. In *Cultural Psychology,* ed. J.W. Stigler, R.A. Shweder & G. Herdt. Cambridge: Cambridge University Press.

D'Andrade, R.G. 1991. Culturally Based Reasoning. In *Cognition and Social Worlds,* ed. A. Gellatly and D. Rogers. Oxford: Clarendon Press.

Epstein, S.L. 1990. Learning Plans for Competitive Domains. Proc. 7th International Conference on Machine Learning, 190-197. Morgan Kaufmann.

Epstein, S.L. 1992a. Hard Questions about Easy Tasks. In *Computational Learning Theory and Natural Learning Systems: Constraints and Prospects,* Cambridge, MA: MIT Press. Forthcoming.

Epstein, S.L. 1992b. Prior Knowledge Strengthens Learning to Control Search in Weak Theory Domains. *International Journal of Intelligent Systems.* Forthcoming.

Esfahany, K. 1992. A Pattern Classifier that Learns to Play Games. In preparation.

Flax, M.G., Gelfand, J.J., Lane, S.H. & Handelman, D.A. 1990. Integrating Neural Network and Tree Search Approaches to Produce an Auto-Supervised System that Learns to Play Games. In Proceedings of the 1992 International Joint Conference on Neural Networks, Beijing. Forthcoming.

Kirsh, D. 1991. Today, the Earwig, Tomorrow Man? *Artificial Intelligence* 47: 161-184.

Laird, J. E., Rosenbloom, P. S. and Newell, A. 1987. SOAR: An Architecture for General Intelligence. *Artificial Intelligence* 33: 1-64.

Levinson, R. 1991. Personal communication.

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

Maes, P. and Brooks, R.A. 1990. Learning to Coordinate Behaviors. In Proceedings of the Eighth National Conference on AI, 796-802. AAAI Press.

Mahadevan, S. and Connell, J. 1991. Scaling Reinforcement Learning to Robotics by Exploiting the Subsumption Architecture. In Proceedings of the Eighth International Machine Learning Workshop, 328-332. San Mateo: Morgan Kaufmann.

Minton, S. 1988. *Learning Search Control Knowledge.* Boston: Kluwer Academic.

Mitchell, T., et al. 1990. Theo: A Framework for Self-Improving Systems. In *Architectures for Intelligence,* ed. K. Vanlehn. Boston: Erlbaum.

Painter, J. 1992. Pattern Recognition for Decision Making in a Competitive Environment. Master's thesis, Dept. of Computer Science, Hunter College. In preparation.

Schank, R. and Abelson, R. 1977. *Scripts, Plans, Goals, and Understanding: An Inquiry into Human Knowledge Structures.* Hillsdale, NJ: Erlbaum.

Wierzbicka, A. 1985. *Lexicography and Conceptual Analysis.* Ann Arbor, MI: Karoma Publishers.

Zaslavsky, C. 1982. *Tic Tac Toe and Other Three-in-a-Row Games.* Crowell.