Panel:
The Role of Chess in Artificial Intelligence Research

Robert Levinson (Chairperson)
Computer and Information Sciences
Applied Sciences Building
University of California at Santa Cruz
Santa Cruz, CA 95064
(408)459-2087
ARPANET:levinson@cis.ucsc,edu
FAX:429-0146

Feng-hsiung Hsu
IBM T.J. Watson Research Center
PO Box 704
York town Heights
NY 10598

Jonathan Schaeffer
Computing Science
University of Alberta
Edmonton
Canada T6G 2H1

T. Anthony Marsland
Computing Science
University of Alberta
Edmonton
Canada T6G 2H1

David E* Wilkins
SRI International EJ227
333 Ravenswood Ave
Menlo Park
CA 94025

Abstract

Our eminent researchers including John McCarthy, Allen Newell, Claude Shannon, Herb Simon, Ken Thompson and Alan Turing put significant effort into computer chess research. Now that computers have reached the grandmaster level, and are beginning to vie for the World Championship, the AI community should pause to evaluate the significance of chess in the evolving objectives of AI, evaluate the contributions made to date, and assess what can be expected in the future. Despite the general interest in chess amongst computer scientists and the significant progress in the last twenty years, there seems to be a Jack of appreciation for the field in the AI community. On one hand this is the fruit of success (brute force works, why study anything else?), but also the result of a focus on performance above all else in the chess community. Also, chess has proved to be too challenging for many of the AI techniques that have been thrown at it. We wish to promote chess as the fundamental test bed recognized by our founding researchers and increase awareness of its contribution to date.

Panel Summary

The factors that make chess an excellent domain for AI research include:

- Richness of the problem-solving domain.
- Ability to monitor and record progress accurately through competition and rating, because of its well-defined structure.
- Chess has been around for centuries - the basics are well-understood internationally, expertise is readily available and is (generally!) beyond proprietary or nationalistic interests. Has been considered a "game of intelligence." Many players of the game feel mentally "stretched."
- Detailed psychological studies of chess playing exist. These studies suggest that human players use different reasoning modes from those in current chess programs. Further, the reasoning modes are also used in many other problem-solving domains.
- Excellent test bed for uncertainty management schemes - the basis of most expert problem-solving. The well-definedness and discreteness of the game have led many to ignore this.

The above factors make chess a useful tool regardless of the strength of the current programs. Because of the success of the current methods there remains a vast arena of other methods that have not been explored. The most obvious lack is in the application and development of machine learning techniques to chess, but other areas, including knowledge representation and compilation, planning and control, also seem to be applicable. AI researchers should be encouraged to use chess as a test bed for their techniques, with the understanding that chess is not the end in itself. Chess may provide the avenue by which bridges may be built between cognitive science, AI and connectionist modeling-

Wit h the current and future battle for the World Human-Computer Championship the AI community should be made more sensitive to the issues involved and their bearing on intelligence research: Is search sufficient? How much detailed chess knowledge is required? How is this knowledge implemented and incorporated with search? We are fortunate to have a World Champion who promotes creativity over the chess board and is willing to face the challenge from computers head-on.

The members of the panel and the presentations have been designed to address these topics in a way that supports our objectives to make chess an important and respected AI tool in this new decade. *Jonathan Schaeffer* will emphasize those areas of computer chess research that have been ignored, because the approach has been a competitive/engineering one instead of scientific. *Feng-hsiung Esu* of the Deep Thought team will discuss the role of knowledge in current chess programming and argue that more responsibility for the knowledge should be put on the machines themselves, *Tony Marsland* will present specific open research issues in computer chess that will require AI solutions. *Robert Levinson* will describe an alternative model of chess computation, a self-learning pattern-oriented chess program ("Morph") whose knowledge is learned incrementally from experience, without many examples being stored (and with little guidance about relevant features). *David Wilkins* will provide balance to the discussion by pointing out the limitations of chess and claiming that Go is a better domain. He will also describe a new type of games tournament that prevents the human tailoring of evaluation functions and encourages the use of learning and more robust approaches.

The timing for this panel is particularly good with the current World Championship having completed, a more powerful Deep Thought on the scene, a recent article in Scientific American [Hsu *et al,* 1990] and new books by Levy and Newborn [1991], and by Marsland and Schaeffer [1990].

Presentations

Computer Chess: Science or Engineering?

Jonathan Schaeffer
University of Alberta

Research into artificial intelligence using chess as the application domain has produced several important contributions to AI:

- The effectiveness of brute-force search. Chess has clearly demonstrated that simple, brute-force approaches should not be quickly discarded.
- Iterative search. Some of the ideas developed for alpha-beta search, iterative deepening in particular, are applicable to other search domains.
- The inadequacy of conventional AI techniques for realtime computation. No competitive computer chess program uses AI languages or knowledge representation methods. Why? They are too slow for a real-time, high performance application.

Although these (and other, lesser contributions) have enhanced our knowledge, it is not clear whether the effort expended justifies the results obtained.

It is easy to question the usefulness of computer chess research. It is important to distinguish between computer chess research and research using chess as a test bed. Unfortunately, the latter has evolved into the former. An entirely new field of "computer chess" has evolved, with the emphasis on chess performance and chess research - not generally of much interest to the AI community. There is a much deserved credibility problem here. The unfortunate correlation between program speed and performance encourages short-term projects (speeding up a move generator 10%) at the sacrifice of long-term research projects (such as chess programs that learn).

After over 30 years of work on chess programs, where are the scientific advances in:

- knowledge-based search algorithms? There has been some good work in this area, but none has progressed enough to be used in competitive chess programs. Alpha-beta simplifies the programming task, but the exponential search limits what can be achieved.
- knowledge representation and acquisition? These areas are of considerable importance to chess programs, yet the computer chess community has done embarrassing little research in this area.
- error analysis? While extensive error analysis has been done on search algorithms, little has been done to quantify errors in evaluation functions and how they interact with the search.
- tool development? With the right tool, work that might take days could be done in minutes. No tools are being developed to help build chess programs. For example, why isn't someone working on tools for defining chess knowledge?

If the community were committed to research, many of these problems would have been addressed by now. Sadly, much of the work currently being done on computer chess programs is engineering, not science. For example, the engineering of special-purpose VLSI chips to increase the speed of a chess program only underlines the importance chess programmers attach to speed.

In my opinion, conventional computer-chess methods will yield little of further interest to the AI community. I believe they will be inadequate to defeat the human World Champion in a match for a long time to come. It is still very easy to set up a position for which the computer has no idea what is going on - even if you

speed up the machine 1000-fold. The current computer chess work will only underscore the need for better ways of adding and manipulating knowledge reliably.

The defeat of the human World Chess Champion sooner rather than later will help artificial intelligence. This will help to re-establish chess as an ideal problem domain for experimenting with the fundamental problems of artificial intelligence, as elaborated more fully by Donskoy and Schaeffer [1989].

"Expert Inputs" are Sometimes Harmful

Feng-hsiung Hsu
IBM T J. Watson Research Center

Experience from the chess machine Deep Thought suggests that inputs from chess experts, while generally useful, cannot be trusted completely. A good example of this is Deep Thought's evaluation function. Several changes by capable human chess experts failed to produce significant improvements and occasionally even affected the machine's performance negatively. Here, human experts, along with their expertise, introduced their own prejudices into the program. One way of solving this problem is to limit the type and the amount of expert inputs allowed into the program; in other words, having an almost "knowledge-free" machine. The availability of on-line high quality chess game databases makes this an attractive approach. Instead of having the value of, say, an isolated pawn set by human experts either explicitly or in functional form, one can simply tell the program that isolated pawns are important features and statistical procedures, with some additional expert inputs, can then be used to decide the functional form and the proper weighting of the features in question.

That more responsibility for knowledge should be placed on machines is consistent with recent efforts to handle the knowledge acquisition problem in expert systems and also in memory-based reasoning schemes where knowledge is generated statistically rather than relying on symbolic learning, abstraction or domain modeling.

Open Problems and Lessons for AI

T. Anthony Marsland
University of Alberta

Based on predicted advances in computer technology, particularly the faster speeds and increasing memory of low cost systems, it is reasonable to assume that within the next decade the World Chess Champion will lose an informal game to a computer, and within twenty-five years lose a 12-game match. The early losses will reflect more breaks in concentration at first and later a recognition of the inevitable, as arose when trains started to out-pace runners. Although the defeat of humans by machines will be significant, it will mean neither the destruction of chess as a pastime and learning medium, nor the end of interest in computer chess *per se.* Instead it will focus attention even more sharply on precisely how and why humans can become so expert at selecting sound (often optimal) variations in seemingly complex situations, without resorting to the exhaustive techniques used by computer programs.

Some fundamental AI questions that will remain are:

- Given a patient and seemingly perfect teacher (that is a superior chess-playing machine), how should one use it to "teach" an AI-based learning program about strategies for playing chess (given that the rules of chess themselves are already perfectly known)?
- A related but perhaps simpler problem comes from the realm of endgame play. Given a perfect N-piece database holding an optimal move for each position (or perhaps only the length of the optimal sequence from that position, or even less, whether the position was won), develop a program that can deduce a sound set of rules or strategies for playing the endgame perfectly (or at lea#t better than any other expert).
- Given endgame positions which cannot be solved by search or databases alone, deduce a plan or playing strategy that will transform the position into a known (win/draw) state. In one class of positions the remaining pieces are held to few squares. Progress can only be made by a freeing move that converts a short-term loss of material (or perhaps position) to the achievement of a later, more significant goal, Consider the Duchess-Chaos game [Frey, 1983, pp. 269-274], which is still thought to be beyond brute force search. Related examples abound, for instance giving up a passed pawn on one side of the board to win a pawn race on the other.
- Given a well-defined threat (for example mate) deep in the tree, identify un-examined moves at an earlier level along the current path that have the potential to deny the threat explicitly. This a form of dynamic re-ordering of moves, but is also (if no potential denials exist) a good forward pruning criteria - providing evidence to abandon this line of play.

The first two projects rely on perfect domain knowledge and the availability of an un-tiring teacher to whom questions can be posed. The need for convergence to a solution within some arbitrary or unreasonably short time-frame will thus be eliminated. The learning mechanisms used will have the benefit of drawing on results obtained by exhaustive means. Even simple rote learning, for example, has its place in Artificial Intelligence. Scherzer *et at.* [1990] have shown how to use the moves made during a series of chess games to ensure that a poor move sequence will eventually not be replayed. By holding the computer's memory of played games in a hash table the information can be used to extend the depth of search during later play. In related work, the expert knowledge from an encyclopedia of games has been optically read into a computer which also corrected the typographical computer to play through all known games, identify "losing moves" and by backtracking develop "innovations*' that correct the flaws. It is not trivial work because it requires making a plausible re-construction from imperfect data, but Ken Thompson has shown the way here [Marsland, 1987] Finally, once errors are found, a backtracking mechanism will be needed to find the best place earlier in the game-tree path to correct (avoid) the flaw that follows. The approach of recording, correcting and innovating appears to be fundamental to Artificial Intelligence.

The remaining problems are linked closely to formal or probabilistic pruning methods. Computer chess is computationally expensive enough that one can afford to expend considerable time eliminating parts of the search space by deduction. Over the years humans have developed techniques that allow them to reduce the search space through judicious use of forward pruning (that is, by temporarily abandoning certain variations) and either deducing by analogy that further consideration would be irrelevant, or (upon questioning the validity of the pruning) force reconsideration of the omitted lines. Pruning by analogy is a powerful general-purpose too! and if developed satisfactorily for a perfect information game like chess would almost certainly be applicable to related decision-tree searches. The intent is to be more selective about variations that are to be expanded fully. Methods like the null-move heuristic [Beal, 1989], conspiracy number search and singular extensions [Anantharaman *et* al., 1988] all do this by expanding non-quiescent lines of play. On the other hand, the more formal probabilistic methods [Palay, 1985] attempt to limit the width of search at any node by estimating the probability that a better move exists in the moves that remain to be searched (Kozdrowicki and Cooper's [1973] so called "Fischer Set"). In effect this problem requires looking again at the method of analogies [Adelson-Velsky *et* al,, 1975]- It is remarkable that no significant improvement has been made to that method, despite the passage of 15 years. Not even attempts to implement simple forms of the idea in serious chess programs. The fundamental work here is to determine how best to make the method of analogies pay for itself. In this era of faster processors and parallel computation this must be a topic that is ripe for xploitation.

One important lesson for the AI community is the importance of competitive testing and performance comparison of algorithms. In a sense 20 years of computer chess championships have provided a long-running series of experiments proving conclusively that progress has been made, identifying clearly those methods that have been effective and making a direct comparison from year to year possible. In principle theorem proving programs could be tested the same way, as indeed could language translation systems. These forms of comparison are standard for pattern recognition systems, why not for natural language understanding? In conclusion AI would benefit if more of its work were done on a direct competitive basis to identify more sharply those methods that are truly generally applicable.

Morph: An Adaptive, Pat tern-Oriented Chess System

Robert Levinson
University of California

Although chess computers now are competitive at master and grandmaster levels, that is where their resemblance to human players ends. Psychological evidence indicates that human chess players search very few positions, and base their positional assessments on structural/perceptual patterns learned through experience. Morph is a computer chess program that has been developed to be more consistent with the cognitive models.

The main objectives of the project are to demonstrate capacity of the system to learn, to deepen our understanding of the interaction of knowledge and search, and to build bridges in this area between AI and cognitive science.

The current model of chess programming came into its own in the 70's and early 80's and has been refined ever since [Slate and Atkin, 1977]. The main characteristic of the model is the use of brute-force alpha-beta minimax search with selective extensions for special situations such as forcing variations. This has been further enhanced by special purpose hardware. This model has been so successful that little else has been tried.

The alternative AI approaches have not fared well, perhaps because of the expense in applying the "knowledge" that had been supplied to the system. When chess has been used as a test bed [Flann and Dietterich,. 1989; Minton, 1984; Pitrat, 1976; Quinlan, 1983; Wilkins, 19821 only a small sub-domain of the game was used, so that fundamental efficiency issues that AI must grapple with have been largely unaddressed. However, we feel that there is a third approach that relies neither on search nor on the symbolic computation approach of knowledge-oriented AI. In what we shall call the "pattern-oriented approach," configurations of interaction between squares and pieces are stored along with their significance. A uniform (and hence efficient method) is used to combine the significance in a given position to reach a final evaluation for that position. That such an approach is possible is evidenced by psychological models of human chess play [de Groot, 1965; Pfleger and Treppner, 1987].

Morph[1] is a system developed over the past 3 years that implements the pattern oriented approach [Levinson, 1989; Levinson and Snyder, 1991]. It is not conceivable that the detailed knowledge required to evaluate positions in this way could be supplied directly to the system, thus learning is required.

To strengthen the connections with the cognitive literature the system's knowledge is to come from its own playing experience, no sets of pre-classified examples are given and beyond its chess pattern representation scheme little chess knowledge such as the fact that having pieces is valuable (leave alone their values) has been provided to the system. Further, the system is limited to using only 1-ply of search.[2]

System Design Morph makes a move by generating all legal successors of the current position, evaluating each position using the current pattern database and choosing the position that is considered least favorable to the opponent. After each game patterns are created, deleted and generalized and weights are changed to make its evaluations more accurate, based on the outcome of the game. A more detailed summary of the design has recently appeared [Levinson and Syn-

The name "Morph" comes from the Greek *morph* meaning form and the chess great, Paul Morphy.

[2]Though nothing in the method except perhaps efficiency, prevents deeper search.

der, 1991].

Morph stores two types of pattern: Graph patterns which represent attacks and defends relationships between pieces and squares and Material patterns that are vectors giving the relative material difference between the players, e.g. "up 2 pawns and down 1 rook," "even material," and so on. Along with each pattern is stored a weight that reflects the significance of the pattern. The weight is a real number in [0,1] that is an estimate of the expected true minmax evaluation of states that satisfy the pattern.

Results with Morph There have been many encouraging signs in the three months since Morph was fully implemented, and some preliminary results have been published [Levinson and Snyder, 1991].

Relationship of Morph to other approaches The chess system combines threads of a variety of machine-learning techniques that have been successful in other settings. It is this combination, and exactly what is done to achieve it, that is the basis for Morph's contributions. The learning-method areas and their involvement in Morph include genetic algorithms [Goldberg, 1989], neural nets (weight updating), temporal-difference learning, explanation-based generalization (EBG), and similarity-based learning. To combine these methods some design constraints usually associated with these methods are relaxed. With genetic algorithms, structured patterns rather than bit strings are used. In contrast to neural networks the nodes in Morph's hierarchy are assigned particular semantic/structural values. Temporal-difference learning is usually applied to methods with fixed evaluation function forms (in which the features are known but not their weights) but here the features change and the hierarchical database organisation produce atypical discontinuities in the function.

Once a chess graph is constructed from a game board, the semantics of the nodes and edges in the graphs are unknown to the system. The only information a pattern contains as far as the system is concerned is the significance (weight) that has been attached to the pattern , in no place after pattern creation *do* we special case pieces or edges. Such a syntactic approach to the learning of search knowledge is substantially different from many of the traditional symbolic AI approaches to chess and to the learning of control knowledge.

In summary, in addition to a unique combination of methods, what distinguishes Morph is:

- A uniform representation of search knowledge.
- A syntactic approach to playing and learning.
- An attempt to play a complete game of chess rather than a small subdomain.
- Rejection of a learning-by-examples framework for an experiential framework that is more cognitively-inspired,
- Responsibility for feature discovery given to the system.
- Non-reliance on search (though at some point small guided search may be incorporated, bringing us even closer to the cognitive model).

Chess Was Good for AI Research.

David E. Wilkins
SRI International

Over the years, chess has proven to be a fertile ground for ideas and techniques that have spread to other areas of AI. These include database enumeration techniques [Bratko, 1978], chunking [Campbell, 1988], search techniques (minimax, alpha-beta, iterative deepening), and the utility of information [Good, 1977], Considering the lack of funding for chess, it is significant that it has produced so many results.

Chess has been fertile because it provides a complex reasoning problem from a simple domain with a built-in performance criteria. The simple domain permits research to progress with little initial overhead. Having a hostile opponent adds complexity to the reasoning. In many domains (natural language understanding comes to mind), progress can be hindered by lack of performance criteria - it can be hard to tell whether the latest thesis is an improvement on the current state of the art, Chess provides precise answers to performance questions.

However, hardware advances have made chess a less fertile ground for addressing the basic issues of AI. The game is small enough that brute-force search techniques have dominated competitive computer chess, and I see little AI interest in squeezing out the last few hundred points on the chess ratings, except for the psychological impact of having a computer beat the human world champion.

Obviously, many basic issues in AI are not naturally addressed in a game-playing environment and should be explored in other domains. These include communication, forming models of one's environment, sensor analysis and integration, and (perhaps) reasoning about uncertainty. In addition, real-world domains force AI researchers to address issues such as economy of scale, noise, realtime response, failed actions, novel phenomena, and multiple agents - issues that can be ignored in chess.

Of the AI areas well-suited to a game-playing domain, there are better domains than chess. In particular, Go has all the advantages of chess but provides more complex reasoning and an even simpler domain. A successful symbolic Go program would have to plan, would have to use goal-directed search, would encourage machine learning, and would promote visual reasoning - all basic AI research issues that are now ignored in competitive computer chess.

Even better than Go, Barney Pell [1991] has proposed an event where programs compete against each other, but are only given a description of the game to be played at the beginning of the match. Chess is particularly well suited to this adaptation. One could, for example, have a competition using a chess board and pieces, where the match begins by giving the programs a declarative statement of now the pieces move, how they capture, the initial position, and what the objective of the game is. The programs would have to play this newly defined game under time constraints. A longish series of games could be required. This would require machine learning and a robust symbolic problem-solving capa-

bility that is not tailored to a specific game. For each new game, the programs would have to learn evaluation functions (if needed), learn what goals are advantageous to attempt, and learn heuristics or features for selecting moves. Brute force techniques would be disadvantaged by the lack of opportunity to fine tune both an evaluation function and a quiescence search for the game at hand.

References

[Adelson-Velsky *et al,* 1975] G. Adelson-Velsky, V. Arl&z&rov, and M. Donsky. Some methods of controlling the tree search in chess programs. *Artificial Intelligence,* 6(4):361-371, 1975,

[Anantharaman *et ah,* 1988] T, Anantharaman, M. S. Campbell, and F- H, Hsu. Singular extensions: Adding selectivity to brute force search. *Intern. Computer Chess Assoc. Journal* 11 (4): 135-143, 1988.

[Beal, 1989] D. Beal. Experiments with the null move. In *Advances in Computer Chess 5,* pages 65-79. Elsevier, 1989.

[Bratko *et al.,* 1978] I. Bratko, D. Kopek, and D- Michie. Pattern-based representation of chess end-game knowledge. *Computer Journal,* 21(2):149-153, May 1978.

[Campbell, 1988) M.S.Campbell. *Chunking as an Abstraction Mechanism.* PhD thesis, Carnegie Mellon University, 1988,

[de Groot, 1965] A- D. de Groot. *Thought and Choice in Chess,* The Hague, 1965.

[Donskoy and Schaeffer, 1989] M. Donskoy and J. Schaeffer, Perspectives on falling from grace. *Intern. Computer Chess Assoc. Journal,* 12(3): 155-163, 1989. Reprinted by Maisland and Schaeffer [1990, pages 259-268].

[Flann and Dietterich, 1989] N. S. Flann and T. G. Dietterich. A study of explanation-based methods for inductive learning. *Machine Learning,* 4:187-226, 1989.

[Frey, 1983] P. W. Frey, editor. *Chess Skill in Man and Machine.* Springer-Verlag, 1983.

[Goldberg, 1989] D. E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning,* Addison-Welsley, Reading, MA, 1989.

[Good, 1977] I. J. Good, Dynamic probability, computer chess, and the measurement of knowledge. In E. W. Elcock and D. Michie, editors, *Machine Intelligence 8*, pages 139-150. Ellis Horwood, Chichester, 1977.

[Hsu *et al.,* 1990] F- Hsu, T. Anantharaman, M. S. Campbell, and A. Nowatzyk. A grandmaster chess machine. *Scientific Amercican,* 263(4):44-50, 1990.

[Kozdrowicki and Cooper, 1973] E. Kozdrowicki and D. CoopeT. COKO III: The cooper-koz chess program. *Comm. ACM,* 16(7):411-427, 1973.

[Levinson and Snyder, 1991] R. Levinson and R. Snyder, Adaptive pattern oriented chess. In *Proceedings of AAAI-91.* Morgan-Kaufman, 1991.

[Levinson, 1989] R. Levinson. A self-learning, pattern-oriented chess program. *Intern. Computer Chess Assoc. Journal,* 12(4):207-215, December 1989.

[Levy and Newborn, 1991} D. Levy and M. Newborn. *How Computers Play Chess.* Computer Science Press, New York, NY, 1991,

[Marsland and Schaeffer, 1990] T. A, Marsland and J. Schaeffer, editors. *Computers, Chess and Cognition.* Springer-Verlag, 1990.

[Marsland, 1987] T.A. Marsland. Workshop report: Theory and practice in computer chese. *Intern. Computer Chess Assoc. Journal,* 10(4):205-210, 1987.

[Michalski and Negri, 1977] R. S. Michalski and P. Negri. An experiment on inductive learning in chess end games. In E. W. Elcock and D. Michie, editors, *Machine Intelligence 8,* pages 175-192. Ellis Horwood, Chichester, 1977.

[Minton, 1984] S. Minton. Constraint based generalization-learning game playing plans from single examples. In *Proceedings of AAAI-84,* pages 251-254. AAAI, 1984.

[Niblett and Shapiro, 1981] T. Niblett and A. Shapiro. Automatic induction of classification rules for chess endgames. Technical Report MIP-R-129, Machine Intelligence Research Unit, University of Edinburgh, 1981.

[Palay, 1985] A.J. Palay, *Searching with Probabilities.* Pitman, 1985.

[Pell, 1991] B. Pell. A computer game-learning tournament. (In Preparation), 1991.

[Pfleger and Treppner, 1987] H. Pfleger and G. TVeppner. *Chess: The Mechanics of the Mind.* The Crowood Press, North Pomfret, VT, 1987.

[Pitrat, 1976] J. Pitrat. A program for learning to play chess. In *Pattern Recognition and Artificial Intelligence.* Academic Press, 1976.

[Quinlan, 1983] J. R. Quinlan. Learning efficient classification procedures and their application to chess end games. In R. S. Michalski, J. G. Carbon ell, and T. M. Mitchell, editors, Machine *Learning,* pages 463-482. Morgan Kaufmann, San Mateo, CA, 1983.

[Scherzer *et at.,* 1990] T. Scherxzer, L. Scherzer, and D. Tjaden. Learning in Bebe. In T. A. Marsland and J. Schaeffer, editors, *Computers, Chess and Cognition,* chapter 12, pages 197-216. Springer-Verlag, 1990.

[Slate and Atkin, 1977] D. J. Slate and L. R. Atkin. Chess 4.5-the northwestern university chess program. In P. W. Frey, editor, *Chess Skill in Man and Machine,* pages 82-118. Springer-Verlag, 1977.

[Tadepalli, 1989] P. Tadepalli. Lazy explanation-based learning: A solution to the intractable theory problem. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence,* Detroit, MI, 1989. Morgan Kaufmann.

[Wilkins, 1982] D. Wilkins. Using knowldge to control tree searching. *Artificial Intelligence,* 18(1): 1-51, 1982.