Deriving Concepts and Strategies from Chess Tablebases

Matej Guid, Martin Možina, Aleksander Sadikov, and Ivan Bratko

Artificial Intelligence Laboratory, Faculty of Computer and Information Science, University of Ljubljana, Slovenia

Abstract. Complete tablebases, indicating best moves for every position, exist for chess endgames. There is no doubt that tablebases contain a wealth of knowledge, however, mining for this knowledge, manually or automatically, proved as extremely difficult. Recently, we developed an approach that combines specialized minimax search with argument-based machine learning (ABML) paradigm. In this paper, we put this approach to test in an attempt to elicit human-understandable knowledge from tablebases. Specifically, we semi-automatically synthesize knowledge from the KBNK tablebase for teaching the difficult king, bishop, and knight versus the lone king endgame.

1 Introduction

Chess tablebases [1] have enabled people a glimpse of how a perfect play looks like. It seems, however, that people are ill adapted to understanding this perfection. While tablebases are of enormous help to computers, people are for the most part puzzled by the style of play generated by tablebases. Yet, people would very much like to learn as much as possible, and there is no doubt that tablebases contain an enormous amount of potential knowledge — but in a form not easily accessible to a human mind.

There have been many attempts to extract knowledge from tablebases. Perhaps two best documented examples are a research project carried out by a chess study specialist John Roycroft [2] and the work of grandmaster John Nunn resulting in two books on pawnless endings [3, 4]. All the attempts, however, had at most limited success.

The goal of Roycroft's study was that he would learn himself to reliably play the KBBKN endgame (king and two bishops vs. king and knight). This endgame was for a long time considered generally to be drawn, until the KBBKN tablebase was computed. The tablebase showed that the side with two bishops can usually force a win, but the winning play is extremely difficult and takes a long sequence of moves under optimal play by both sides. Many moves in the optimal play for the stronger side are completely obscure to a human. Roycroft tried to extract a human-executable winning strategy by the help of this tablebase, trying to manually discover important concepts in this endgame which would enable a human to win reliably. After a one year's effort, the project ended with rather

limited success when Roycroft's accumulated skill for this endgame was still not quite sufficient to actually win against the tablebase in many of the won KBBKN positions.

The task of learning is not any easier for the computer. In an overview of the learning methods in games, Fürnkranz indicated that machine learning of understandable and usable concepts over the years did not yield much success [5]. There have been various attempts to bridge the gap between perfect information stored in tablebases and human-usable strategies. While some of these approaches succeeded for relatively small domains (such as KRK endgame in chess), the resulting models are hardly intelligible to human experts [6], not to mention beginners and novices. All related work did not result in breakthroughs in more complex domains. Moreover, the research questions how to learn human-understandable models and use them to generate instructions suitable for teaching humans remained open.

In a way learning from tablebases resembles closely the extraction of expert's tacit knowledge when constructing a knowledge base of an expert system — in both cases the knowledge is difficult to extract. Recently, we proposed a new paradigm, which facilitates semi-automatic elicitation of knowledge in the form of rules. We successfully applied it to creating a knowledge base of an expert system that recognizes bad bishops in chess middlegames and is able to explain its decisions [7, 8].

The goal of this paper is a practical demonstration of the usefulness of our approach for semi-automatic elicitation of knowledge. We used a recently developed method within the aforementioned paradigm for learning strategic goal-based rules. We harvested the tablebase to extract useful concepts and strategies which the domain expert in close collaboration with the machine learning tool turned into a textbook (and computer aid) for teaching a difficult-to-master KBNK (king, bishop and knight versus a lone king) endgame. It is important to note that at the beginning of the process the expert was unable to express such precise instructions on his own and was even unaware of some of the important concepts that were later used in the instructions.

The paper is organized as follows. We first present the obtained textbook instructions for teaching the KBNK endgame. Section 3 explains the guidelines for interaction between the machine and the expert in order to obtain a human-understandable rule-based model for playing a chess endgame, and how the instructions were derived semi-automatically from our rule-based model for KBNK. Note, however, that the details of the algorithm for obtaining the model are not a subject of this paper. In Section 4, we present the evaluation of the instructions by several renown chess teachers, and an evaluation of human-like style of play generated by our method by four international grandmasters. Finally, we give some conclusions and intentions for further work. The rule-based model for KBNK, the description of the algorithm and example games containing automatically generated instructions can be found in a web appendix at `http://www.ailab.si/matej/KBNK/`.

2 Semi-Automatically Derived Instructions for the Bishop and Knight Checkmate

We present here the instructions in the form of goals for delivering checkmate from any given KBNK position. These instructions were semi-automatically derived from the tablebases. In the following hierarchical set of goals, to successfully deliver a checkmate, the chess-player is instructed to always try to execute the highest *achievable* goal listed below. The goals are listed in order of preference, goal 1 being the most preferred. The chess-player is expected to know how to avoid stalemate, piece blunders, and threefold repetitions. Apart from descriptions of the goals we also illustrate most of the concepts behind them.

Goal 1: Deliver Checkmate A checkmating procedure is the following: two consecutive checks with the minor pieces are delivered, the later one resulting in one of the two types of checkmate positions shown in Fig. 1.

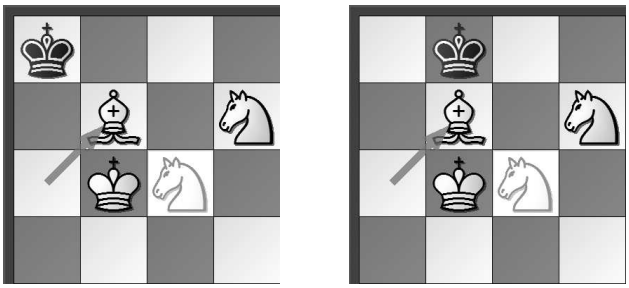

Fig. 1. Checkmate can be delivered by the bishop or the knight, always in the corner of the bishop's color ("right" corner). Each arrow indicates last bishop's move.

Goal 2: Prepare the Knight for Checkmate This goal applies when the king and the bishop restrain the defender's king to only two squares: the corner square and a square on the edge of the board right beside the corner square (see Fig. 2). The task of the attacker is to prepare the knight so that it is ready for the checkmating procedure.

Goal 3: Restrain Defending King to A Minimal Area Beside The Right Corner The task of the attacker is to take squares away from the defending king until it is driven to the edge of the board, and consequently to the corner square. The chess-player is advised to aim for the type of position shown in Fig. 2 where the king and the bishop restrain the defending king to a minimal area beside the right corner.

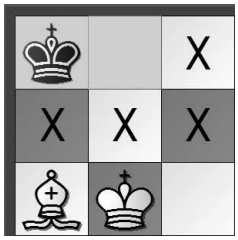

Fig. 2. A minimal area.

Goal 4: Build A Barrier and Squeeze Defending King The attacker is advised to build *a barrier* that holds the defending king in an area beside the right-colored corner. When such barrier is built, the attacker should aim to squeeze the constrained area in order to further restrain the defending king (see Fig. 3).

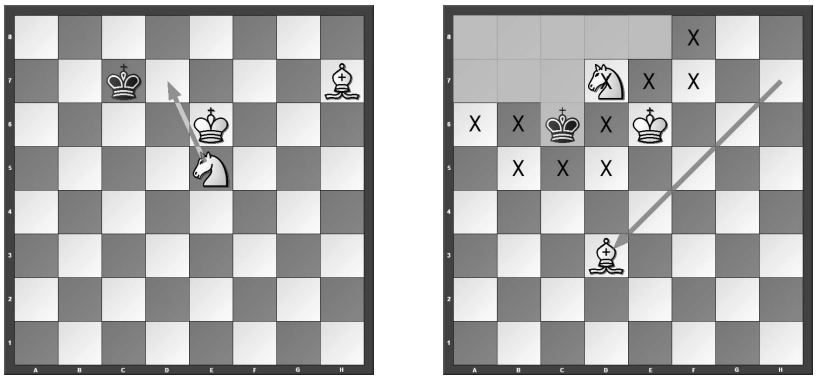

Fig. 3. In the position shown in the left diagram, the attacking side could build the barrier in the following manner: 1.Ne5-d7 Kc7-c6 2.Bh7-d3!, leading to the position on the right. The area around the right-colored corner to which the defending king is confined, could be squeezed further, *e.g.*, after 2...Kc6-c7 3.Bd3-b5.

Goal 5: Approach Defending King from Central Side A part of the basic strategy is to drive the opposing king to the edge of the board. In order to achieve this, it is beneficial for the attacking side to occupy squares closer to the center of the chessboard then the defending king does. The attacker should aim to approach the opposing king from the central side of the board.

Goal 6: Block the Way to the Wrong Corner When the defender's king is already pushed to the edge of the board, the attacker's task is to constrain as much as possible the defending king's way to the wrong-colored corner. At the same time, the attacker should try to keep restraining the king to the edge of the board. Fig. 4 shows an example of a typical position that often occurred in simulated games.

Goal 7: Push Defending King Towards The Right Corner The attacker is advised to push the defending king towards the right-colored corner, at the same time not allowing it to move further away from the edge of the board (see Fig. 5).

Goal 8: Push Defending King Towards the Edge The attacker is advised to arrange the pieces in such way so that the defending king is pushed towards the edge of the board, and cannot immediately increase the distance from the edge.

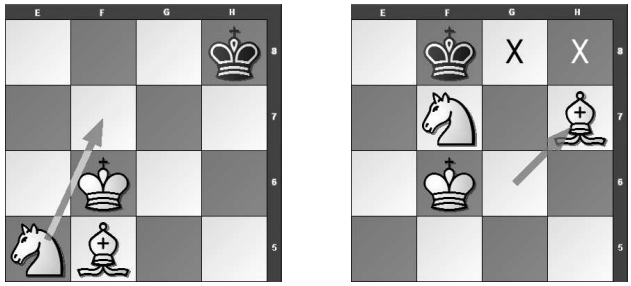

Fig. 4. In the position on the left, white pieces lure the defending king out of the wrong corner: 1.Ne5-f7+ Kh8-g8 2.Bf5-g6 Kf8 (note that this is the only available square, since h8 is attacked by the knight) 3.Bh7! The last move in this sequence takes under control square g8, and sets up the blockade one square farther from the wrong corner.

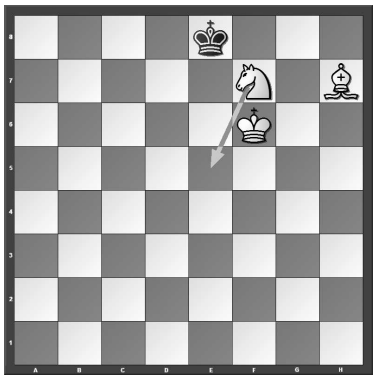

Fig. 5. Black king's distance from the right corner (a8) should decrease, and black king should not be allowed to move away from the edge of the chessboard. This is achieved by the move 1.Nf7-e5, and black cannot resist white's goals: even after suboptimal 1...Ke8-f8 (optimal move according to tablebases is 1...Ke8-d8) white could play 2.Ne5-d7+ Kf8-e8 3.Kf6-e6 and black should move closer to the right corner with the only available move 3...Ke8-d8.

Goal 9: Approach With the King The attacker is advised to move the king closer to the opposing king.

Goal 10: Bring the Knight Closer to the Defending King The attacker is advised to bring the knight closer to the defending king.

Default Goal: Keep the Kings Close If none of the above goals is achievable, at least keep the king as close as possible to the defending king, and - if possible - strive for the opposition of the kings.

2.1 Example Games with Automatically-Generated Suggestions

The instructions given in the previous subsection are accompanied by example games containing automatically generated instructions. An instruction is given each time the previous suggested goal was accomplished. These games serve to illustrate how the teaching process would run with the help of a computer.

The student would first read the instructions and then be presented a random position to play against the computer. At any point in the game, the computer is able to give an appropriate suggestion to the student in the form of a goal to accomplish. These suggestions/goals could be further augmented by occasionally displaying a side diagram containing the position associated with the given goal. We give the games in a web appendix at `http://www.ailab.si/matej/KBNK/`.

3 The Process of Synthesizing Instructions

3.1 Basic Description of Our Approach

As already mentioned, the rules were induced by a recently developed method for goal-based rule induction [9]. This method extracts a strategy for solving problems that require search (like chess, checkers etc.). A strategy is an ordered list of goals that lead to the solution of the problem, similar to advice list in Advice Languages [10]. These goals can then be used to teach a human, who is incapable of extensive search, how to act in these domains and be able to solve these problems simply by following suggested goals. The method combines ideas from the Argument Based Machine Learning (ABML) [11] with specialized minimax search to extract a strategy for solving problems that require search.

3.2 Obtaining Knowledge from Domain Expert

In order to obtain meaningful and human-understandable instructions, the knowledge has to be elicited from a chess expert (in our case this was a FIDE master). Each chess position is described with a set of features that correspond to some well-known chess concepts. The features are obtained by domain expert as a result of the knowledge elicitation process.

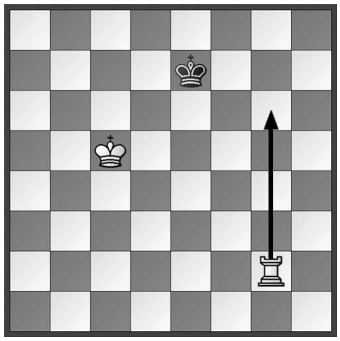

Fig. 6. Computer: *"What goal would you suggest for white in this position? What are the reasons for this goal to apply in this position?"* The expert used his domain knowledge to produce the following answer: "Black king is close to the edge of the board, but the king is not constrained by white pieces. Therefore I would suggest White to constrain black king."

The knowledge elicitation process is similar as in [7, 8]: domain expert and machine learning algorithm improve the model iteratively. A typical interaction between the method and the expert is shown in Fig. 6. As a result of this particular interaction, a new attribute *king_constrained* was introduced.

3.3 Strategic Goal-Based Rules

Our hierarchical model consists of an ordered set of rules of the following form:

$$\texttt{IF } \textit{preconditions } \texttt{THEN } \textit{goal}$$

The rule's *preconditions* and *goals* are both expressed by using the aforementioned features. The method used the expert's argument given in Fig. 6 to induce the following rule:

$$\begin{gathered}\texttt{IF } \textit{edist} < 3 \texttt{ AND } \textit{king\_constrained} = \textit{false} \\ \texttt{THEN } \textit{king\_constrained} = \textit{true} \texttt{ AND } \textit{edist should not increase}\end{gathered}$$

where *edist* is the distance between black king and the edge of the board. The subgoal *edist should not increase* was added by the computer. The method recognized that allowing Black to move away from the edge of the board would increase the distance to mate. The expert can accept or reject such suggestions before the rule's acceptance, but doing so it is important to rely on his or her *common knowledge* about the domain.

Preconditions can be a conjunction of various conditions (if none are given, the goal is tried each time when no higher goal in the hierarchy is achievable). Similarly, a goal is a conjunction of subgoals, where a subgoal can specify the desired value of an attribute (*true*/*false*, $<$, $>$, etc.), its optimization (*minimize*, *maximize*), and any of four possible qualitative changes: *decrease*, *increase*, *not decrease*, *not increase*. Each rule should contain exactly one progressive subgoal (as is the change of the value of *king_constrained* in the above rule). Note that a subgoal *edist should not increase* is not progressive, since it allows a chess-player to merely maintain the status, not progressing towards delivering checkmate.

3.4 Allowing Non-Optimal Play

Often, *counter examples* are detected by the method, and presented to the expert. Counter examples are positions where the goal can be achieved, but the resulting play nevertheless leads to *increased* distance to mate. Among such positions, the one with the highest distance to mate is chosen as the key counter example. Figure 7 illustrates.

Since human players typically choose a longer path to win by systematically achieving intermediate goals, the expert is instructed to accept counter examples where such subgoals are executed, although they often do not lead to optimal play in the sense of shortest win against best defence. However, the resulting play in counter examples should lead to overall progress towards achieving the final goal of delivering checkmate. Constraining the black king in the above counter example was judged to lead to such progress.

The expert may also find the execution of the goal in a counter example to be unacceptable. In this case, he or she may add, modify, and/or remove any of the preconditions and subgoals. Again, doing any of these, it is important that the expert relies on his or her common knowledge about the domain.

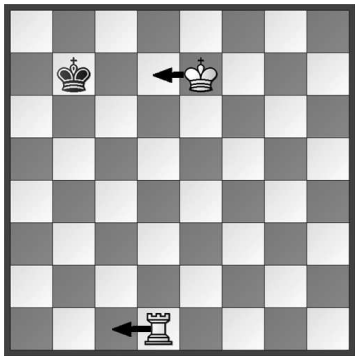

Fig. 7. Computer: *"Would you admonish a student if he or she played 1.Rd1-c1 in this position?"* The expert found this move to be perfectly acceptable. Despite of its non-optimality: from the tablebase point of view 1.Ke7-d7 is a much better move - 1.Rd1-c1 (the worst possible execution of the suggested goal) achieves mate in 11 moves whereas after 1.Ke7-d7 only 6 moves are necessary (after 1...Kb7-b6 2.Rd1-d5!).

3.5 Hierarchy of Goals

When a rule triggers, all the goals higher in the hierarchy are also taken into account. The goal is achievable when at least one of these goals can be executed regardless of the defender's play (optimal or non-optimal). Such hierarchy of goals is typical of a human way of thinking. For example, when the goal is to push the defender's king towards the right-colored corner in the KBNK endgame and the defender resists the goal by allowing the opponent to deliver checkmate (that would not be achievable without the opponent's help), one is expected to see such a possibility. It would be redundant to express goals in the following way: "Push the defending king towards the right corner *or deliver a checkmate, if the opponent plays badly and allows it.*"

3.6 Constructing Human-Friendly Instructions from Semi-Automatically Generated Rules

The role of preconditions and non-progressive subgoals is merely to allow a computer program to detect positions where specific rule triggers and to execute goals appropriately. All the goals in the instructions are obtained by stating only the progressive subgoal. The exception is the last, default goal, since it is desirable to always be able to give advice to the student. Let us demonstrate this on the following rule (the descriptions of the attributes are available in a web appendix at `http://www.ailab.si/matej/KBNK/`):

`IF` *edist* < 1 `THEN` *edist should not increase* `AND` *knight_on_edge* $=$ *false*
`AND` *wrong_corner_way should decrease* `AND` *wrong_corner_way minimise*
`AND` *white_king_more_central* $=$ *true*

The precondition *edist* < 1 enables the program to try to achieve the goal only when the defending king is confined to the edge of the board. The progressive subgoal is *wrong_corner_way should decrease*, so when this rule triggers (*i.e.*, this goal is achievable and no higher rule triggers) the student is given the following advice: "Block the way to the wrong corner." It is expected from a human to recognize (at least eventually) that moving the knight to the edge, allowing

the opponent to move away from the edge, and putting the king closer to the edge than the opponent's king does not lead to progress. For computer, such constraints are necessary to enable a sensible execution of the goals.

It is also desirable to obtain sensible diagrams and variations that are supposed to provide most useful representation of the goals and concepts in a given domain. We obtained these by executing simulations of delivering checkmate from randomly chosen initial positions using the hierarchy of goals. The execution of goals in these simulations was optimal in sense of minimizing the distance to mate (quickest play). For each goal, the position that occurred most frequently in the simulations, was chosen to be presented by a diagram. When several positions occurred equally frequently, more diagrams were used. Due to space limitations, we only presented the diagrams that occurred most frequently.

4 Discussion and Evaluation

The bishop and knight checkmate (KBNK) is regarded as the most difficult of the elementary mates. Several chess books give the general strategy for playing this endgame as follows. Since checkmate can only be forced in the corner of the same color as the squares on which the bishop moves, an opponent will try to stay first in the center of the board, and then retreat in the wrong-colored corner. The checkmating process can be divided into three phases: (1) driving the opposing king to the edge of the board, (2) forcing the king to the appropriate corner, and (3) delivering a checkmate. However, only knowing this basic strategy hardly suffices for anyone to effectively checkmate the opponent.[1] Another strategy is known as *Delétang's triangles*, involving confining the lone king in a series of three shrinking isosceles right-angled triangles (pioneered by Delétang in 1923 [12]). This strategy usually takes five to ten moves longer to deliver checkmate. Since state-of-the-art endgame manuals (*e.g.*, [13]) prefer teaching the aforementioned three-phase checkmating process, we decided to aim for obtaining the rules for executing that (quicker) strategy.

To the best of our knowledge, no formalized models for KBNK endgame suitable for teaching purposes were derived by any machine-learning programs. As H.J. van den Herik et al. in 2002 (and still valid today) nicely put it: "The current state of the art of machine-learning programs is that many ad hoc recipes are produced. Moreover, they are hardly intelligible to human experts. In fact, the database itself is a long list of ad hoc recipes. Hence, the research question is how to combine them into tractable clusters of analogue positions and then to formulate a human-understandable rule."[6]

Based on the aforementioned Delétang's triangles method, van den Herik constructed a formalized model for playing KBNK endgame and successfully implemented it in a chess-playing program [14]. The knowledge in the model was partitioned into 28 patterned equivalence classes (introduced by Bramer [15]) aimed to correspond to some significant recognizable *features* of the endgame as

[1] For example, grandmaster Epishin (Kempinski-Epishin, Bundesliga 2001) failed to force the defending king to the appropriate corner and the game ended in a draw.

perceived by chess-players such as "confinement in the wrong corner" and "intermediate class between the large and the middle bishop triangle" (the obtained equivalence classes and patterns are fully described in [16]). The model was derived from chess theory books, discussions with (grand)masters, and the author's experience, but without any machine-learning programs or chess-tablebases support. Similarly as with our approach, the resulting knowledge is intended to be used jointly with tree search with a maximum search depth decided in advance and does not necessarily produce optimal play. There are several important differences between [14] and our approach, the most important two of them being:

- Obtaining the formalized model for KBNK in [14] required *existence* of some method for playing the endgame in question (in this case, the method discovered by Delétang), while our ABML-based knowledge elicitation process provides a potential for obtaining goal-based instructions for *any* chess endgame where tablebases are available. Note that no known method for delivering checkmate exist for more complex endgames (such as KBBKN, for example).
- Using pattern-classes based model leads to instructions in form of descriptions of *states* the chess-player should aim to achieve from a given position, while our strategic goal-based rules suggest the relative change (improvement) in a position given in terms of *one progressive goal* per instruction. For a student, it seems easier to memorize instructions containing a single progressive goal then a sequence of several states.

In another attempt to obtain a formalized model for KBNK endgame, van den Herik and Herschberg [17] strived for optimal play, using tablebases. After the very limited success, the task of translating perfect information into a set of rules to be followed by human or computer was reported to be extremely difficult.

The extracted strategy as described in Section 2 was presented to three chess teachers (among them a selector of Slovenian women's squad and a selector of Slovenian youth squad) to evaluate its appropriateness for teaching chess-players. They all agreed on the usefulness of the presented concepts and found the derived strategy suitable for educational purposes. Among the reasons to support this assessment was that the instructions "clearly demonstrate the intermediate subgoals of delivering checkmate."

We also evaluated the rules by using them as a heuristic function for 6-ply minimax search to play 100 randomly chosen KBNK positions (each requiring at least 28 moves to mate providing optimal play[2]) against perfect defender. We tested two different strategies for cases when the heuristic suggests several moves achieving the goal: either (a) a move that minimizes distance to mate (quickest play), or (b) a move that maximizes distance to mate (slowest play) was chosen. Our rules were able to achieve mate in 100% of the cases using both quickest play (average game length was 32 moves) and slowest play (average game length was 38 moves). Therefore, even with slowest possible realization of strategic goals, the strategic rules are expected to guide a student reliably towards the final goal

[2] KBNK is a 33-move game in the maximin sense, as it was established after the complete tablebases were computed by Dekker and van den Herik in 1982 [16].

of achieving checkmate within allowed 50 moves. It is worth noting that state-of-the-art chess engines such as Rybka 2.1, Zappa 1.1, and Toga II 1.3.1, when limited to a 6-ply search only, were not able to deliver checkmate within 50 moves against optimal defender from any of the 100 starting KBNK positions.

Our ABML-based approach leads to obtaining domain's strategic goal-based rules using the same arguments and the same domain language attributes as the expert does. We therefore expect the resulting models to produce "human-like" style of play, in the sense that it would be clearly understandable by human players. As typical for humans, such play would not aspire to minimize distance to win. To verify this hypothesis, we applied our approach to constructing strategic rules for the KRK chess endgame, where it is commonly accepted that a traditional way of delivering mate differentiates from optimal (tablebase) play.

We verified our hypothesis with a kind of a Turing test. Four strong grandmasters were asked to observe 30 games: 10 games played by our KRK chess program guided only by the rules obtained with our ABML-based method, 10 games by a perfect (tablebase) player, and 10 additional games (to further complicate the evaluators' job), all facing a perfect (tablebase) opponent. They were only told that at least in some of the games white player was a computer program, while black always defended optimally. The grandmasters were asked to express their assessment for each game to what degree (marks 1 to 10) they find the play to be human-like. The mark of 1 means that the attacker's play seems totally computer-like, and mark 10 means that it seems totally human-like. The average scores given to our KRK rules by the four grandmasters were 4.1, 7.1, 8.2, and 7.3, while the average scores given to tablebase player were 2.2, 3.1, 1.8, and 2.0, an obvious difference!

5 Conclusions

We developed a procedure for semi-automatic synthesis of textbook instructions for teaching the KBNK endgame, accompanied by example games containing generated instructions. The presentation of the derived strategy includes, importantly, a number of concepts and key positions from this endgame that help the human learner to easily understand the main principles of this strategy. The key positions were detected automatically from simulated games played by the derived strategy. The key positions in Figures 4, 5 and 3 belong to a frequent sequence of seven moves in tablebase play. In our case, this sequence was reduced to the three key positions and conceptualized in terms of goals along the sequence. In contrast to memorizing the optimal sequence itself, the extracted generalization also enables correct play against sub-optimal defence. The derived strategy is human-friendly in the sense of being easy to memorize, but produces suboptimal play. In the opinion of chess coaches who commented on the derived strategy, the tutorial presentation of this strategy is appropriate for teaching chess students to play this ending.

We view the positive assessment of derived textbook instructions by chess coaches as a confirmation that our approach is able to facilitate knowledge ex-

traction from the tablebases. We explained the guidelines for interaction between the machine and the expert in order to obtain a human-understandable rule-based model for playing a chess endgame, and how the instructions, including illustrative diagrams, could be derived semi-automatically from such a model.

Our next goal is to create a computer tool for teaching the KBNK endgame. All the main ingredients are already available as described in this paper, and all that remains is to package them into an actual application as described in Sect. 2.1. Another goal is to tackle a much harder endgame, namely KBBKN.

References

1. Thompson, K.: Retrograde analysis of certain endgames. International Computer Chess Association Journal **9**(3) (1986) 131–139
2. Roycroft, A.J.: Expert against oracle. In Hayes, J.E., Michie, D., Richards, J., eds.: Machine Intelligence 11, Oxford University Press, Oxford, UK (1988) 347–373
3. Nunn, J.: Secrets of Minor-Piece Endings. Batsford (1995)
4. Nunn, J.: Secrets of Pawnless Endings. Gambit Publications Limited (2002)
5. Fürnkranz, J.: Machine learning in games: A survey. In Fürnkranz, J., Kubat, M., eds.: Machines that Learn to Play Games, New York, NJ: Nova Scientific Publishers (2001)
6. van den Herik, H., Uiterwijk, J., van Rijswijck, J.: Games solved: Now and in the future. Artificial Intelligence **134** (2002) 277–311
7. Možina, M., Guid, M., Krivec, J., Sadikov, A., Bratko, I.: Fighting knowledge acquisition bottleneck with argument based machine learning. In: The 18th European Conference on Artificial Intelligence (ECAI). (2008) 234–238
8. Guid, M., Možina, M., Krivec, J., Sadikov, A., Bratko, I.: Learning positional features for annotating chess games: A case study. In: Lecture Notes in Computer Science. Volume 5131. (2008) 192–204
9. Možina, M., Guid, M., Sadikov, A., Bratko, I.: Goal-Based Rule Learning. Technical report, Faculty of Computer and Information Science, University of Ljubljana, Slovenia (2009) Also available as http://www.ailab.si/matej/KBNK/GBRL.pdf.
10. Bratko, I.: Knowledge-based problem-solving in AL3. Machine Intelligence **10** (1982) 73–100
11. Možina, M., Žabkar, J., Bratko, I.: Argument based machine learning. Artificial Intelligence **171**(10/15) (2007) 922–937
12. Delétang, D.: Mat avec le fou et le cavalier. Las Stratgie **56**(2) (1923) 25–32
13. Dvoretsky, M.: Dvoretsky's Endgame Manual, 2nd edition. Russell Enterprises, Inc. (2008)
14. van den Herik, H.J.: Representation of experts' knowledge in a subdomain of chess intelligence. In: IJCAI. (1983) 252–255
15. Bramer, M.: Representation of knowledge for chess endgames: towards a self-improving system. PhD thesis, The Open University: Faculty of Mathematics, Milton Keynes, England (1977)
16. van den Herik, H.J.: Computerschaak, Schaakwereld en Kunstmatige Intelligentie. PhD thesis, Delft University of Technology – Universidade Nova de Lisboa (ITQB/UNL), Academic Service, The Hague (1983)
17. van den Herik, H.J., Herschberg, I.: Omniscience, the rulegiver? In B. Pernici, M.S., ed.: Proceedings of L'Intelligenza Artificiale Ed Il Gioco Degli Scacchi, III Convegno Internazionale. (1986) 1–17