Automated Chess Tutor

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

Artificial Intelligence Laboratory, Faculty of Computer and Information Science,
University of Ljubljana, Slovenia
{aleksander.sadikov,martin.mozina,matej.guid,bratko}@fri.uni-lj.si

Abstract. While recently the strength of chess-playing programs has grown immensely, their capability of explaining in human understandable terms why some moves are good or bad has enjoyed little attention. Progress towards programs' with an ability to provide intelligent comment on chess games, either played by a program or by a human, has been negligible in comparison with the progress concerning playing strength. The typical style of program's "comments" (in terms of the best variations and their numerical scores) is of little use to a human who wants to learn important concepts behind the variations.
In this paper, we present some core mechanisms for automated commenting in terms of relevant goals to be achieved or preserved in a given position. By combining these mechanisms with an actual chess engine we were able to transform this engine into a chess tutor/annotator that is capable of generating rather intelligent commentary. The main advantage of our work over related approaches is: (a) it has the ability to act as a tutor for the whole game of chess, and (b) it has a relatively solid understanding and therefore an adequate commenting of positional aspects.

1 Introduction

In the last five years the strength of computer chess programs has grown immensely. They have now reached and most likely surpassed the level of the human World Champion. Some of the programs are free to use. They run on ordinary personal computers, thus enabling practically everyone to have a world-champion caliber player at home to play with whenever they wish to. This is a commodity that no one imagined possible a decade or two ago.

However, there is one big difference between having a strong computer program and having a human grandmaster at home. The latter can be asked questions, you can learn from him or her [1], you can analyze games together and he or she can explain complex concepts to you. But why cannot a computer program do that? A program is tactically impeccable and in play it exhibits an amount of strategic knowledge comparable to a solid grandmaster. In 2003, Garry Kasparov, the highest rated chess player in the world, said that computers are over

[1] In the remainder of this paper we use 'he' and 'his' whenever 'he or she' and 'his of her' are meant

2800 Eloin tactics and about 2500 Elo in strategy. Since then, they improved in both categories. So why cannot they give us lessons?

The answer lies in the output the programs give. They usually output nothing more than a principal variation and the numerical evaluation of several best moves — in other words, some moves and numbers are given. Typically, people are not good in comprehending numbers, but in this case we grew accustomed to interpreting that a value of −3 would mean losing a Knight or some other equivalent of three Pawns. Yet, we are puzzled by that cheeky evaluation of −0.39; what does that mean? That we are 0.39 Pawns behind in material? What is 0.39 Pawns? The point we are trying to make is that there exists a communication barrier between humans and chess programs. We clearly do not speak the same language. While we can understand the tactical lines with a forced win or a loss of material (even though it would not hurt hearing it in words like "you lose a Knight"), we may have difficulties understanding the quiet, strategic lines.

This is a pitty. On the one hand we have at our disposal an oracle of knowledge, yet on the other hand we cannot understand what it is saying for the greater part. Imagine that you were given to understand the meaning of that tricky −0.39, i.e., what it really stands for. For example, "You own a bishop pair, and have a safer King, but it does not compensate for the two isolated Pawns on the queenside and the strong Knight's outpost on d3. Your position is somewhat (−0.39) worse." This would be something that one could learn from. There is a myriad of games between strong players played or relayed on the internet. With a chess program that could provide comments that we understand, we would be able to learn actually while watching those games.

This is the long-term goal of our research. The current paper presents an initial, though perhaps most important, step towards this goal. We took Robert Hyatt's Crafty [3], a strong freeware chess program, as our underlying chess engine and on top of it built an annotating/tutoring module. The chess program is thus turned into an intelligent chess tutor.

2 Related Work

The idea of the automatic construction of commentary is not new. Its importance was realized long ago. Around 1980, Donald Michie was probably the first to propose research in this direction. The ICGA (then still ICCA) Journal started an annual competition for the best chess annotation program — named after the late Professor Herschberg. The first winner of the competition was Chessmaster 4000 for the year 1993 [1]. The program commented practically every move with some prose, but it was mostly very simplistic. The next year, Jeff Mallet's Innovation, an amateur program, won as the only entrance. It gave interesting comments, but was hampered by a relatively poor chess engine. Then, three years in a row (with no award given for 1997) different versions of the Dutch program Fritz won the prize. It had a very strong engine and gave good analysis — but its commentary lacked prose. This handicap slightly improved from version to version, yet the winner of 1998, – it was the last winner – still lacks a great deal of experience in this respect. It has a good selection of (1) when to comment, (2) which subvariations to give (not easy problems at all) and (3) how long the variations should be. A very fine point is that it also points out the tactical pitfalls the players avoided. Yet, it still explains only tactical nuances and ignores what we discussed in the introduction — why is a position -0.39 in strategic terms.

Still, in the mid 1990s, the spirits were high and the progress made even prompted David Levy and Tony Marsland [4], members of the competition jury, to write: "If progress in this field continues to be made at the same rate we would expect that, by the end of the century, there will be annotation programs available which can offer a very useful service to chess-players of every strength." Surprisingly, though, the 1998 FRITZ was the last entrance in the competition to this day.

On the other side of the field, Seidel [6] made an attempt to create endgame programs that could annotate their play. He tackled elementary endgames of the lone-king type such as KRK and KBBK. At the core of the system is a general rule for generating moves for the stronger side that (1) determines the threats by the weaker side, (2) generates possible actions, and (3) eliminates those moves that do not parry the threats. Threats and actions are very specific for various endgames and rely heavily on chess theory. They represent the chess knowledge of the system and are defined manually. The annotation mechanism follows simply from the move-generating scheme by commenting which actions and threats are encountered. The chess knowledge in this form is both a strong and a weak point — strong, because it enables specific annotations, and weak because a vast amount of manually entered knowledge is required even for very simple endgames. For example, there are six different types of moves (actions) for just a part of the KBBK endgame. This severely limits the extendability of the scheme to more complex endgames, not to mention the whole game. A problem with the whole game, apart from just the raw amount of knowledge needed, is the lack of strict rules — something Seidel's system requires.

Herbeck and Barth [3] took a different and in some ways perhaps a more conventional approach. They combined alpha-beta search with knowledge in the form of rules. Their domains were also chess endgames (up to four pieces), though the knowledge was somewhat more involved than in Seidel's [6] case. Herbeck and Barth's algorithm performed a standard search and then followed the principal variation until a position was encountered in which a rule from the knowledge base stated the outcome (or a bound on the outcome) of the game. This rule (or a set of them) is offered to the user as an explanation. The algorithm is also capable of giving comments (rules) for alternative variations. An example of the rules in KPKP is "White wins, because the white Pawn promotes first with check and Black is in a bad position". As in Seidel's case the rule base can be viewed both as a strong and a weak point of the scheme — strong because the annotations are quite good, and weak because (a) the rules are manually crafted, and (b) there are no such exact rules for more complex endgames, let alone for the middlegame. The latter prevents the scheme to be extended beyond relatively simple endgames.

We should also briefly mention here Gadwal et al.'s work on tutoring King and Bishop and two Pawns on the same file versus lone King endgame [2]. As with the other studies mentioned the main problem with it is the lack of extendability beyond the simple endgames.

3 The Tutoring System

Our main underlying idea is to use the chess engine's evaluation function's features to describe the changes in the position when a move is made. These elementary features can later be combined to form higher-level concepts understandable to humans. In this manner we bridge the communication barrier between machines and humans.

Our tutoring system consists of three components: (1) the chess engine, bringing chess knowledge and search to the table, (2) the commenting module for generating raw comments from the engine's evaluations, and (3) a simple expert system for refining the raw comments into (an even more) human understandable language.

The first component, the chess engine, serves (1a) to calculate variations, (1b) analyze positions, and (1c) evaluate possible moves. It is essentially an unmodified version of Crafty (apart for some routines that transfer calculated variations, moves, and detailed evaluations to our commenting module). The depth of search is either fixed or time-controlled; higher depths provide better variations and thus better (in terms of playing strength) commentary. As for the greater part the other two components are relatively independent of the engine, any other chess engine could be used in Crafty's place. In fact, we believe it would be quite advantageous to use an engine that champions knowledge in favor of search.

The other two components, which form the backbone of the tutoring system, will be described in the sections 3.1 and 3.2.

3.1 The Commenting Module

At first glance, moves can be divided into two categories: good moves and bad moves. Yet, when delving into the matter more deeply, one can see that most moves in games, pitting two players with opposing interests against each other, are a sort of tradeoff of positive and negative characteristics of the position. With most moves you gain something and you lose something.

These characteristics and their tradeoffs is what our commenting module is calculating and analyzing. For any given move, it shows what characteristics of the position have changed and on the basis of this and the change in score, as given by the engine, the tutor can elaborate what is the essence of the given move or variation. The general merit of the move, however, is obtained by simply comparing its score with the scores of other possible moves in the given position.

Most chess engines, Crafty being no different, employ an evaluation function in the form of a weighted sum of the position's features. These features, along with their associated weights, are actually the position's characteristics on which our commenting module is operating. The weights are important too, because they define the relative importance of the features (characteristics).

Commenting Good Characteristics. In general, the tutoring system has two possibilities to generate a comment why a certain move is good: (a) the move achieves progress towards some goal, or (b) the move overcomes some weakness or deficiency of the current position. Let us take a look at both options in more detail.

The basic idea behind the first option is that making headway involves achieving progress towards goals, eventually accomplishing them. The goals in our schema are simply the evaluation function's features. We believe this is a natural way to state straightforward, comprehensible goals to be achieved. Later we show how the expert system can combine several straightforward goals into a more structured one thus increasing the expressive power of the tutoring system.

Figure 1a illustrates the setting for this idea. First, a search procedure is employed to obtain a principal variation starting with the move to be commented upon. The final position in the principal variation represents the goal position — this is the position that one can reach from the initial position with the move under investigation. This position might be viewed as envisioned by the player when he made the move. After we have obtained this envisioned position, we calculate which features of the evaluation function have changed and by how much they changed when comparing this position with the starting position. If one looks at the evaluation function's estimation of a position as a vector of values this operation is a simple difference between the corresponding position vectors.

The positive characteristics (or rather positively changing characteristics) achieved are those that have positive values in the resulting vector of differences (here we assume that we comment from White's perspective; otherwise it is just the opposite). In the raw output, each such characteristic represents a separate comment of what the move under investigation aims to achieve. Basically, at this stage, the commentary is a list of positive characteristics the move (or rather the principal variation starting with the move under investigation) aims to achieve.

For example, if the following two characteristics were singled out as the ones that changed positively:

```
WHITE_KNIGHTS_CENTRALIZATION
WHITE_KING_SAFETY
```

then the raw commentary would be "The move aims to centralize the Knight and to improve King's safety".

It should be noted that both, the starting position and the envisioned position must be quiescent. The starting position should be quiescent, because there is

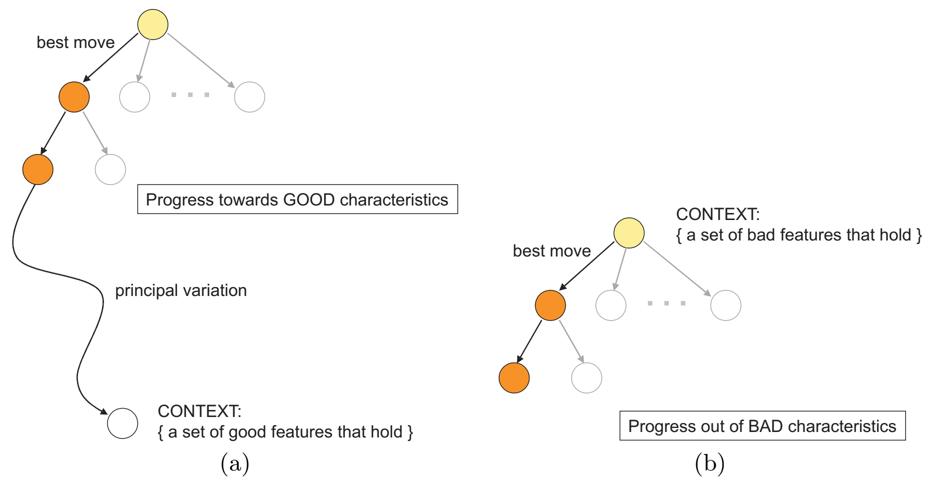

best move
Progress towards GOOD characteristics
principal variation
CONTEXT:
{ a set of good features that hold }
(a)
CONTEXT:
{ a set of bad features that hold }
best move
Progress out of BAD characteristics
(b)

Fig. 1. Commenting good characteristics

no point in commenting in the middle of a tactical sequence; such a sequence should be viewed as a whole. The envisioned position should be quiescent for obvious reasons — the evaluation based on a non-quiescent position is completely unreliable and thus of little use.

Let us now take a look at the other possibility why a move can be good — namely, because it overcomes some weakness or deficiency of the current position. This situation is illustrated in Fig. 1b. However, computationally this possibility is implemented exactly as the first one. We note that good characteristics for the opponent are bad for us, and are represented with negative numbers in the position's vector. For example, one such negative characteristic (position's deficiency) could be:

```
BLACK_BISHOP_PAIR
```

and in this case the generated raw commentary would be "The move aims to eliminate the Black's bishop pair".

Commenting Bad Characteristics. The tutoring system has three possibilities to generate a comment why a certain move is bad: (a) the move creates a weakness or deficiency in the position, (b) the move spoils some good characteristic of the current position, or (c) the move is compared to a better (the best) possible move and the differences are pointed out. Possibilities (a) and (b) are quite similar to the two possibilities we encountered earlier when discussing how commenting of good aspects of a move is accomplished. Therefore we first briefly debate these two options and then take a look at option (c).

Possibility (a) mirrors possibility (a) for generating comments for good moves. The difference is that there we attempted to achieve some positive goals, while here the move achieves some negative goal(s). Similarly, possibility (b) mirrors possibility (b) for generating comments for good moves. Here, the difference is that instead of removing some weakness or deficiency of the current position, the move removes some positive characteristic of the current position.

From the computational point of view, the only difference is that we are now looking at the evaluation features that are negative in the vector of differences between the starting position and the envisioned position. The rest is the same. For example, if the following features were flagged as changed for the worse (negatively changing characteristics):

```
EVALUATE_PAWNS
BLACK_ROOK_BEHIND_PASSED_PAWN
```

the raw commentary generated would be "The move allows the opponent to improve the pawn structure and to get a rook behind a passed Pawn".

However, there is a further difficulty when commenting really bad moves — be it bad in positional or tactical sense. The nature of minimax search is such that it does not search for a sort of "bad envisioned position", but rather allows the mistake (as it is forced upon it by the user) and then searches for best play for both sides from that point on. So, in essence, the envisioned position at the end of the principal variation returned by the search may not necessarily reflect the real weakness of the bad move (which is what we would like to comment upon), because this weakness was perhaps traded for some other weakness later in the variation.

Let us illustrate this by an example. Assume that White made a mistake by moving a Pawn and that as a consequence Black gained a strong outpost for his Knight. Later in the principal variation, stemming from this initial pawn move, however, Black exchanged this strong Knight for White's Bishop and so eliminated White's strong bishop pair and doubled White's Pawns. The tutoring system, comparing the starting position with the position at the end of the principal variation, would comment that "The move allows the opponent to eliminate your bishop pair and to weaken your pawn structure". While in principle this is true, it may be more in the spirit of tutoring to say instead "The move allows the opponent to gain a strong knight outpost". The initial comment can prove too abstract to the user. Or, after all, the user can choose not to follow the principal variation at all.

The difficulty we described is actually a special case of a more general problem — namely, how long should the principal variation be and where in it we should decide to comment. This is a cognitive problem and amongst other things it depends on the chess strength of the user whom the system attempts to tutor. In some cases only a single move should be commented, in other cases the whole variation, and in a third group of cases a part of the variation. This problem indicates our future work.

The idea behind possibility (c) is different from the concepts we discussed so far. Instead of observing what the move achieved, positive or negative, we observe what the move did *not* achieve although it could have. Therefore, we compare the move with the best move that is available. In essence, we generate comments for the move played and for the best move that could have been played, and attempt to compare their respective merits. This possibility is still highly experimental and is mentioned here just to give the reader the idea how we plan to extend our tutoring system's capabilities.

3.2 The Expert System for Commentary Refinement

In principle, the raw comments generated by the commenting module are in itself sufficient to produce relatively intelligent tutoring and annotations. However, it is both easy and rather beneficial to refine these comments to obtain quite human-like commentary. This is the purpose of the expert system for commentary refinement.

It is useful to refine the raw comments for several reasons: (a) to remove the unnecessary complementary comments, (b) to explain further or better the meaning of some features in the given context, and (c) to combine some basic features into higher-level, structured features that are more understandable to humans, thus increasing the expressive power of the system. Below we take a more detailed look at these options.

The structure of CRAFTY's evaluation function (other engines are probably quite similar in this respect) is such that it separately looks for the same features for White and for Black [3]. For example, it looks for White's Rook on an open file and for Black's Rook on an open file. These are two different features. Yet, they can be looked upon as a pair of complementary features. If, in a given variation, the commenting module realized that both these features changed — meaning that both players got a Rook on the open file — it would comment on both of them with something like "The move aims to get a Rook on open file. The move allows the opponent to get a Rook on open file". The expert system removes such comments provided that the complementary features are worth about the same amount (by directly comparing their respective values in the vector of differences)[2].

The features sometimes have a different meaning depending on the context as defined by other features. The expert system detects which meaning is appropriate in a given situation and modifies the raw comment accordingly. Let us explain this by an example. Assume that the commenting module flagged the feature WHITE_BISHOP_PLUS_PAWNS_ON_COLOR as a characteristic that changed positively for White. This feature is designed to modify (reduce) the value of a Bishop if there are own Pawns on the squares of the same color that this Bishop

[2] The complementary features are not always worth the same, for example, White's bishop pair can be better than Black's bishop pair. That is why the expert system has to take their worth into account when attempting to remove comments based on complementary features.

moves on. There are two very different cases when this feature can change positively. In the first case, we improve the Bishop by moving some Pawns thus creating more space for the Bishop. In this case the raw comment "The move aims to improve Bishop plus Pawns with respect to color" is slightly awkward but is still understandable. However, in the second case this (bad) Bishop is simply exchanged. Now the raw comment is completely out of place; though in essence correct if one knows the hidden meaning. By looking at other features (in this case whether the Bishop is exchanged) the expert system knows what commentary should be given; in this case "Bad Bishop was exchanged".

The third role of the expert system is to combine several elementary features into higher-level, structured features that represent concepts understandable to humans. Let us take a look at one example rule in the expert system that does so:

```
if (BLACK_BISHOPS = 2) then
    if (BLACK_BISHOPS_POSITION + BLACK_BISHOPS_MOBILITY +
        BLACK_BISHOPS_BLOCK_CENTER + BLACK_BISHOP_PAIR +
        BLACK_BISHOP_KING_SAFETY <= -65) then
            comment("Black has an active bishop pair.")
```

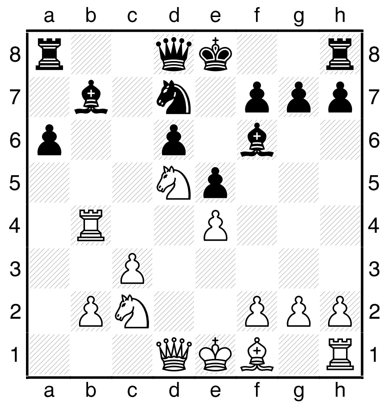
This rule combines six elementary features to derive whether Black has an active bishop pair (higher-level concept). As we can see the bishop pair can be active for various reasons. It can simply be that the Bishops have great mobility. Or it can be that the Bishops guard the King well and at the same time are well positioned. The raw comments for every elementary aspect of the position are less instructive in this case than a single high-level comment. Even more, not all elementary features need to change positively for the same side. In this case the raw comments might even be confusing. The high-level comment "Black has an active bishop pair" is usually preferable to commenting separately on the good positions of the Bishops and how the Bishops defend the King well. Moreover, the commentary can easily be extended to explain why the bishop pair is active — simply state which parts of the rule contribute (most) to the score.

Our expert system consists of about 25 rules. The rules were constructed by chess experts[3] who also set the threshold values in the rules (cf. -65 in the above example rule). The expert system can easily be extended with new rules or by refining the existing ones. Further refining and in particular fine-tuning of the thresholds can perhaps also be accomplished by machine learning methods.

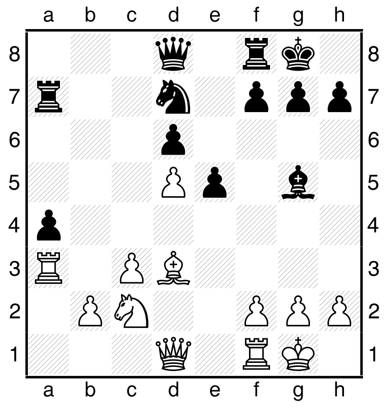
4 Some Tutoring Examples

In the previous sections we explained the ideas behind our tutoring/annotating mechanism. Now we shall demonstrate how they work in practice on some positions taken from grandmaster games. The positions are taken from John Nunn's tutoring book [5].

[3] The chess expertise was provided by Matej Guid and WIM Jana Krivec who are both rated around 2300 ELO.



(a)



(b)

Fig. 2. Kasparov versus Shirov, Horgen, 1994

Let our tutoring system work on the position from the game Kasparov versus Shirov shown in Fig. 2a. It, or rather CRAFTY, evaluates the position after 16. ... Nc5 as slightly worse for Black (+0.43). In its place it recommends the move 16. ... Ra7, which, it suggests, would lead to an equal position (−0.01) after 17. Bd3 O-O 18. O-O a5 19. Rb3 Bxd5 20. exd5 a4 21. Ra3 Bg5. The position after 21. ... Bg5 is our envisioned position and is shown in Fig. 2b.

Table 1. Vector of differences

Feature (positive change)	Score	Feature (negative change)	Score
black bishop pair	20	king tropism	−51
black back rank	12	evaluate pawns	−27
black bishop plus pawns on color	12	white knights outposts	−15
white bishops mobility	12	white back rank	−12
white bishops position	10	white knights centralization	−6
black bishops mobility	6		

Table 1 shows the vector of differences for the 16. ... Ra7 variation. Let the tutor comment from Black's perspective. The positive changes for Black are listed on the right side of the table, and negative changes for Black on the left side of the table. The raw commentary would thus be "The move 16. ... Ra7 aims to improve king tropism, improve pawn structure, eliminate opponent's knight outposts, weakens opponent's back rank, reduce opponent's knights centralization. The move allows the opponent to eliminate your bishop pair, weaken your back rank, weaken your Bishop plus Pawns on same color, increase the Bishops' mobility, improve the Bishop's position, decrease your Bishop's mobility". As we can see, the raw commentary is directly translated from the vector of differences. Let us now take a look how the expert system for commentary refinement improves the raw comments.

First of all, the back rank weakness comments are eliminated as they are worth the same (for their respective sides) and are complementary. Next, from material features it is observed that Black lost the bishop pair. However, truly interesting are the rules below.

```
if (WHITE_ACTIVITY_CHANGE + BLACK_ACTIVITY_CHANGE <= -10) then
        comment("Black has improved the activity of pieces")
if (WHITE_BISHOPS = 1 and WHITE_BISHOPS_MOBILITY >= 12 and
    WHITE_BISHOPS_POSITION >= 6) then
        comment("White has improved the bishop")
if (abs(KING_TROPISM + EVALUATE_KING_SAFETY) > 60) then
        comment("White has the initiative against black's king")
if (EVALUATE_PAWNS <= -10)
        comment("Black has improved the pawn structure")
if (WHITE_KNIGHTS_CENTRALIZATION + WHITE_KNIGHTS_OUTPOSTS <= -14)
        then comment("White no longer has a strong knight")
```

These rules achieve the following. The first one combines two elementary features to detect that Black has improved the activity of pieces. The second rule again combines three elementary features to observe that White's Bishop has improved. The third rule actually removes the king tropism raw comment, because it should be combined with King's safety and the required threshold for the initiative against the King was not reached. The fourth rule is simple (and should be refined with further work) and observes that Black improved the pawn structure. The last rule combines two elementary features to note that White lost (exchanged) a strong Knight. After applying these rules the refined commentary is: "Black has improved the pawn structure, White no longer has a strong Knight, and Black has improved the activity of his pieces. On the other hand: Black no longer has the advantage of a bishop pair, White has improved the Bishop." We can see that the refined commentary, using higher-level concepts, is much better than the original raw commentary.

A second position is from the famous game Short versus Timman. In the position of the diagram in Fig. 3a Short played 17. Bc4 and GM John Nunn in his book agrees with him [5]: "White is not deflected by the prospect of winning a Pawn with 17.Bxd5 exd5 18.Qxd5 Be6, when Black would have reasonable drawing chances in view of his good development and active Bishops." The tutor, however, prefers 17. Bxd5 exd5 18. Qxd5 Be6 19. Qb5 Qxb5 20. axb5 Rfe8 21. Nd4 Bd5 22. Bf4 (position in Fig. 3b) and backs it up with: "White has won a Pawn. Black has gained a bishop pair, Black has improved the Bishops". In essence, the tutor agrees that Black has reasonable compensation for the Pawn.

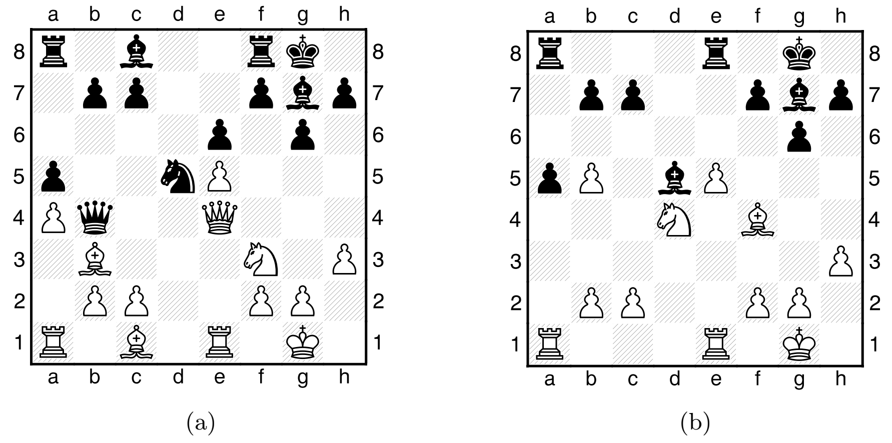


Fig. 3. Short versus Timman, Tilburg, 1991

5 Conclusions and Future Work

The work presented in this paper is intended as a founding step towards turning a chess engine into a competent chess tutor and annotator. Even at this early stage of development, the core mechanism exhibited that it is able to generate rather intelligent commentary as demonstrated by the examples shown. The main advantage of our approach over the related proposals is that it could be, and consequently was, applied to the complete domain of chess. The advantage of our program over various annotating programs is that our program is able to understand and comment on tactical positions, and also shows a solid understanding[4] of positional intricacies of positions. Hence, it is able to provide adequate commenting on these two subdomains.

An interesting side-effect of our approach is that it can potentially prove useful to chess engine programmers as they can now practically see how their programs "think".

There is an array of cognitive issues still to be solved to arrive at really good tutoring system, e.g., (1) when to comment and when not, (2) which move (the best or some similar but better) to compare the move to, (3) the most suitable depth of lookahead, to name but a few. These are obvious tasks for future work.

The system's knowledge is easily extendable, either by employing a more knowledge-oriented chess engine or by enriching the expert system with new concepts. Also, the functionality of the system can be extended by adding various functions, e.g., implementing the ability to ask the tutor questions, but this is more a matter of engineering than of scientific work.

[4] Of course, positional understanding of the tutor is limited with the level of positional understanding as exhibited by today's chess programs — e.g., long-term planning is still absent.

References

1. The Board of the ICCA. The Best Annotation Award for 1993. *ICCA Journal*, 17(2):106–108, 1994.
2. D. Gadwal, J. E. Greer, and G. I. McCalla. Tutoring Bishop-Pawn Endgames: An Experiment in using Knowledge-Based Chess as a Domain for Intelligent Tutoring. *Applied Intelligence*, (3)3:207–224, 1993.
3. H. Herbeck and W. Barth. An Explanation Tool for Chess Endgames Based on Rules. *ICCA Journal*, 19(2):75–82, 1996.
4. D. Levy and T. Marsland. The ICCA Best Annotation Award for 1995. *ICCA Journal*, 19(2):135–136, 1996.
5. J. Nunn. *Understanding Chess Move by Move*. Gambit Publications Limited, 2001.
6. R. Seidel. Self-annotating Elementary Endgames. *ICCA Journal*, 17(2):51–62, 1994.