# Searching over Metapositions in Kriegspiel

Andrea Bolognesi and Paolo Ciancarini

{abologne,cianca}@cs.unibo.it

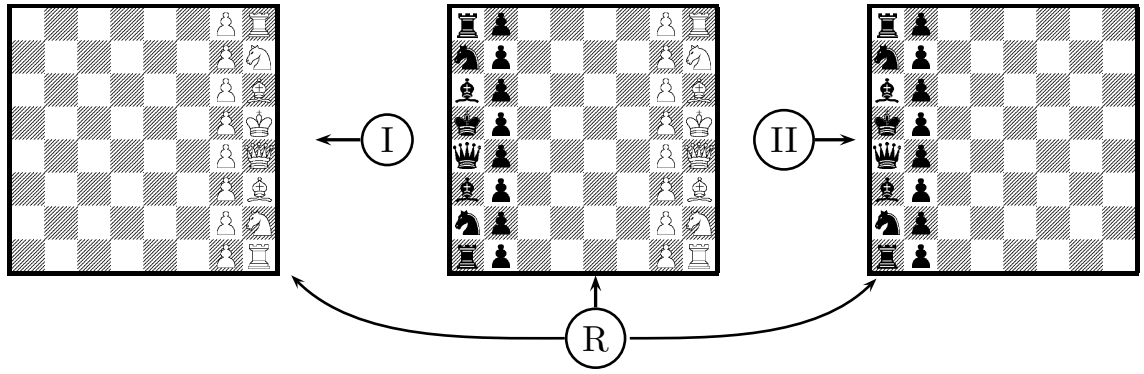
Dipartimento di Scienze dell'Informazione
University of Bologna - Italy

# Index

# Perfect vs imperfect information

- Games of *perfect* information:
  - the current state of the game is fully accessible to each player;
  - the uncertainty is all about future moves;
  - Chess, Checkers.
- Games of *imperfect* information:
  - players have partial knowledge about the current state of the game;
  - a player may not know what any other player has done up to that moment;
  - Poker, Kriegspiel.

# The game of Kriegspiel




- Kriegspiel is a game for two players, but it needs a referee;
- Neither player knows any position of his/her opponent's pieces,
- he/she tries to guess his/her opponent's state of play by choosing moves to which the referee can reply by
  - being *silent (S)*;
  - saying *"illegal" (I)*;
  - saying *"check" (C)*.

# Research Highlights


- M. Leoncini and R. Magari. *Manuale di Scacchi Eterodossi.* Tipografia Senese, Siena,1980.
- J. Boyce. A Kriegspiel Endgame. In D. Klarner editor, *The Mathematical Gardener*, pages 28-36. Prindle, Weber & Smith, 1981.
- T. Ferguson. Mate with Bishop and Knight in Kriegspiel. *Theoretical Computer Science*, 96:389-403, 1992.
- T. Ferguson. Mate with two Bishops in Kriegspiel. Technical report, UCLA, 1995.
- P. Ciancarini, F. Dalla Libera and F. Maran. Decision Making under Uncertainty: A Rational Approach to Kriegspiel. In J. van den Herik and J. Uiterwijk editors, *Advances in Computer Chess 8*, 1997.
- M. Sakuta and H. Iida. Solving Kriegspiel-like Problems: Exploiting a Transposition Table. *ICCA Journal*, 23(4):218-229, 2000.

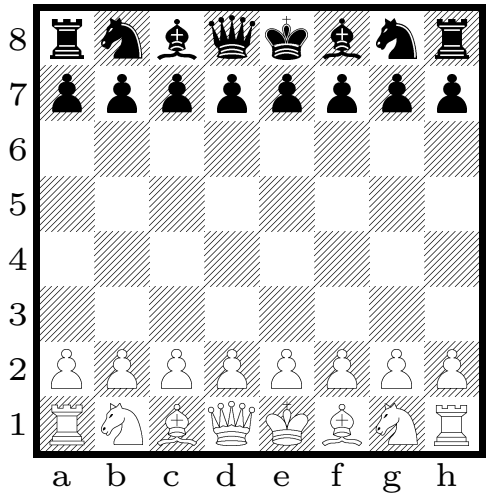
# Example of Kriegspiel game

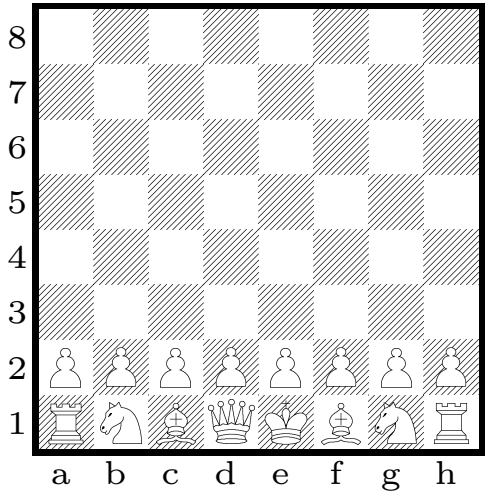
The following animation shows an example of Kriegspiel game played by Paolo Ciancarini agains Krieg on ICC in April 2003.

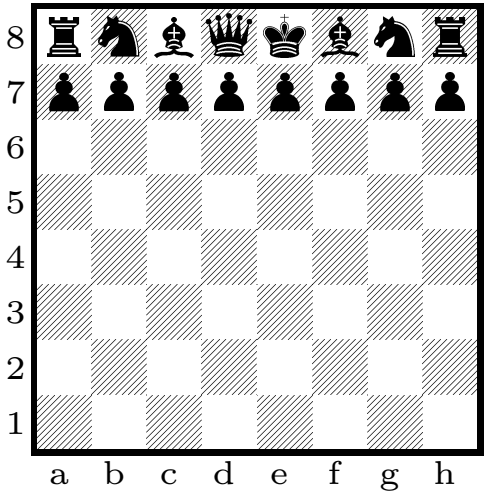
Click here to skip the animation.

SKIP

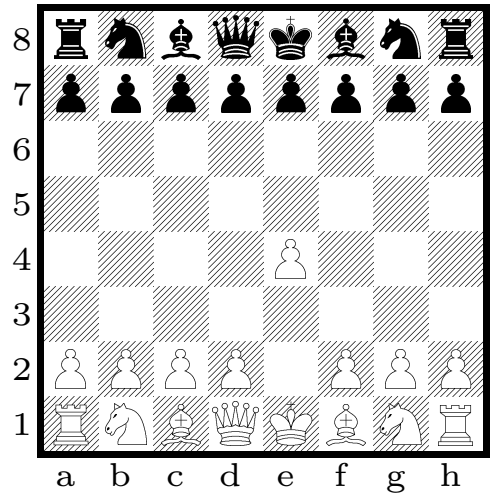
# PaoloC-Krieg, ICC Aprile 2003



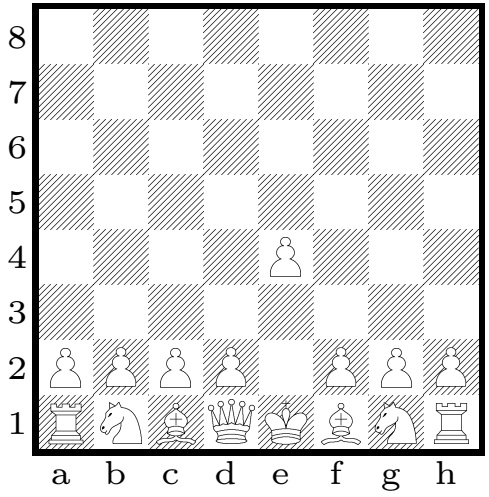


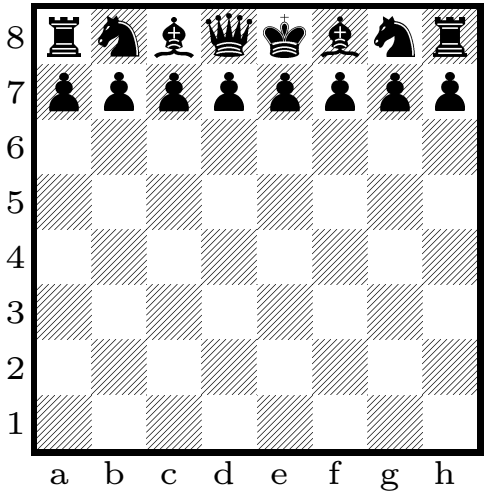


# PaoloC-Krieg, ICC Aprile 2003

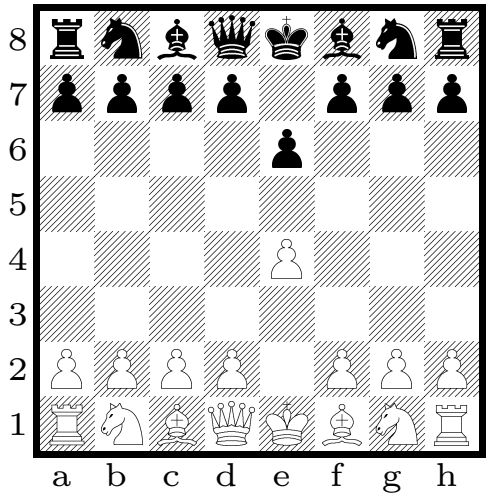


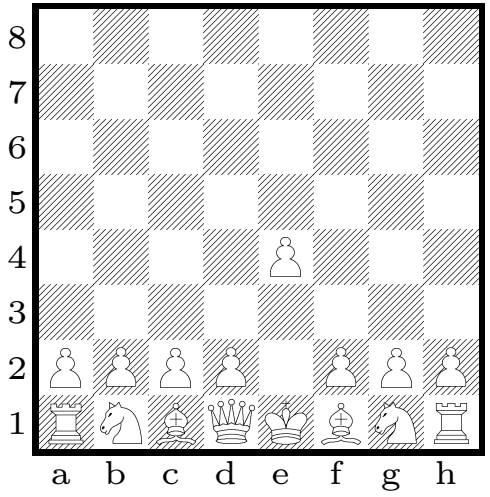
1.e4



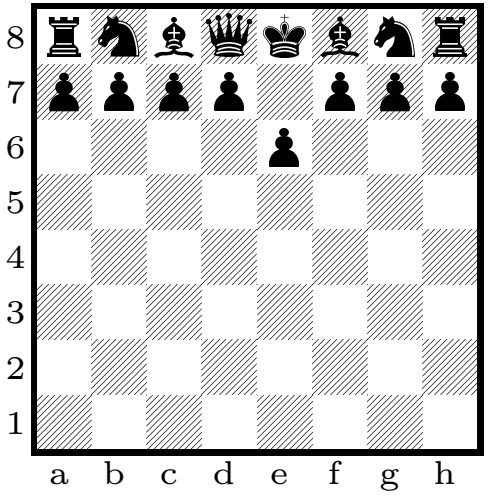


# PaoloC-Krieg, ICC Aprile 2003

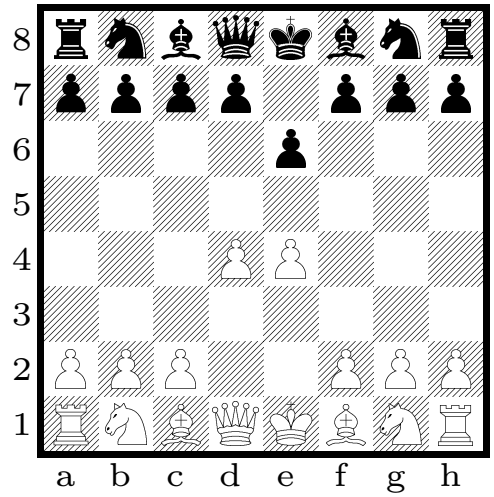




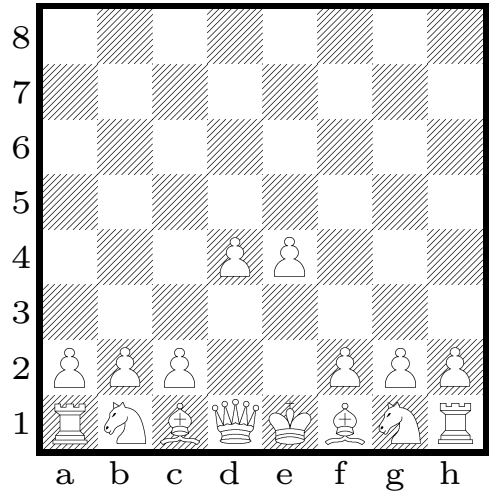
1. ..e6

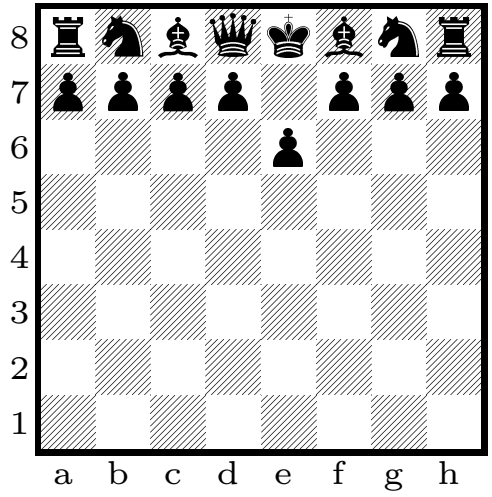


# PaoloC-Krieg, ICC Aprile 2003

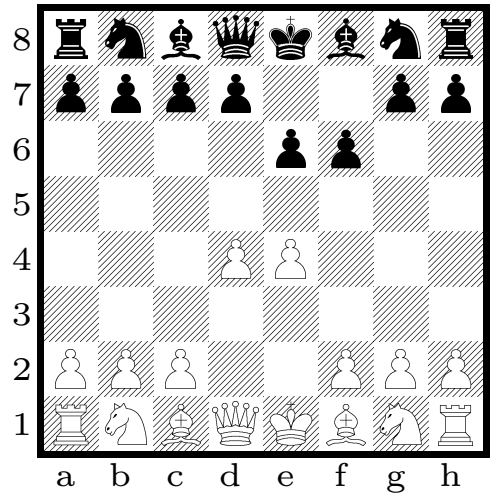


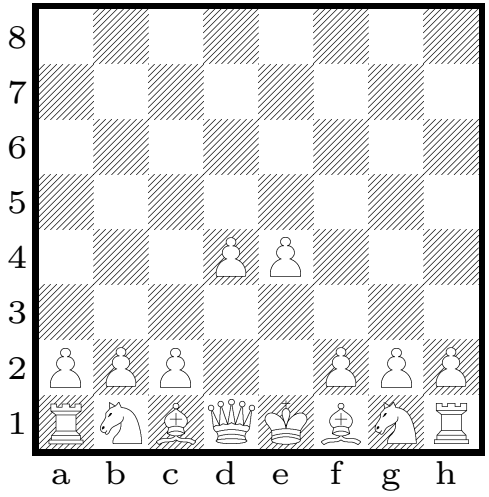
2.d4



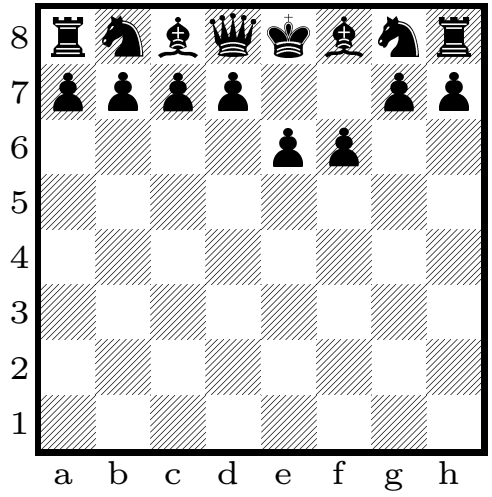


# PaoloC-Krieg, ICC Aprile 2003

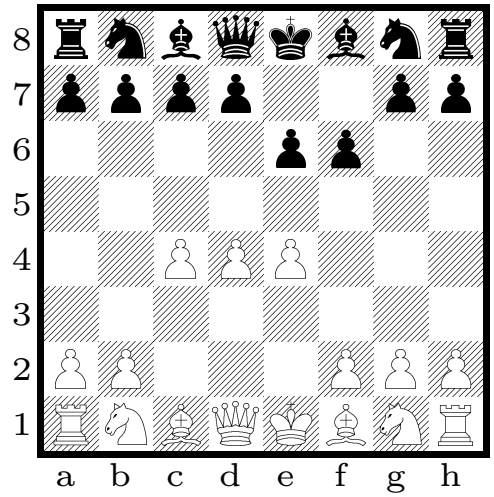




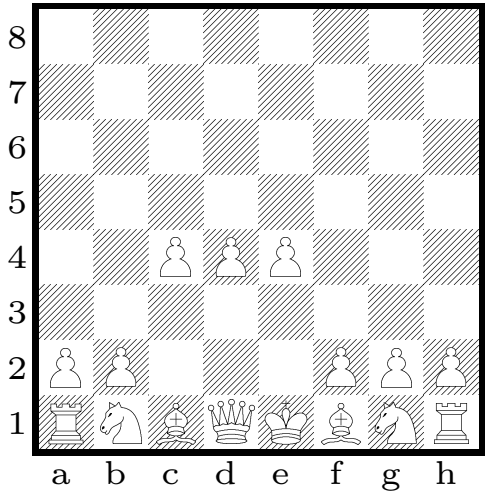
2. ..f6

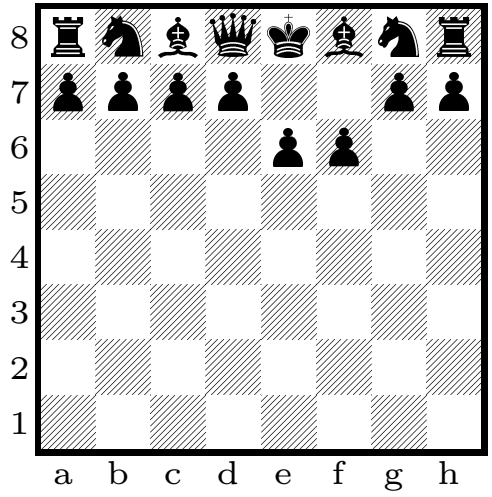


# PaoloC-Krieg, ICC Aprile 2003

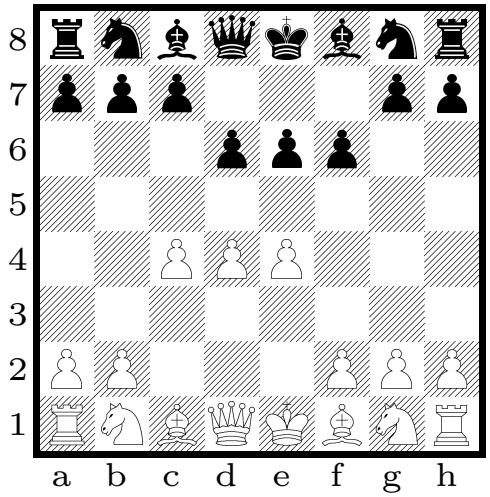


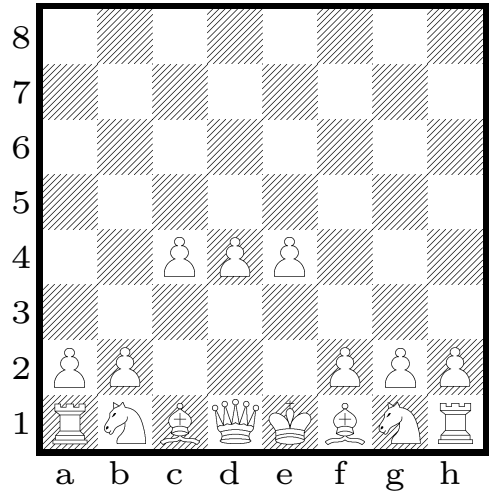
3.c4



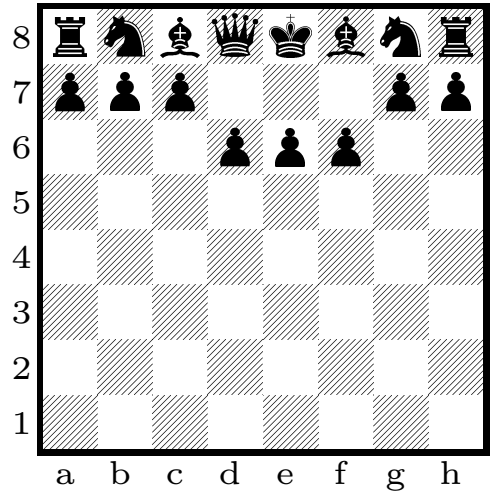


# PaoloC-Krieg, ICC Aprile 2003

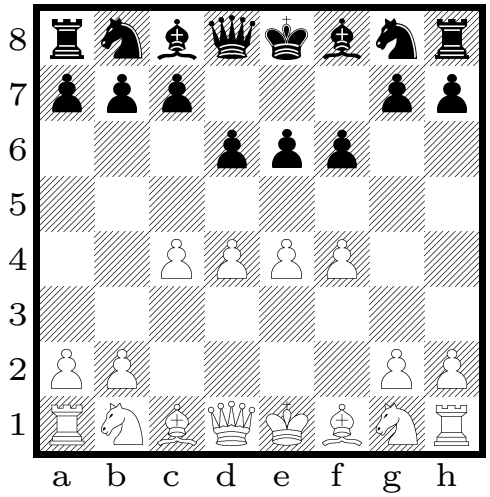




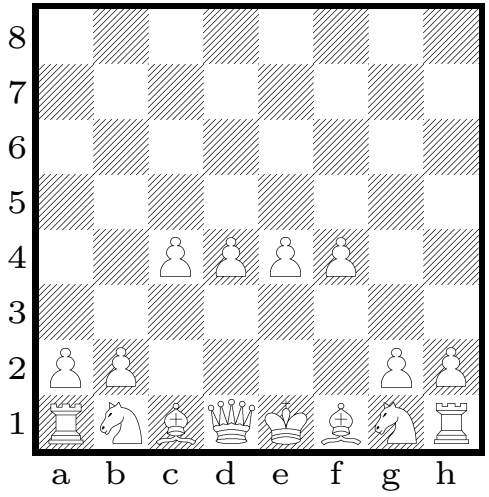
3. ..d6

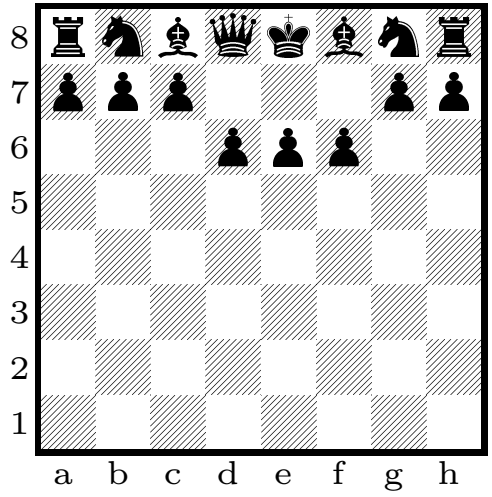


# PaoloC-Krieg, ICC Aprile 2003

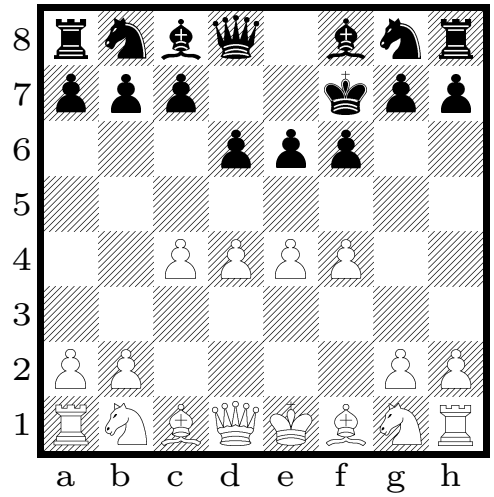


4.f4

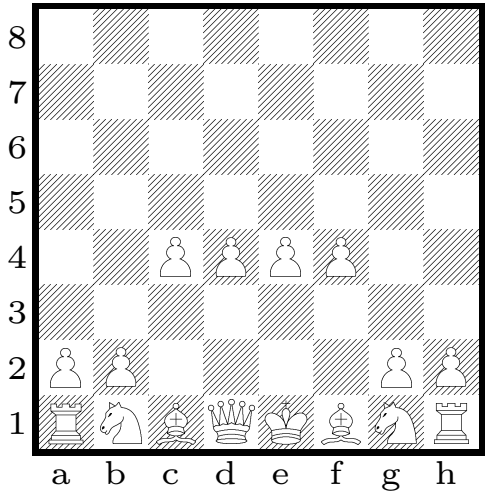


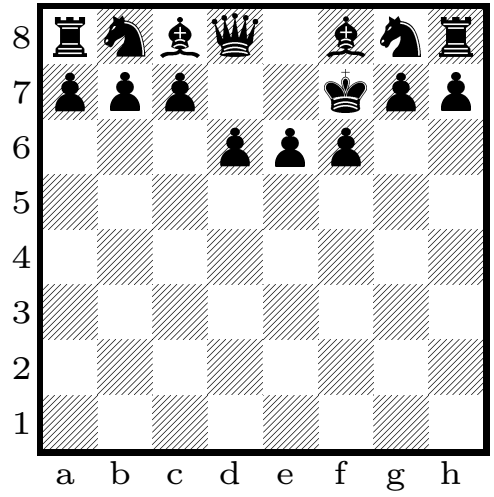


# PaoloC-Krieg, ICC Aprile 2003

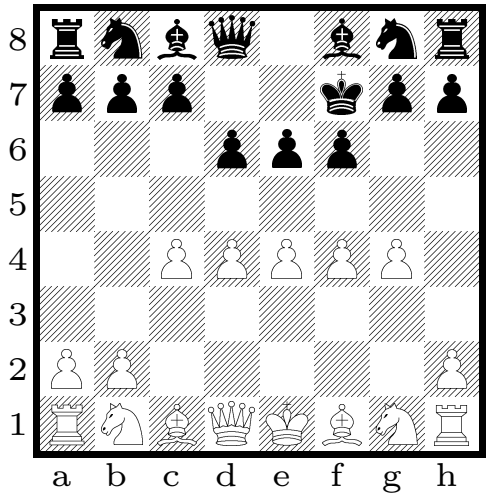


4. ..Kf7

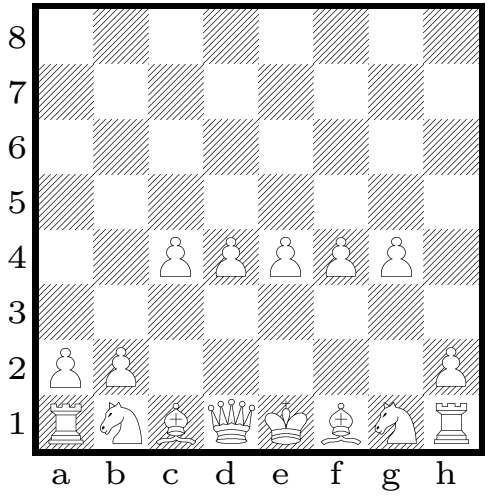


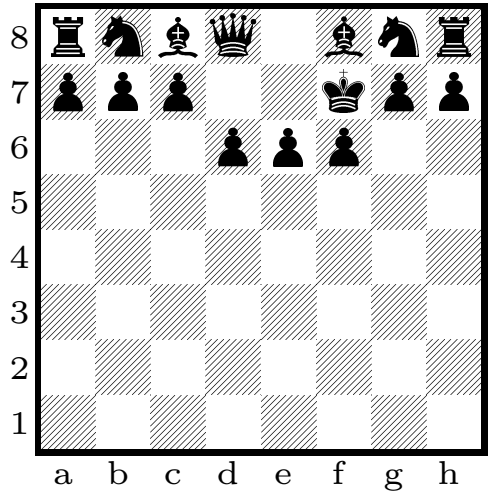


# PaoloC-Krieg, ICC Aprile 2003

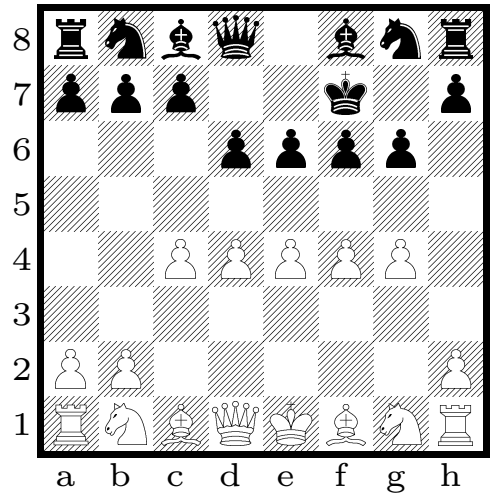


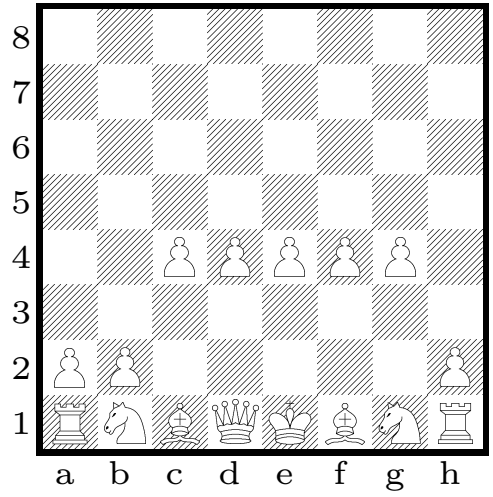
5.g4



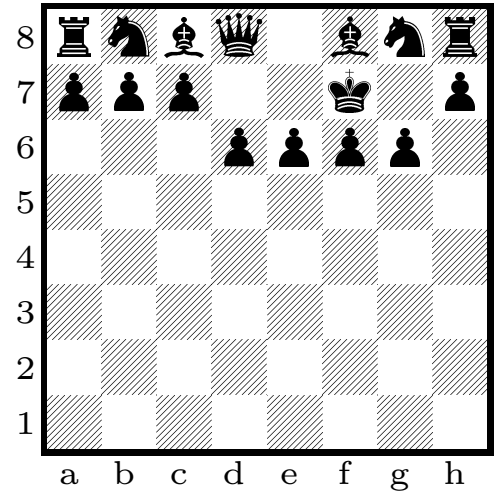


# PaoloC-Krieg, ICC Aprile 2003

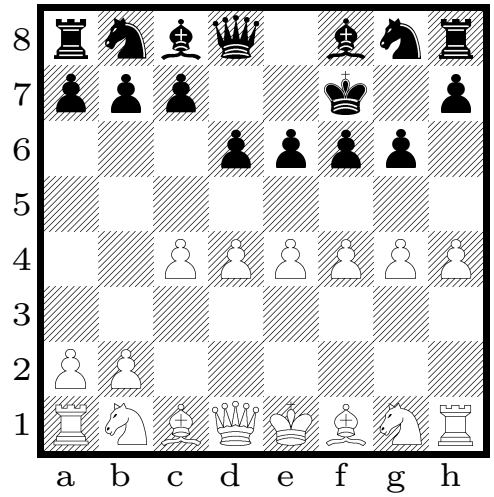




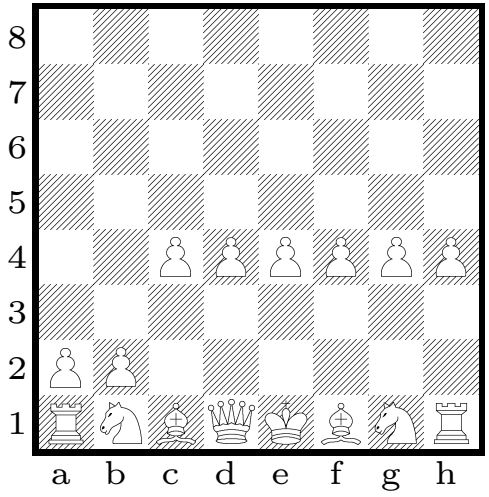
5. ..g6

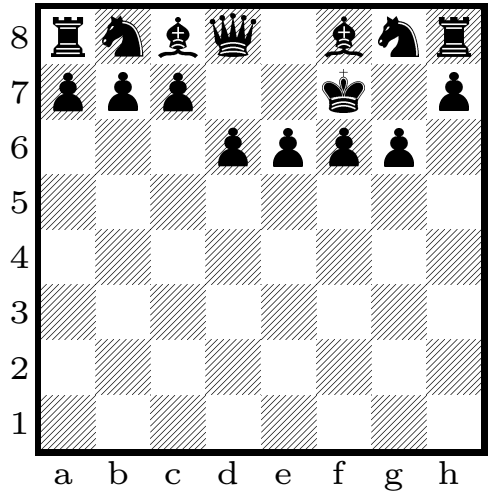


# PaoloC-Krieg, ICC Aprile 2003

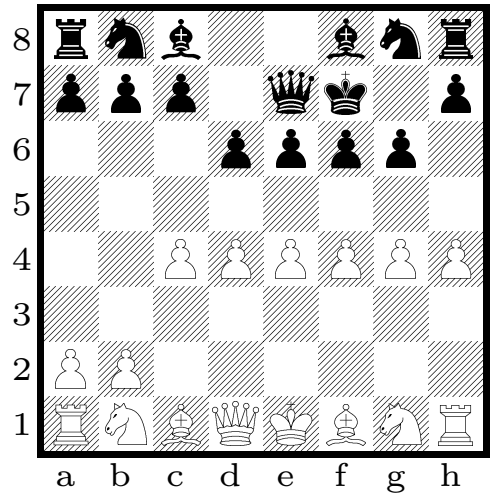


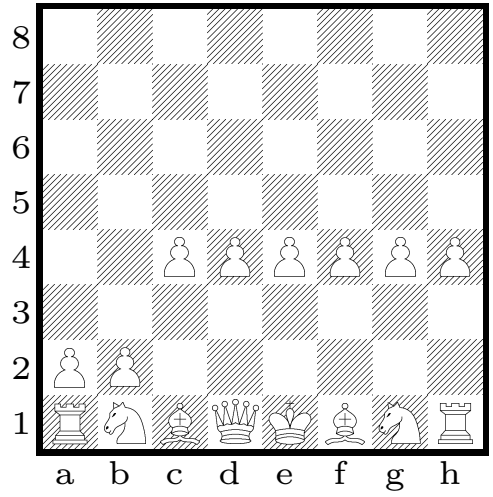
6.h4



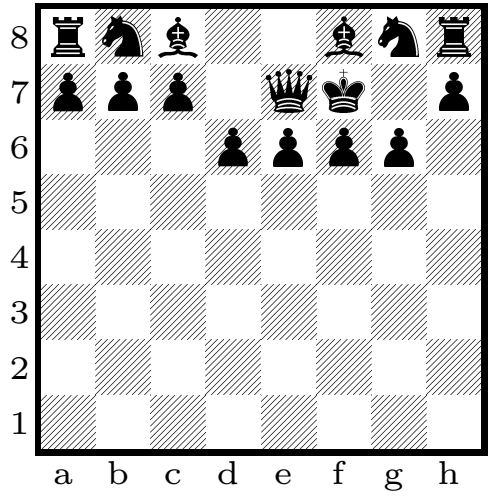


# PaoloC-Krieg, ICC Aprile 2003

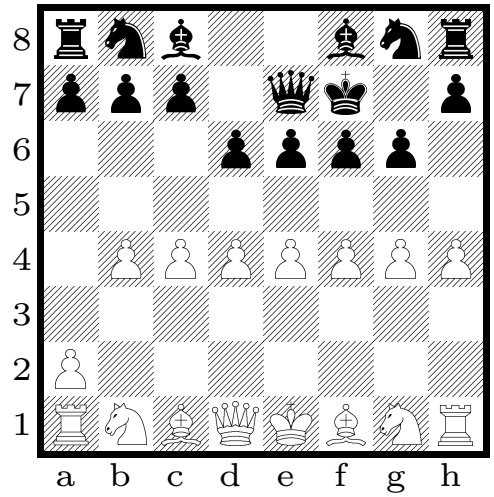




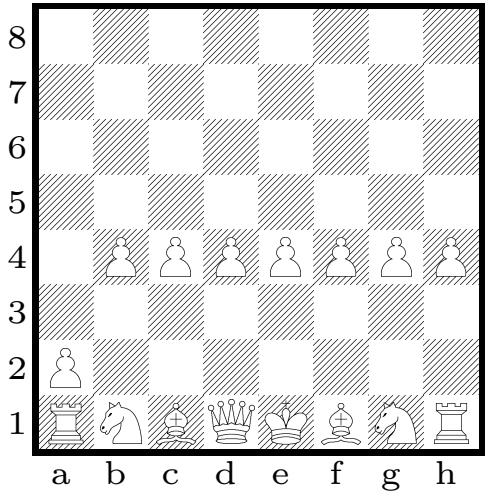
6. ..Qe7

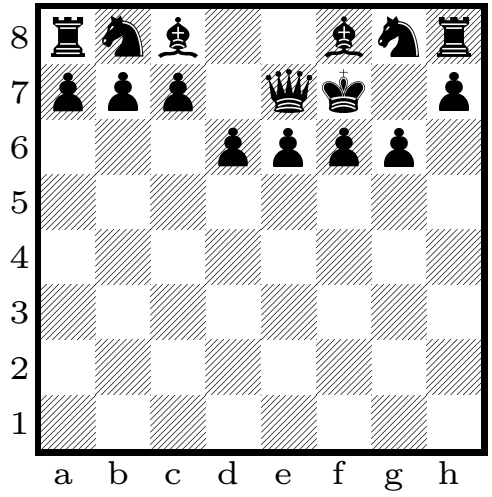


# PaoloC-Krieg, ICC Aprile 2003

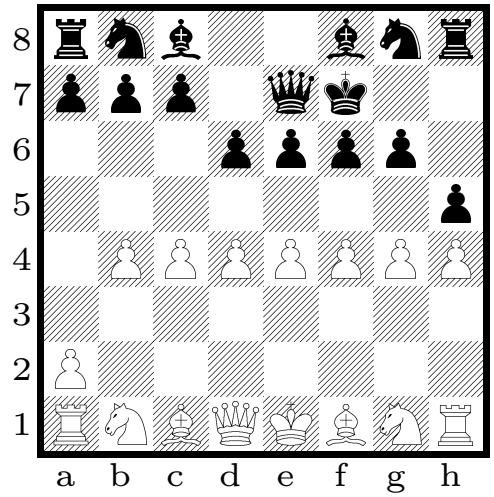


7.b4

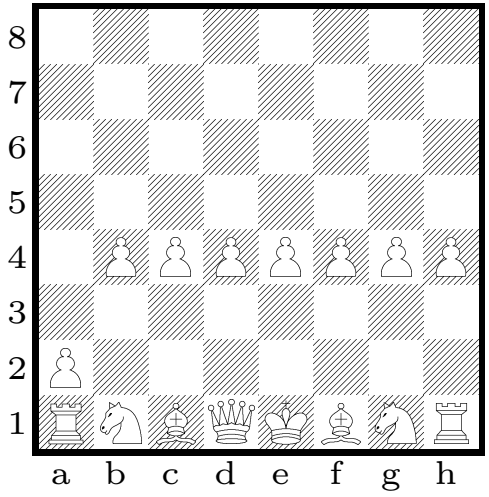


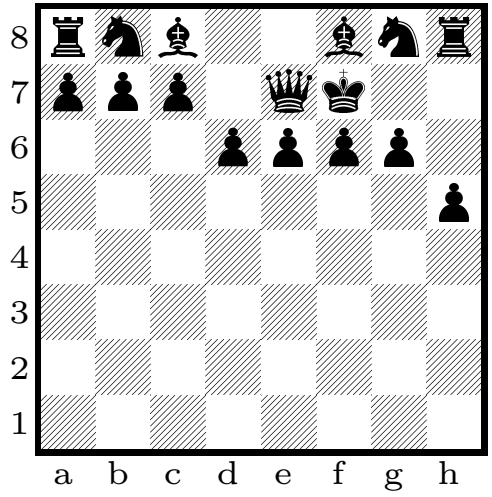


# PaoloC-Krieg, ICC Aprile 2003

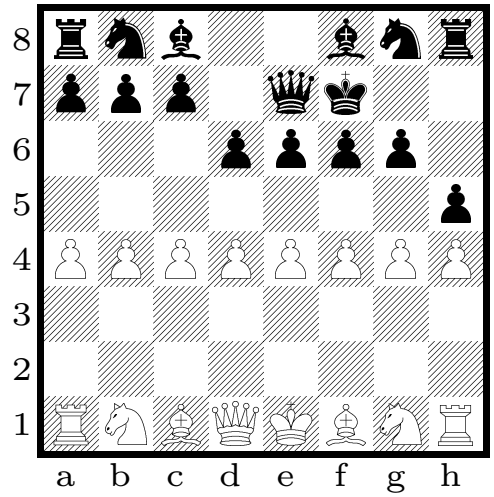


one pawn try 7. ..h5

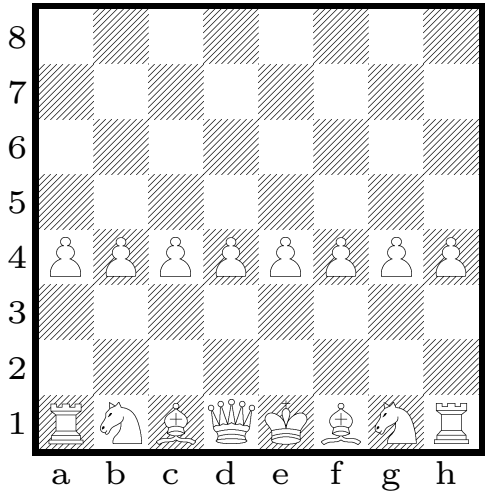


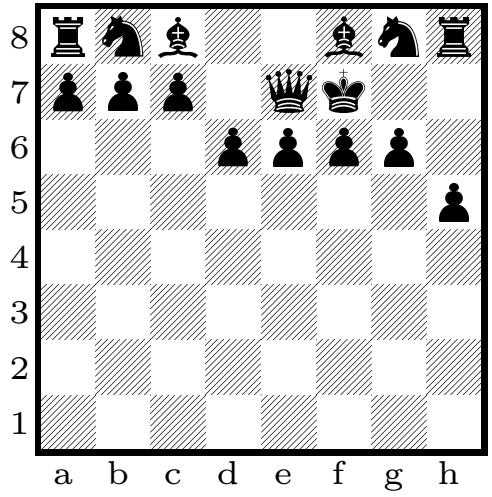


# PaoloC-Krieg, ICC Aprile 2003

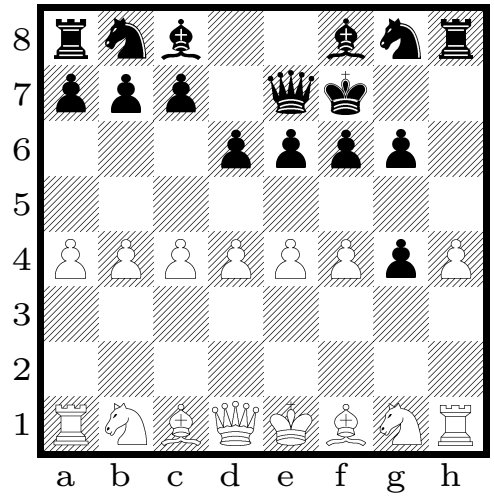


8.a4

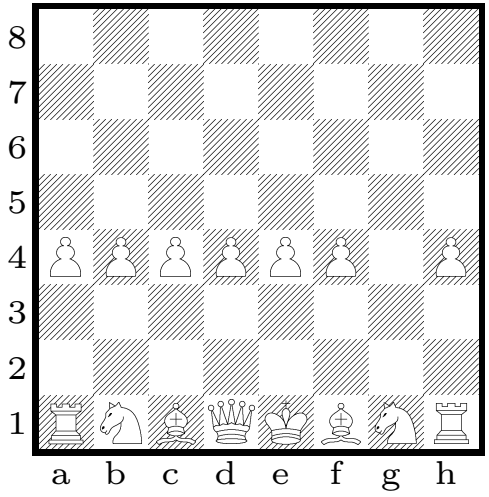


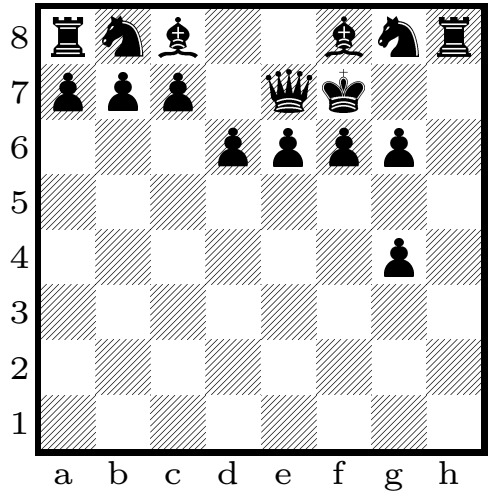


# PaoloC-Krieg, ICC Aprile 2003

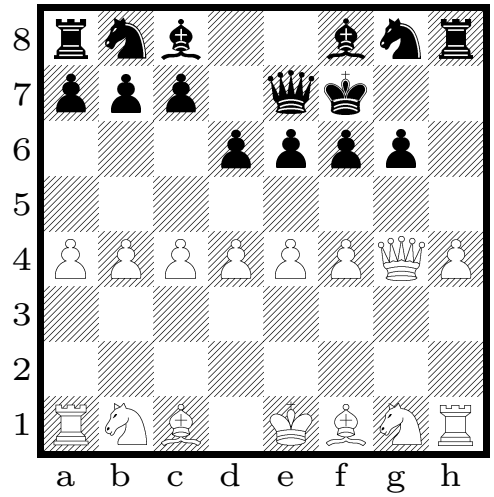


pawn capture in g4 8. ..h×g4

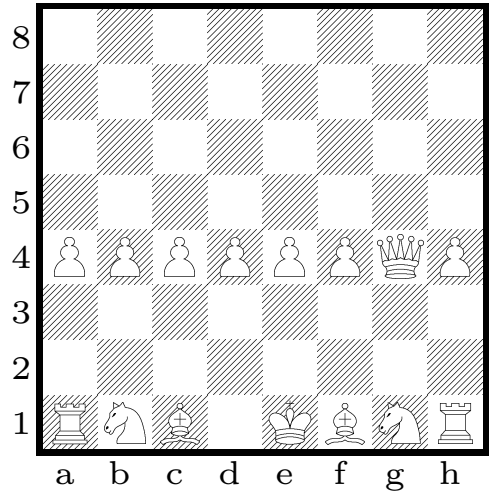


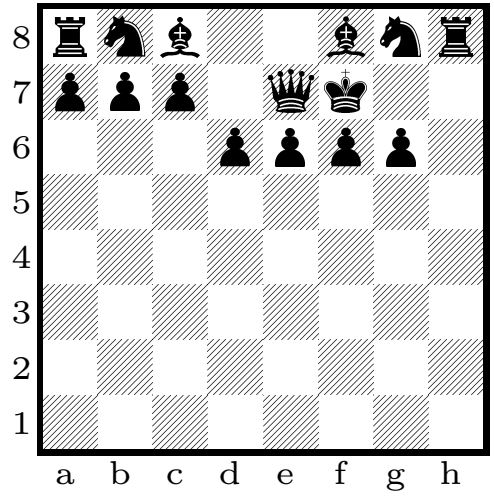


# PaoloC-Krieg, ICC Aprile 2003

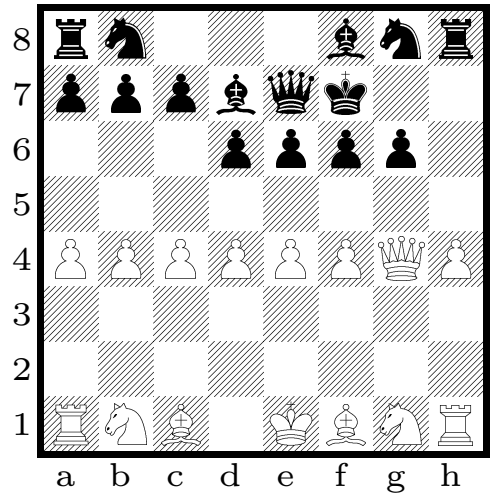


9.Q×g4 pawn capture in g4

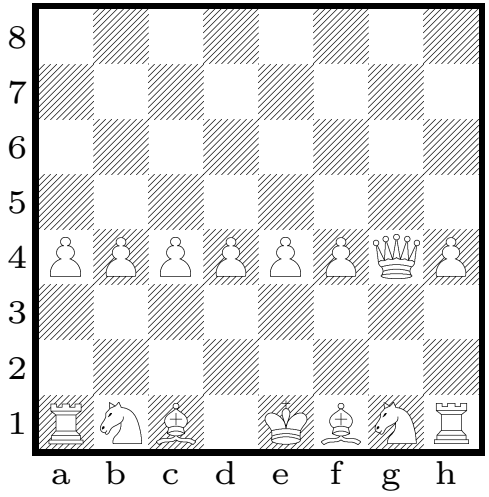


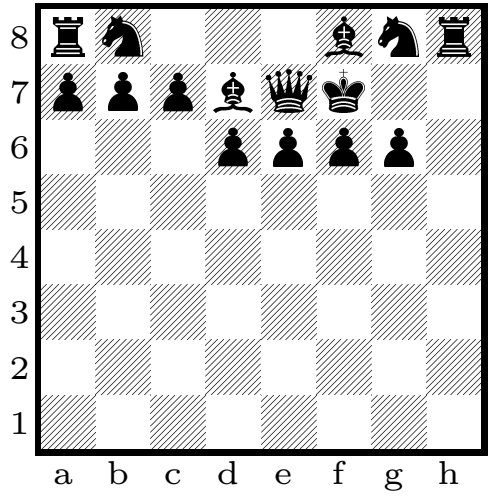


# PaoloC-Krieg, ICC Aprile 2003

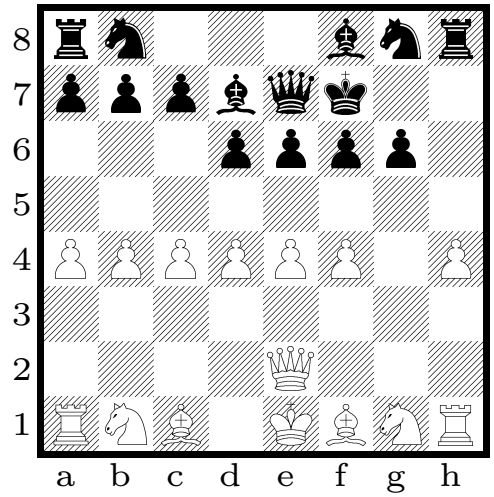


9. ..Bd7

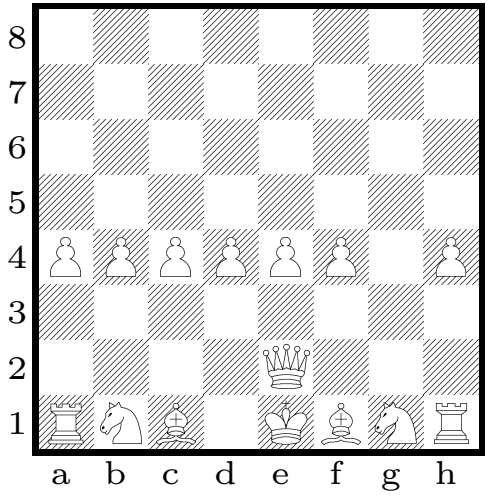


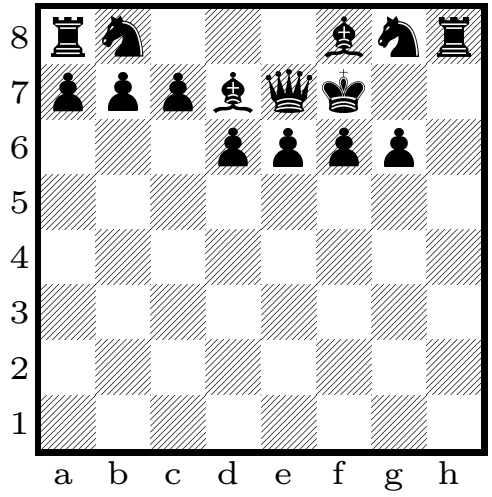


# PaoloC-Krieg, ICC Aprile 2003

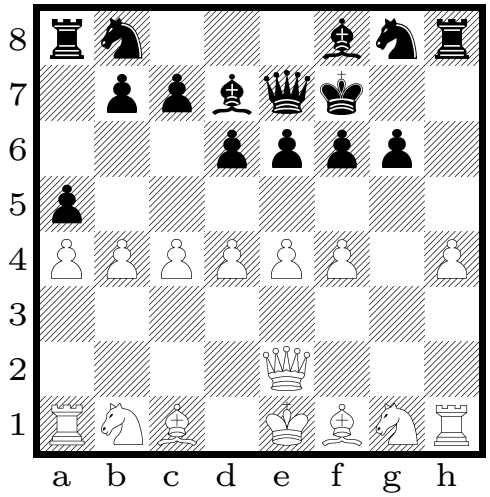


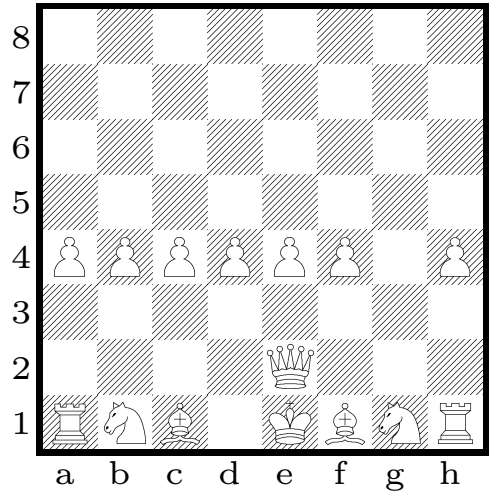
10.Qe2



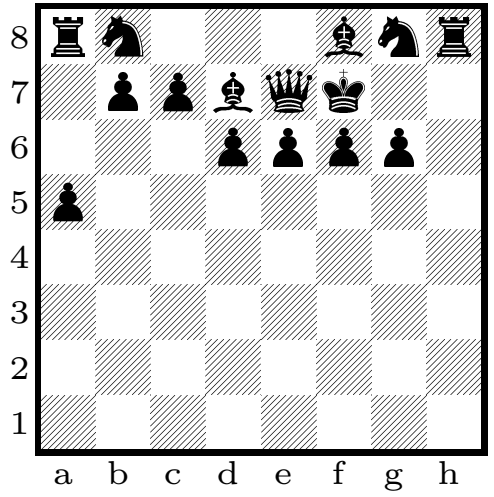


# PaoloC-Krieg, ICC Aprile 2003

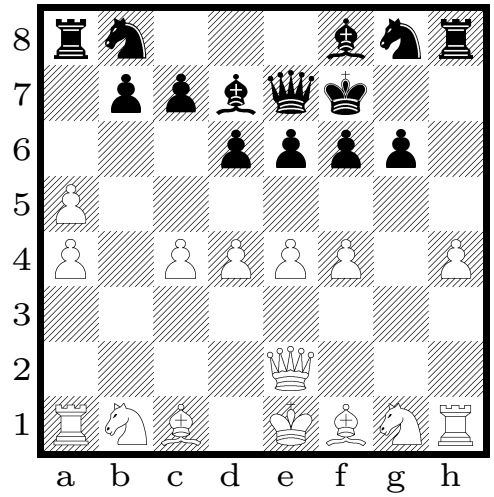




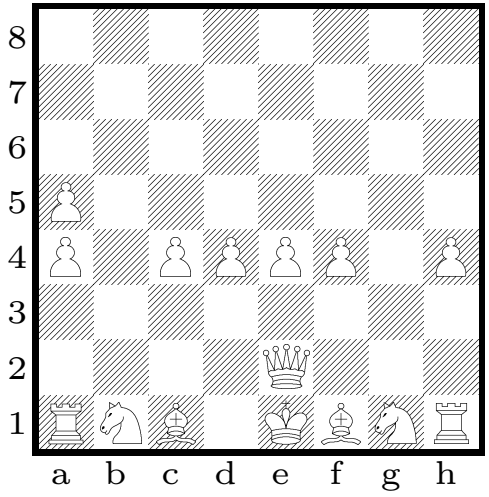
10. ..a5

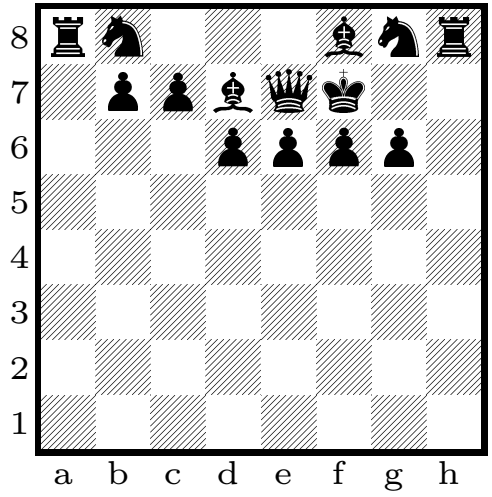


# PaoloC-Krieg, ICC Aprile 2003

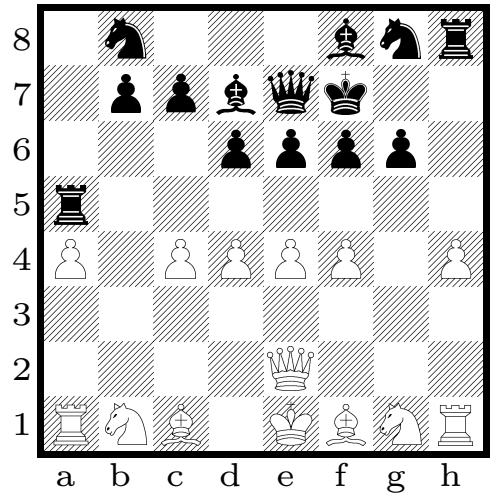


11.b×a5 pawn capture in a5

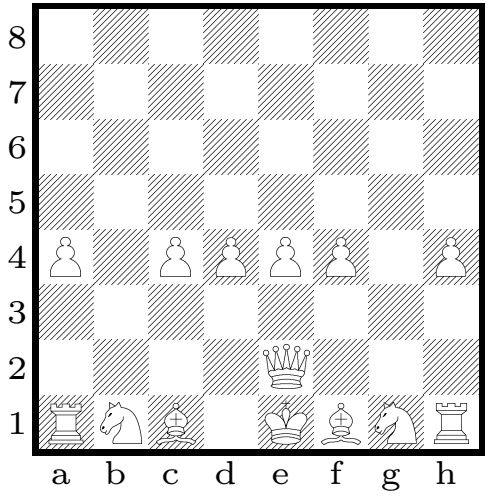


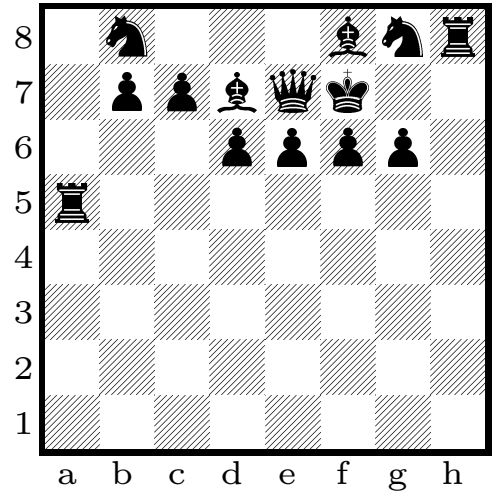


# PaoloC-Krieg, ICC Aprile 2003

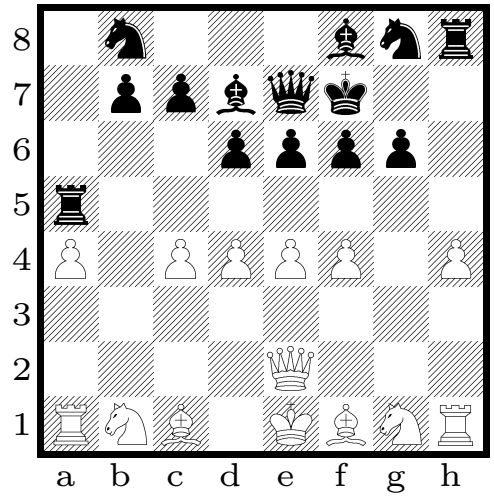


pawn capture in a5 11. ..R×a5

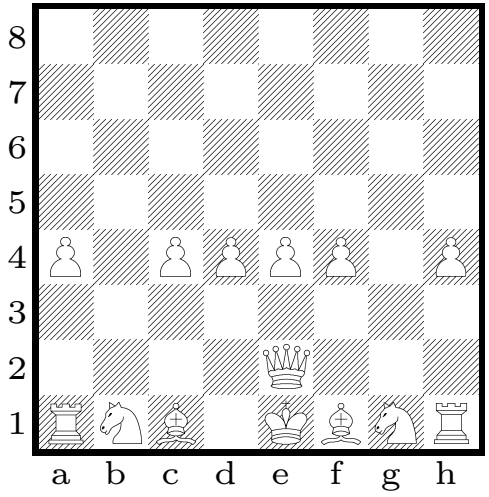


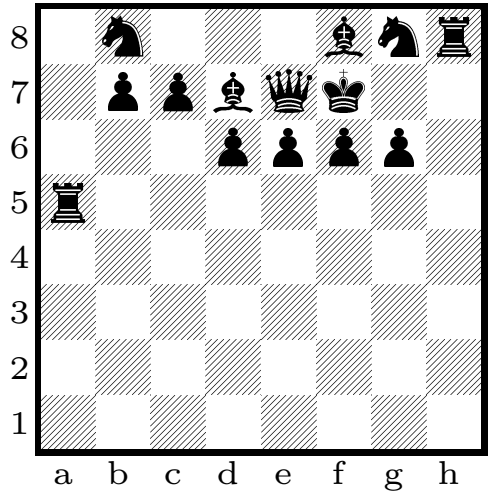


# PaoloC-Krieg, ICC Aprile 2003

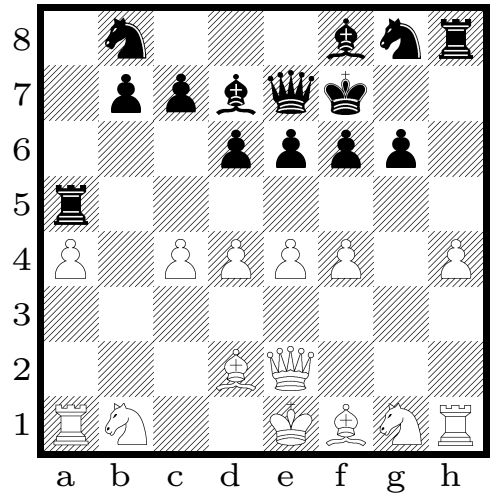


12.ab5 illegal

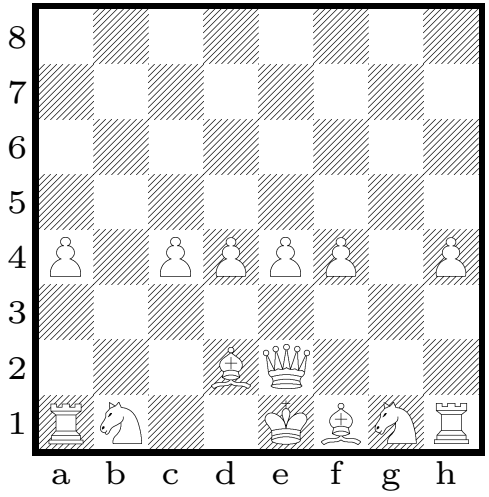


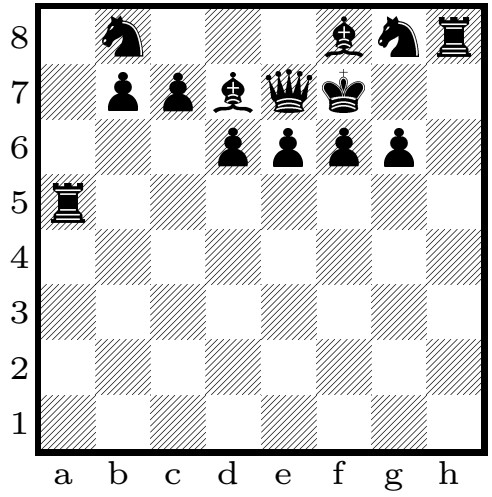


# PaoloC-Krieg, ICC Aprile 2003

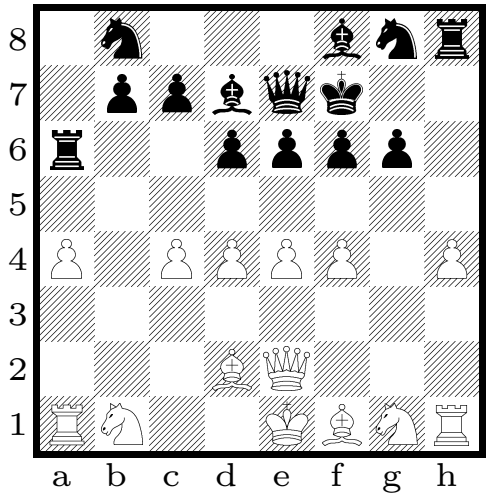


12.Bd2

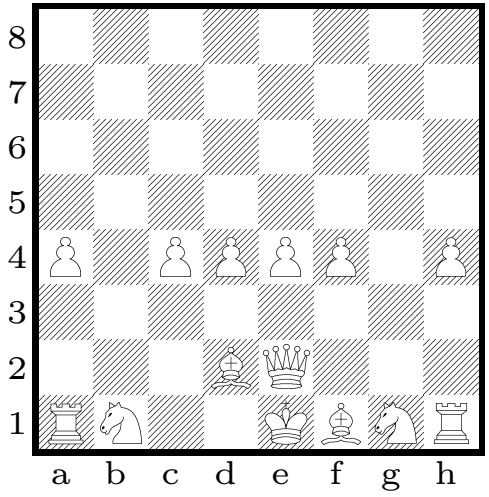


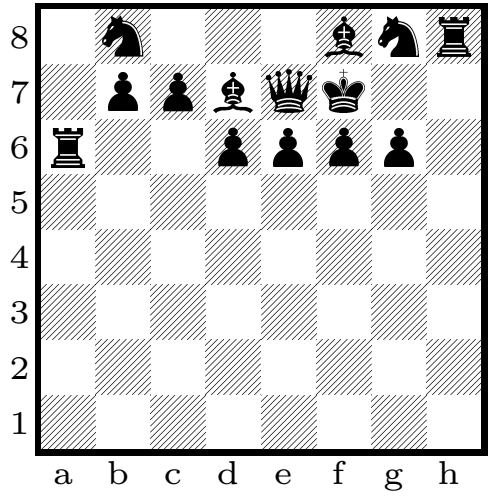


# PaoloC-Krieg, ICC Aprile 2003

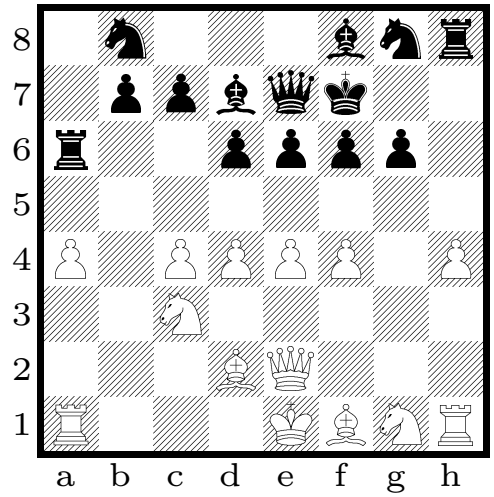


12. ..Ra6

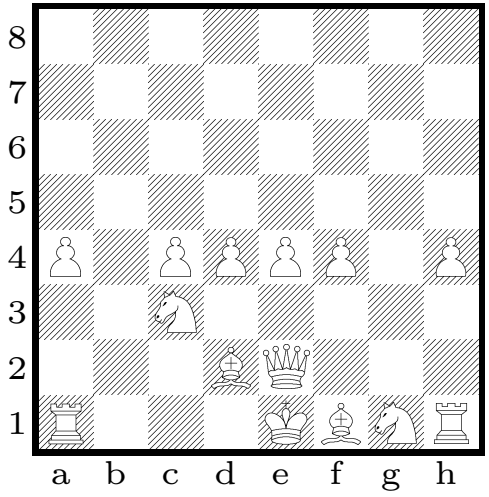


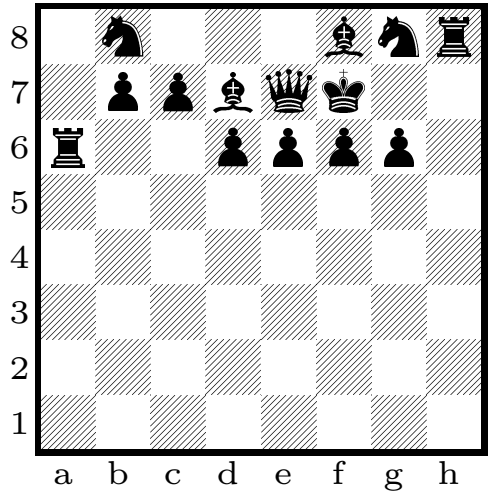


# PaoloC-Krieg, ICC Aprile 2003

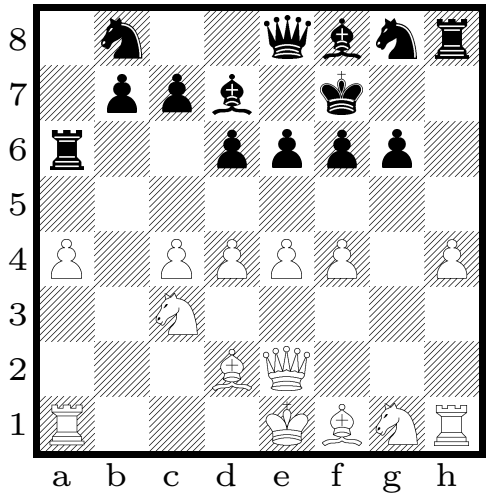


13.Nc3

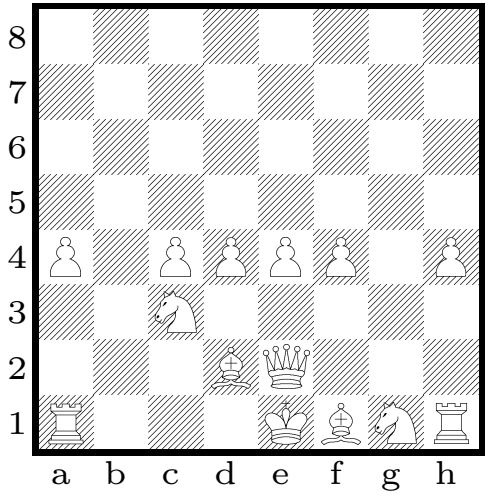


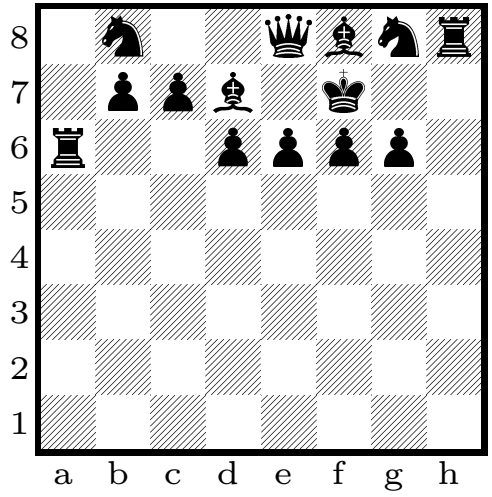


# PaoloC-Krieg, ICC Aprile 2003

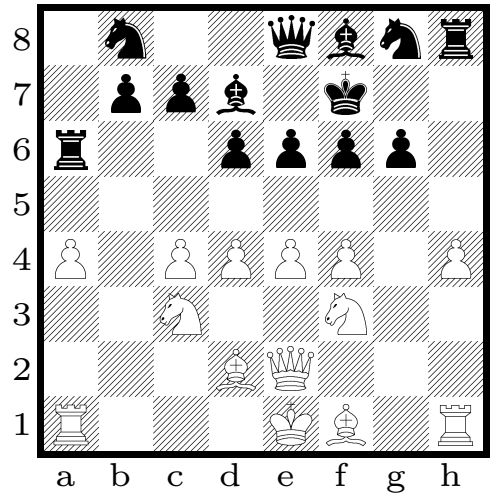


13. ..Qe8

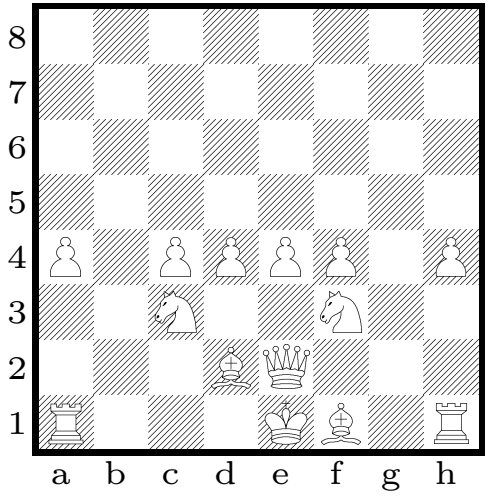


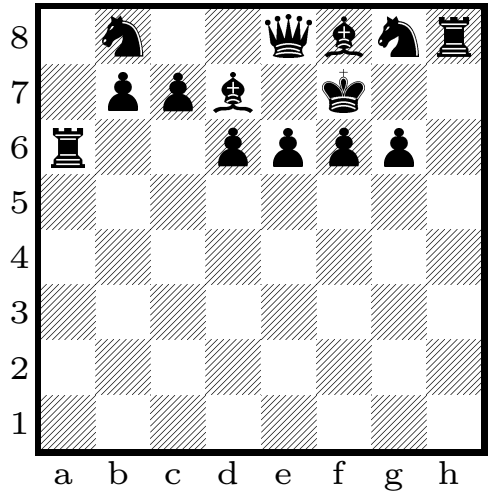


# PaoloC-Krieg, ICC Aprile 2003

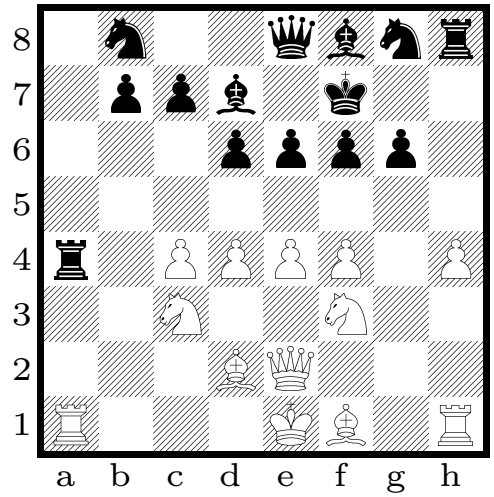


14.Nf3

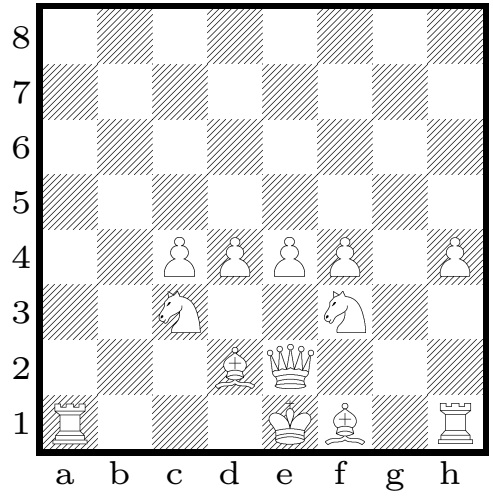


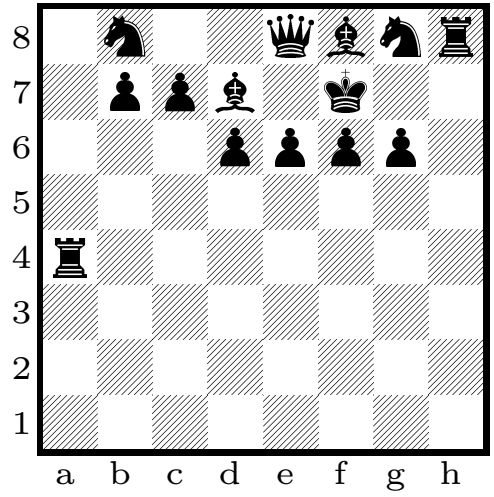


# PaoloC-Krieg, ICC Aprile 2003

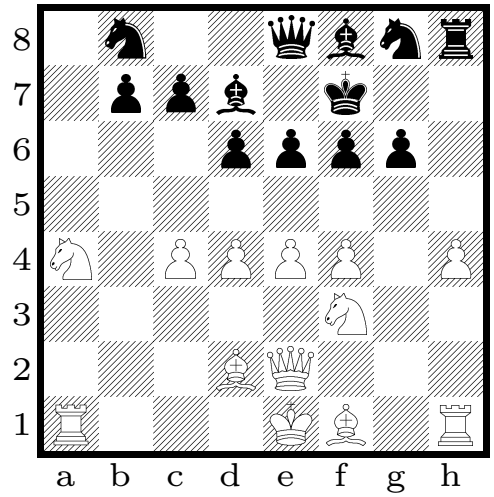


pawn capture in a5 14. ..R×a4

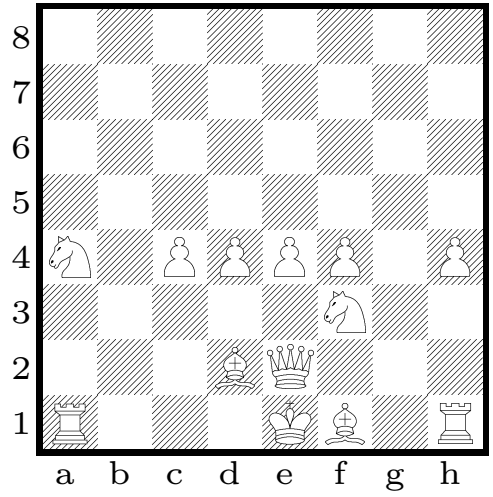


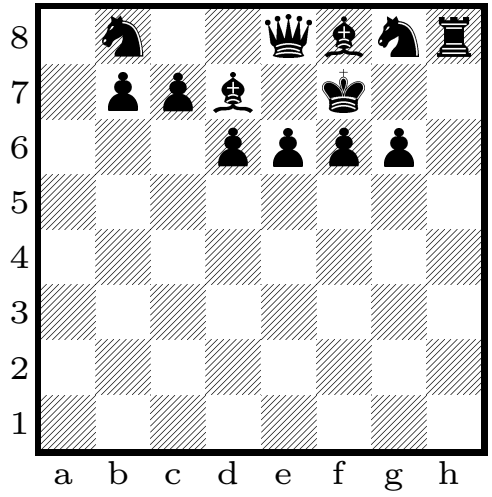


# PaoloC-Krieg, ICC Aprile 2003

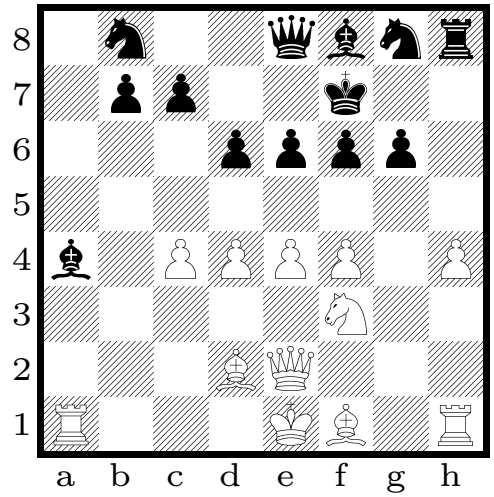


15.N×a4 piece capture in a4

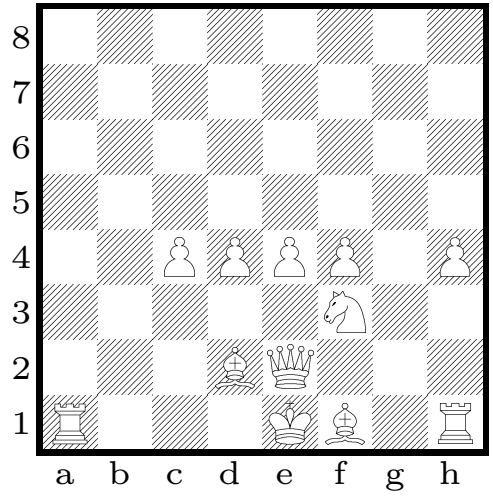


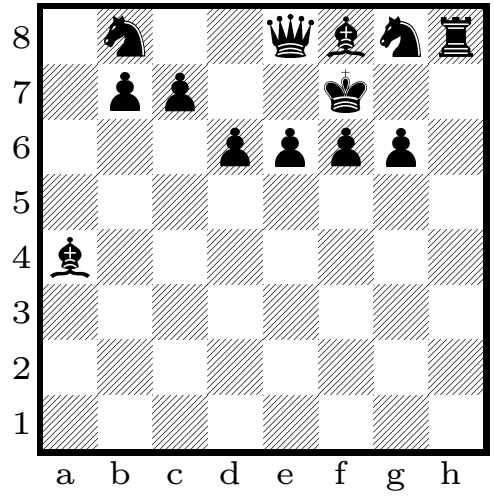


# PaoloC-Krieg, ICC Aprile 2003

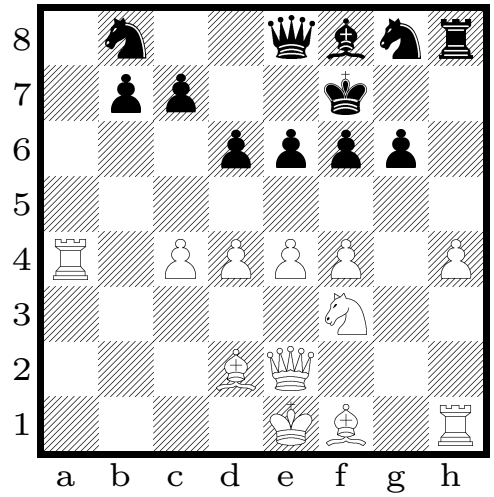


piece capture in a4 15. ..B×a4

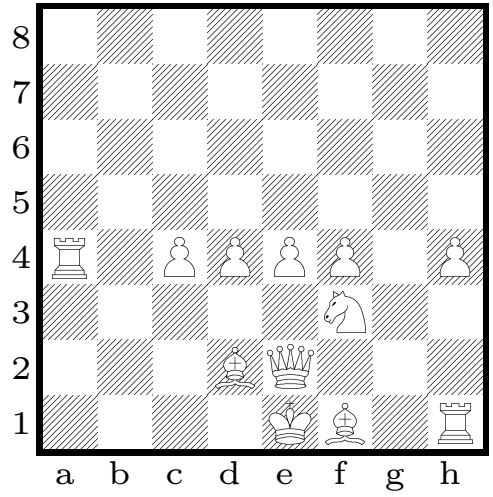


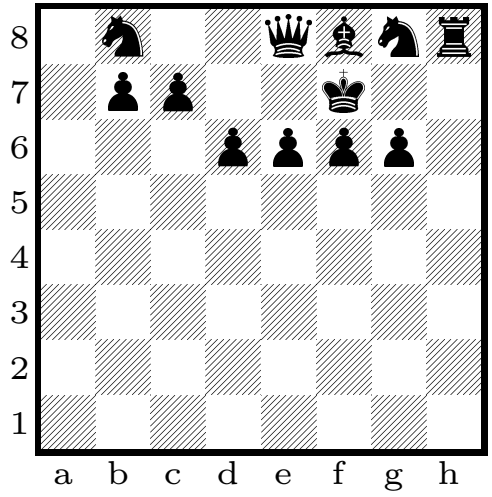


# PaoloC-Krieg, ICC Aprile 2003

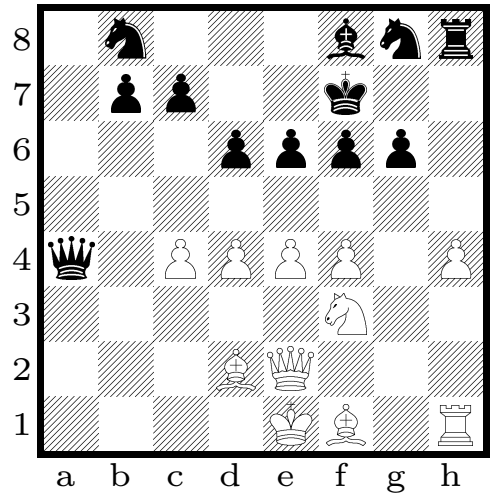


16.R×a4 piece capture in a4

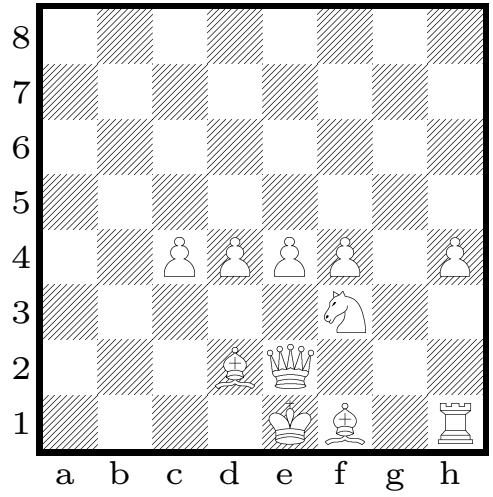


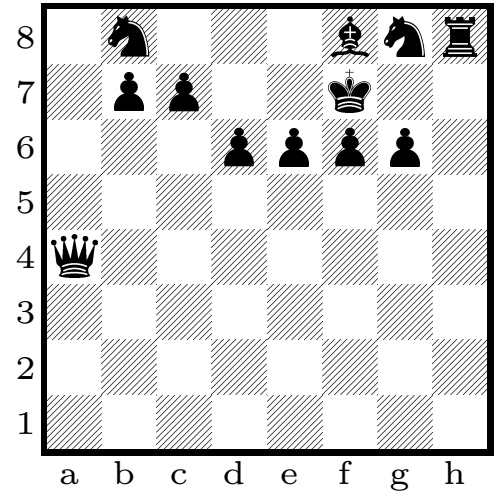


# PaoloC-Krieg, ICC Aprile 2003

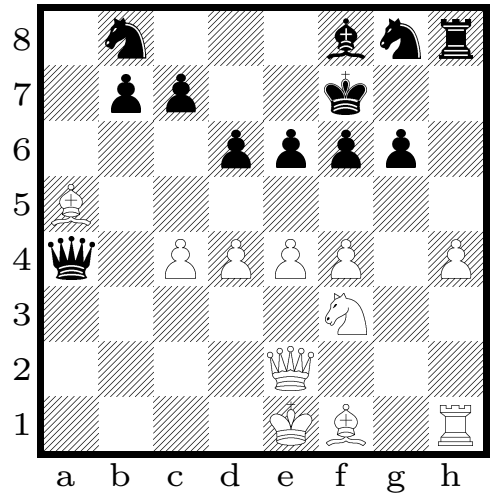


piece capture in a4 16. ..Q×a4

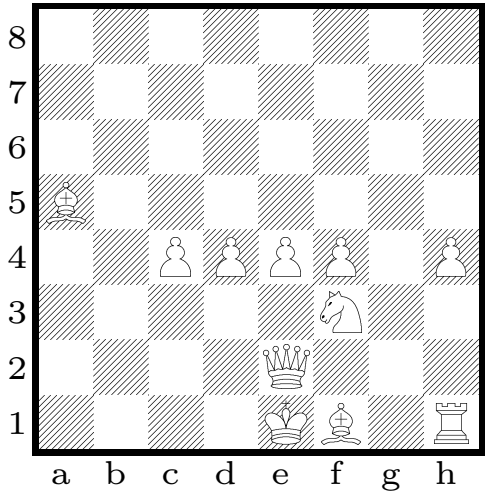


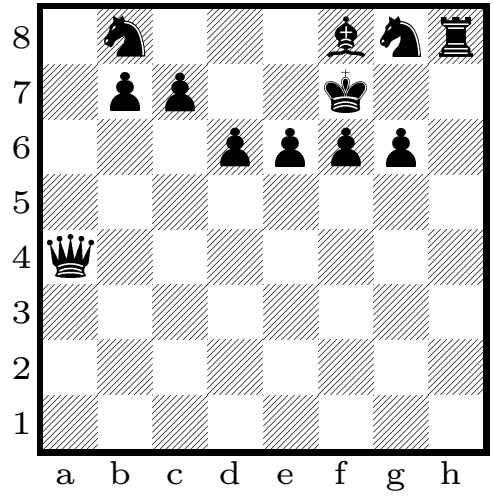


# PaoloC-Krieg, ICC Aprile 2003

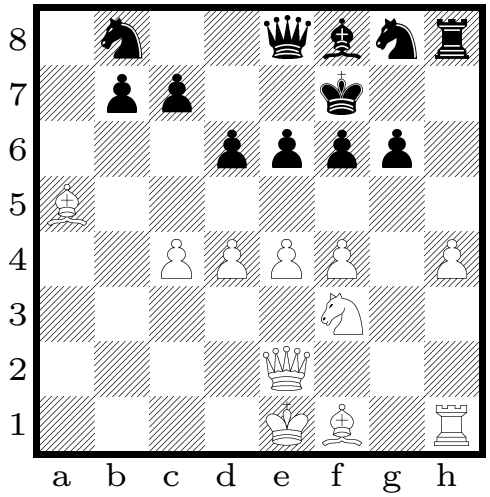


17.Ba5

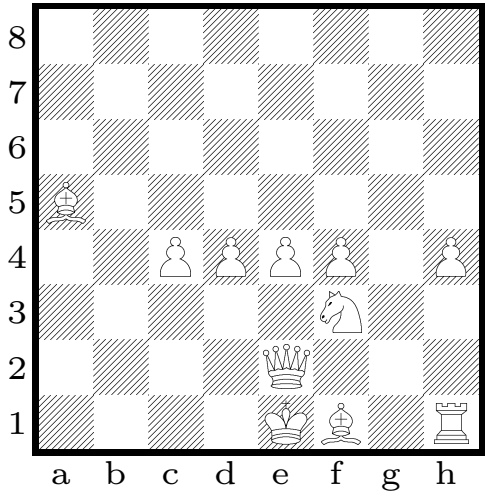


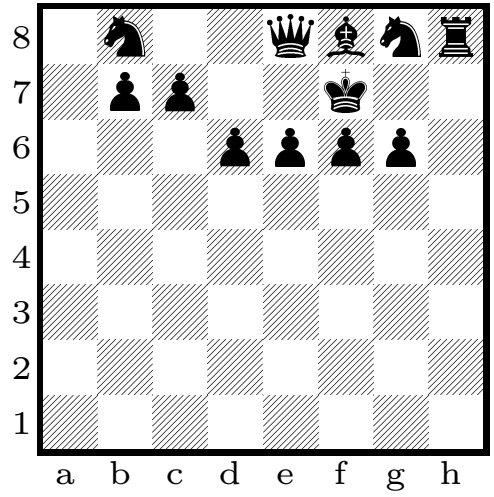


# PaoloC-Krieg, ICC Aprile 2003

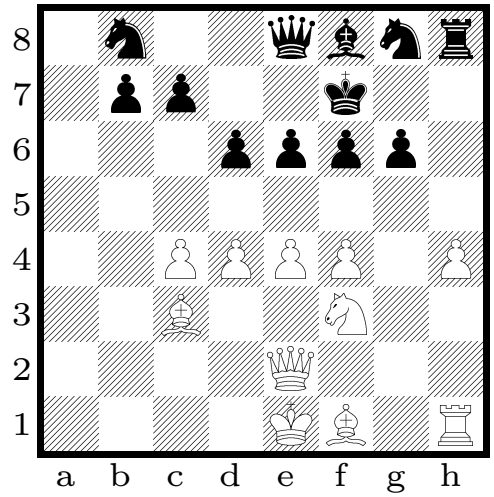


17. ..Qe8

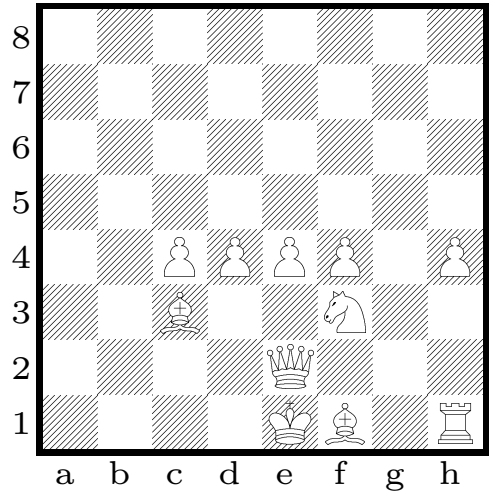


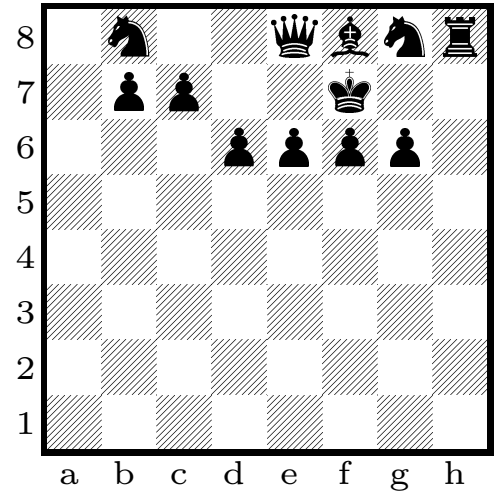


# PaoloC-Krieg, ICC Aprile 2003

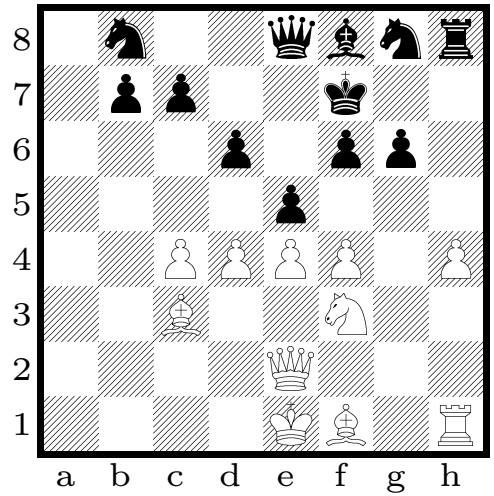


18.Bc3

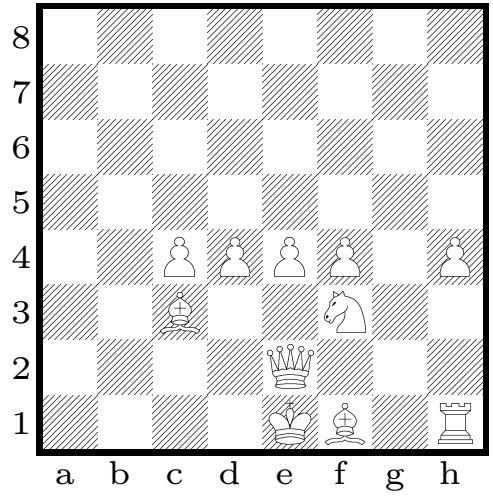


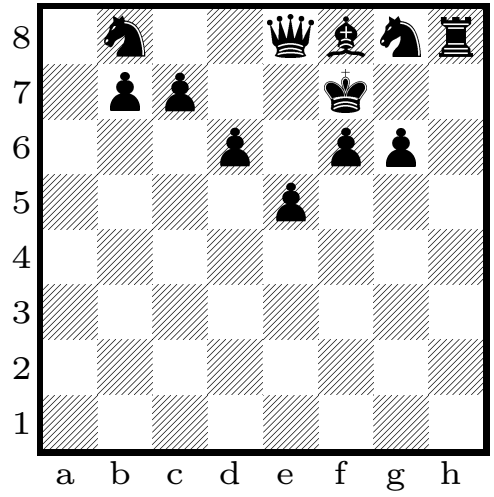


# PaoloC-Krieg, ICC Aprile 2003

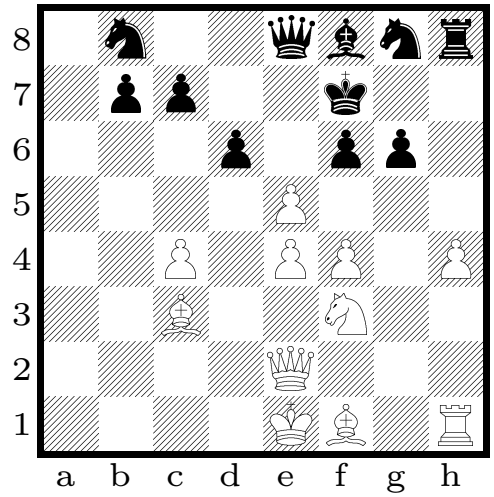


one pawn try 18. ..e5

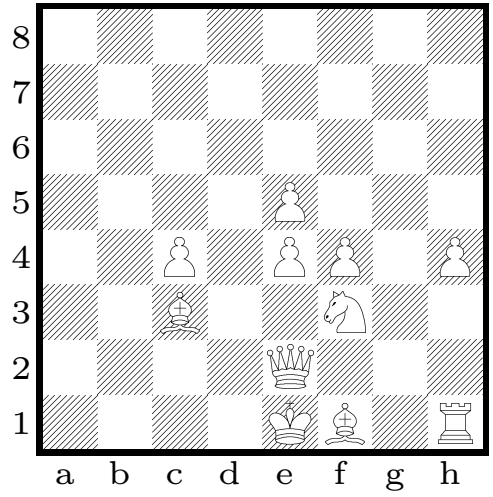


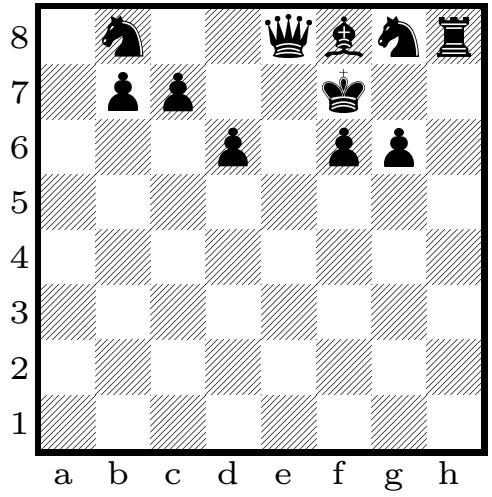


# PaoloC-Krieg, ICC Aprile 2003

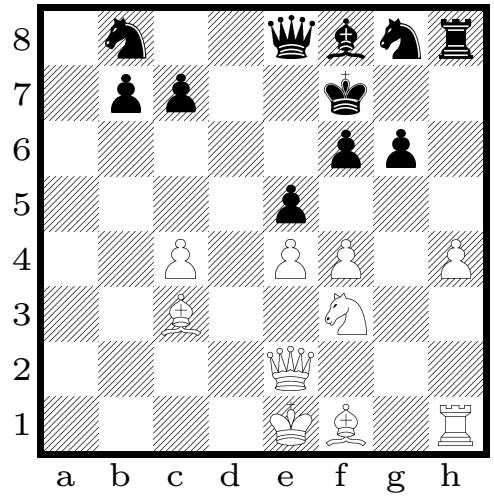


19.d×e5 pawn capture in e5

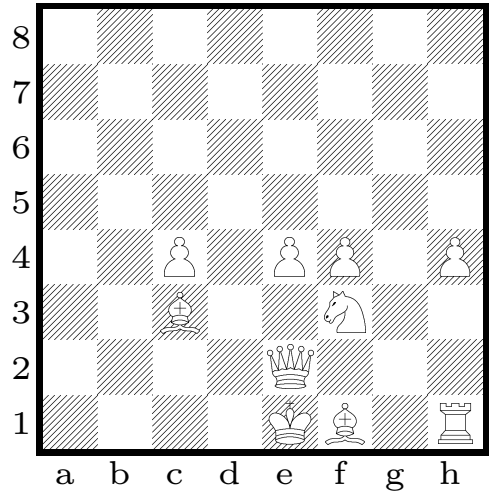


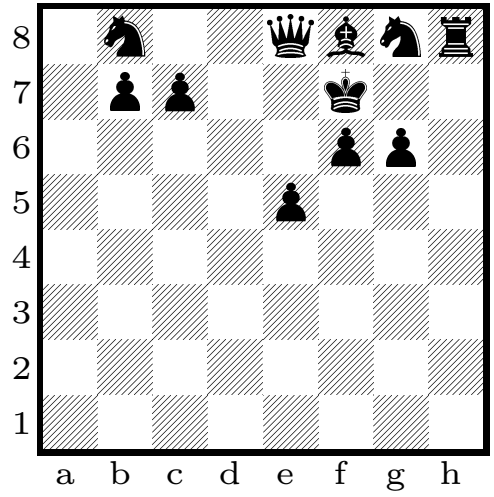


# PaoloC-Krieg, ICC Aprile 2003

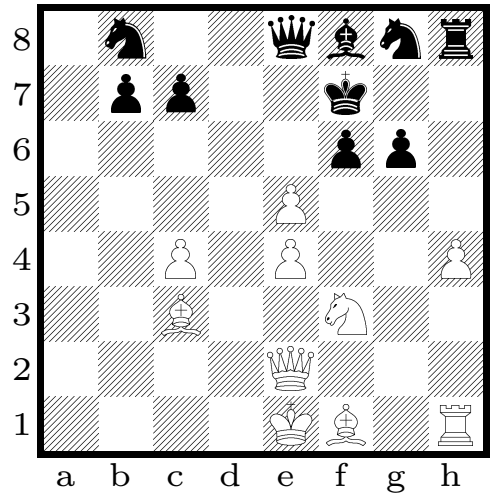


pawn capture in e5 19. ..d×e5

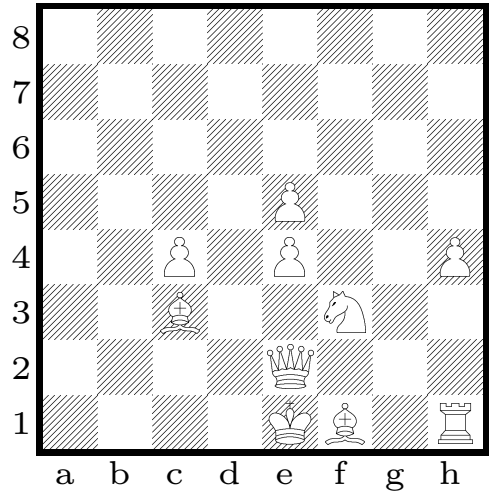


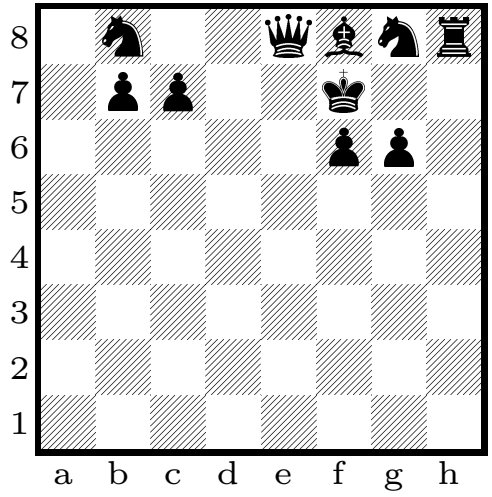


# PaoloC-Krieg, ICC Aprile 2003

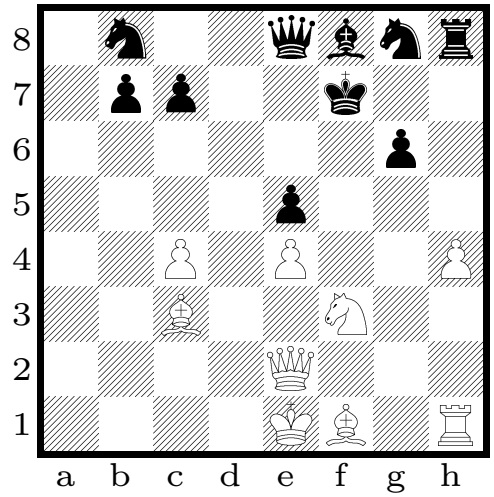


20.f×e5 pawn capture in e5

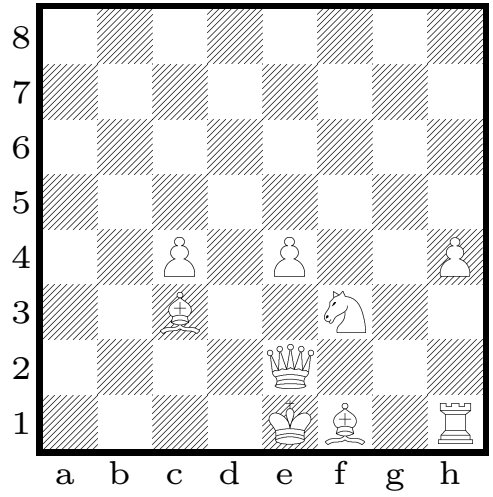


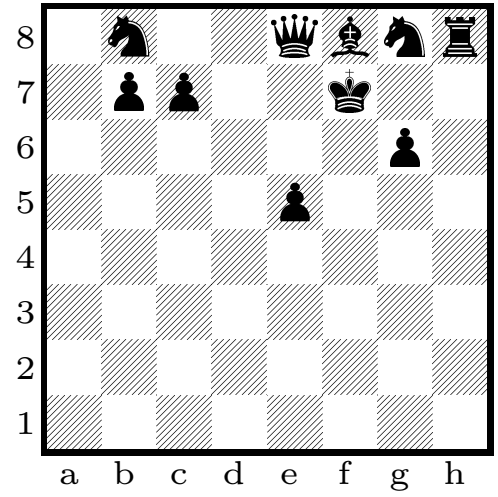


# PaoloC-Krieg, ICC Aprile 2003

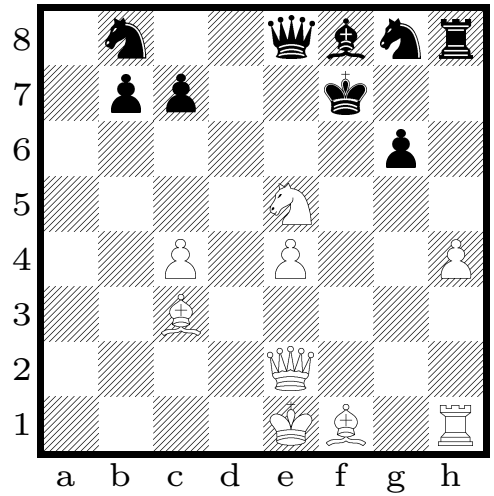


pawn capture in e5 20. ..f×e5

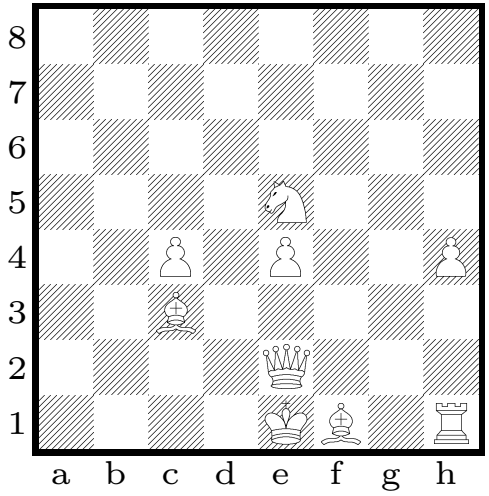


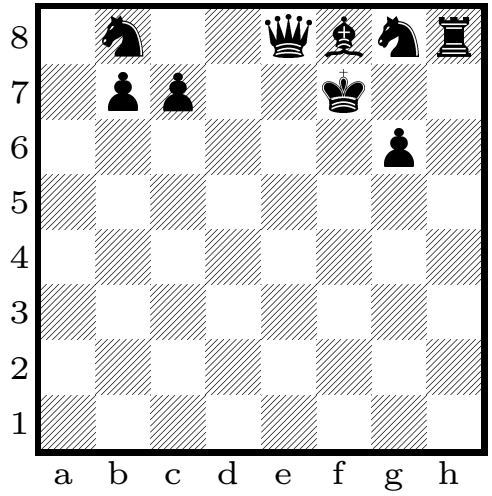


# PaoloC-Krieg, ICC Aprile 2003

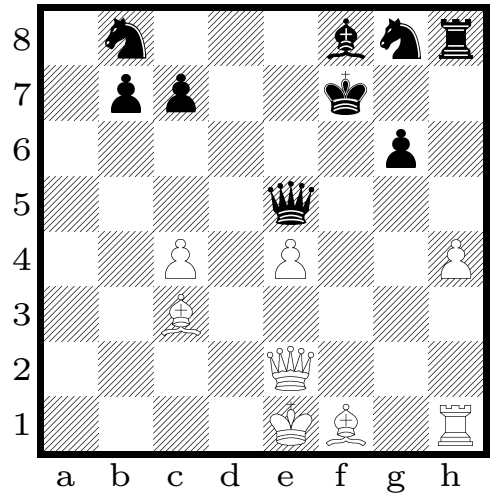


21.N×e5 check by knight; pawn capture in e5

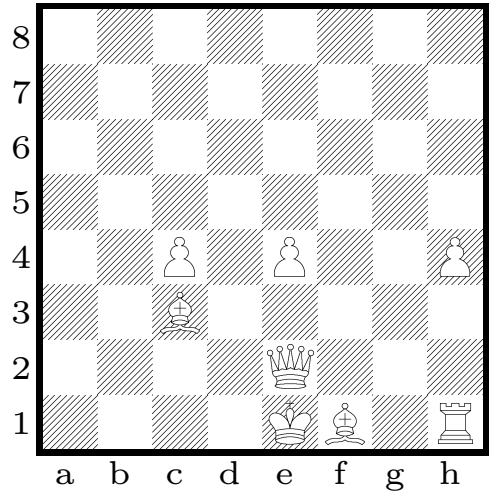


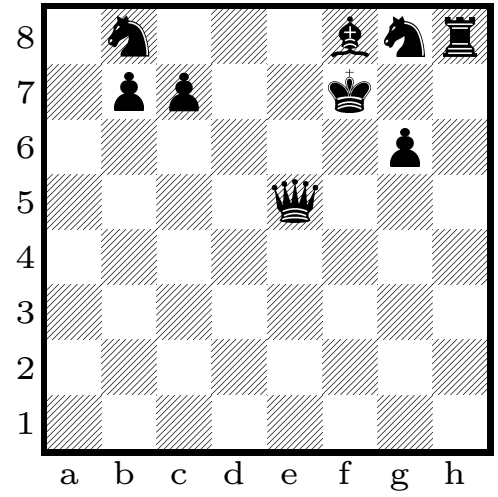


# PaoloC-Krieg, ICC Aprile 2003

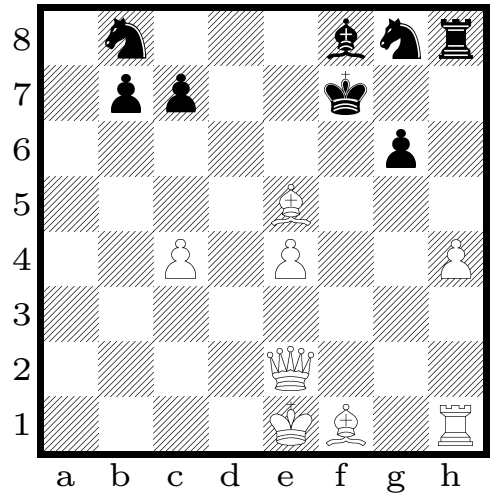


piece capture in e5 21. ..Q×e5

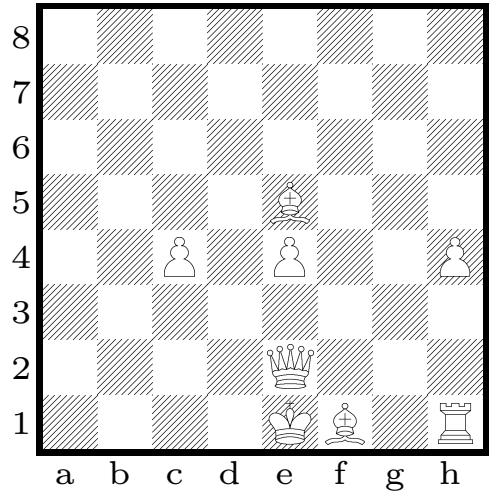


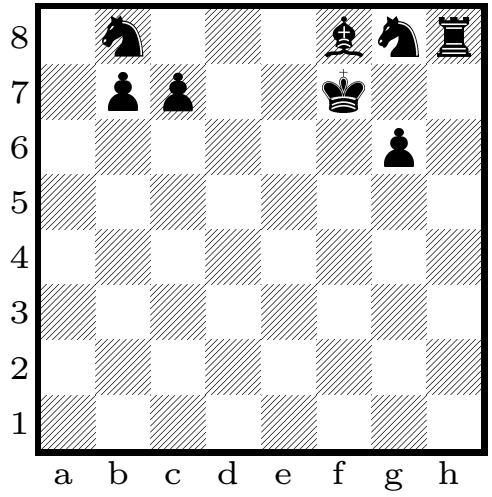


# PaoloC-Krieg, ICC Aprile 2003

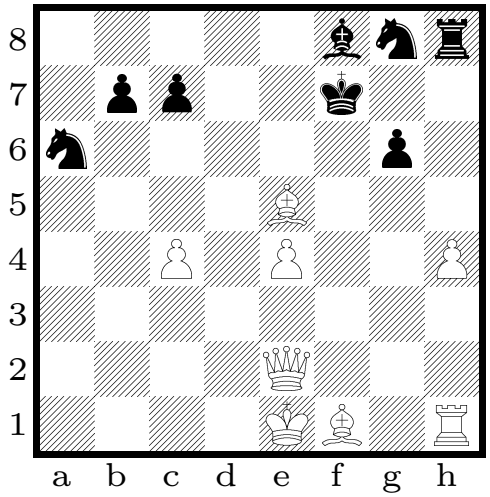


22.B×e5 piece capture in e5

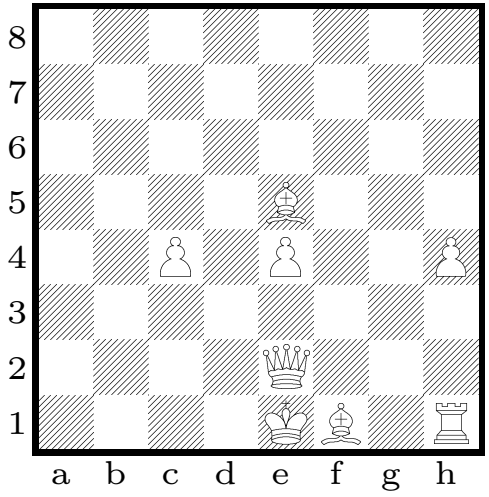


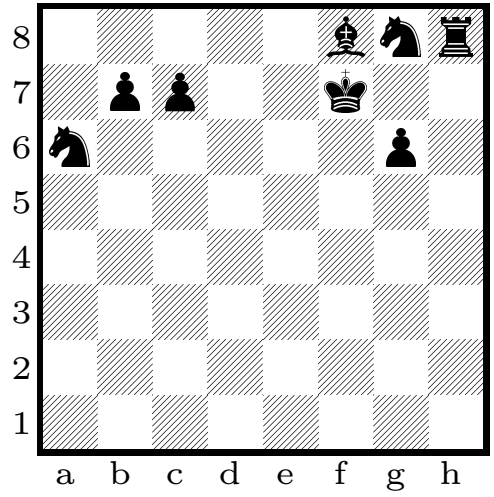


# PaoloC-Krieg, ICC Aprile 2003

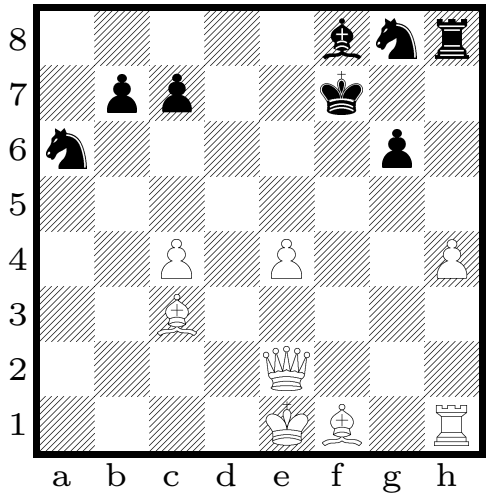


22. ..Na6

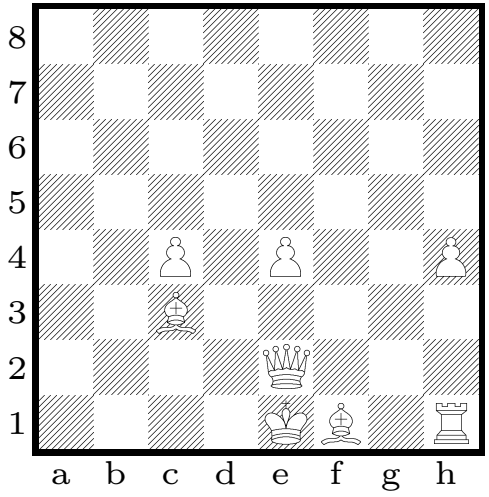


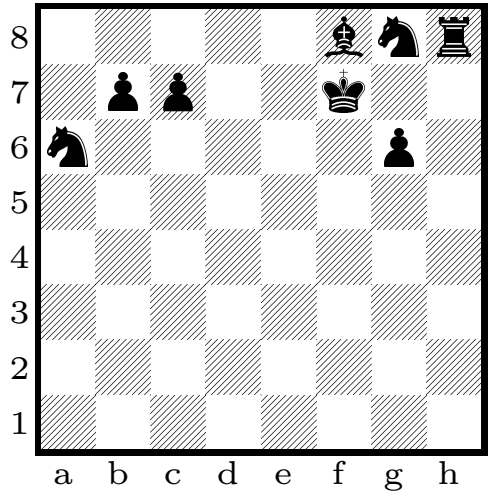


# PaoloC-Krieg, ICC Aprile 2003

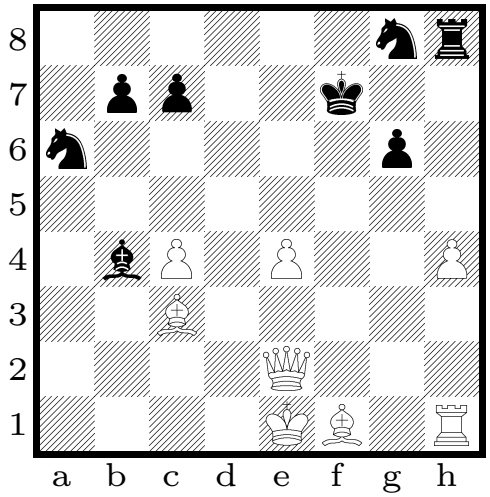


23.Bc3

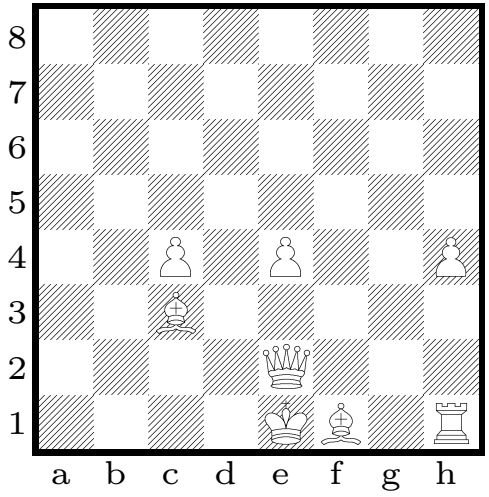


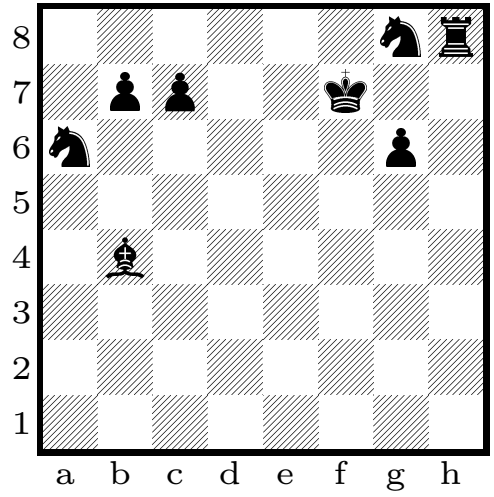


# PaoloC-Krieg, ICC Aprile 2003

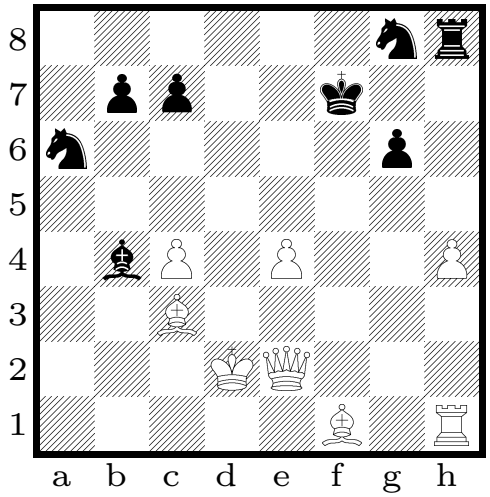


23. ..Bb4

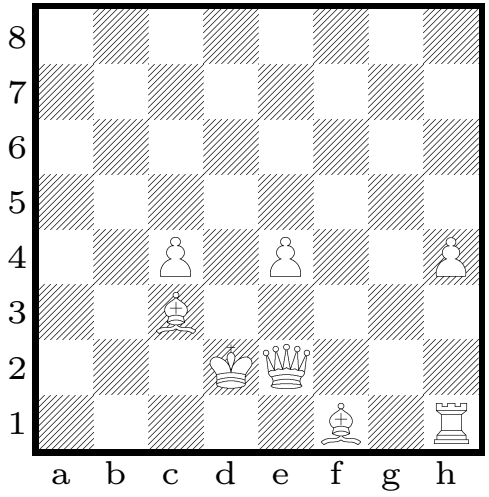


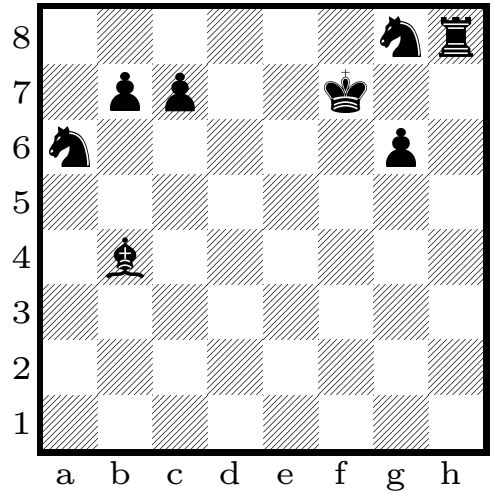


# PaoloC-Krieg, ICC Aprile 2003

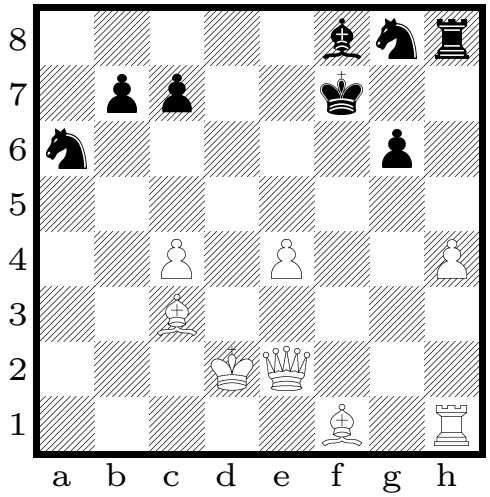


24.Kd2

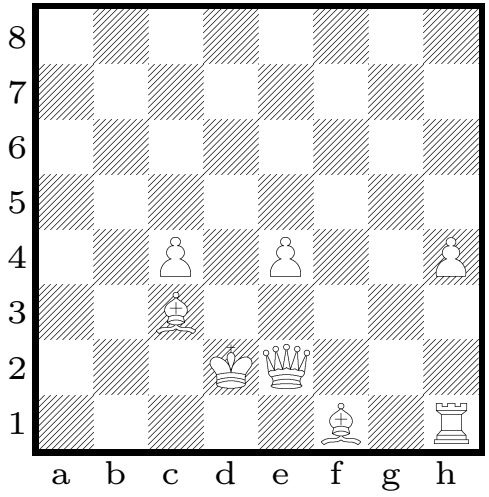


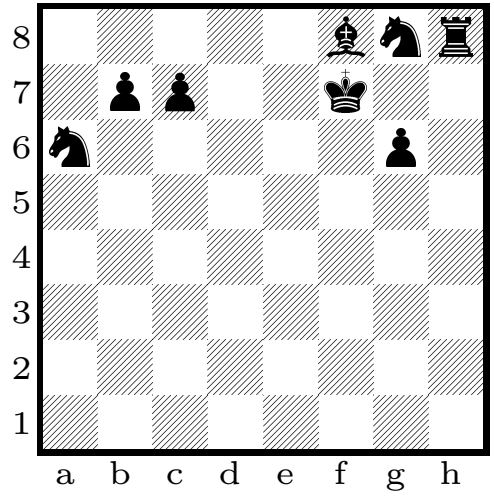


# PaoloC-Krieg, ICC Aprile 2003

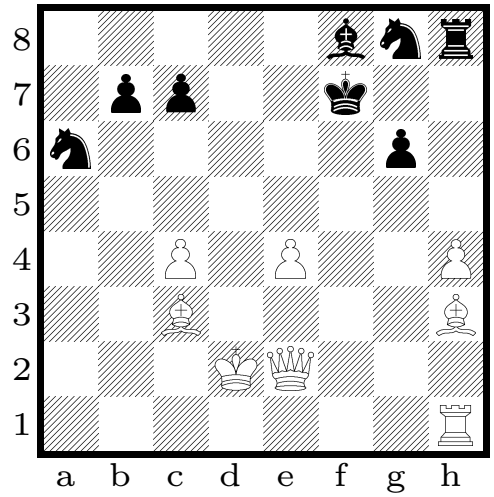


24. ..Bf8

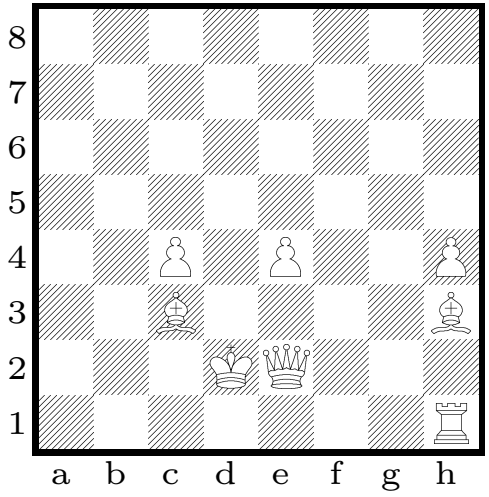


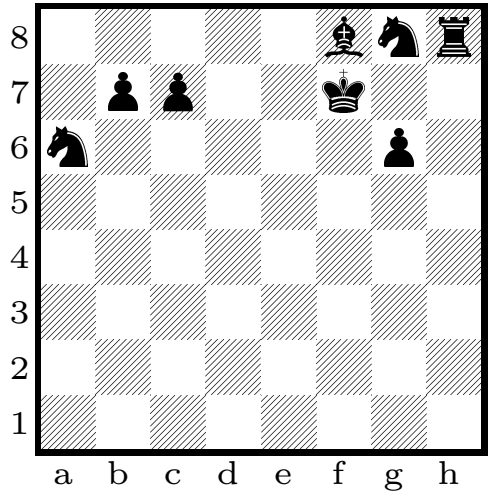


# PaoloC-Krieg, ICC Aprile 2003

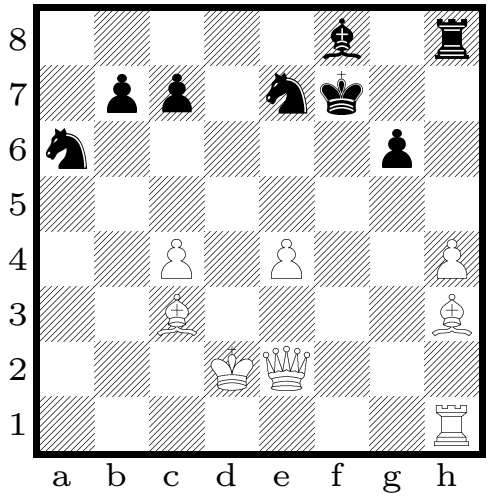


25.Bh3

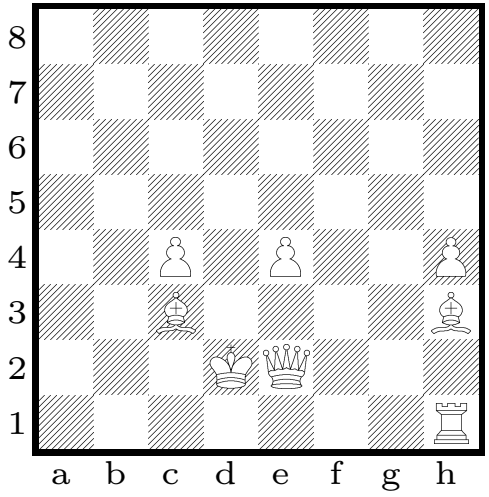


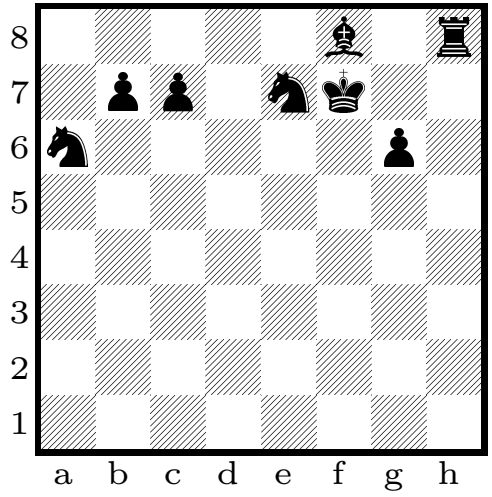


# PaoloC-Krieg, ICC Aprile 2003

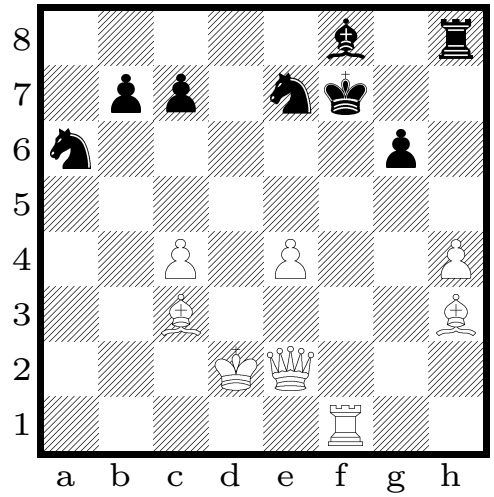


25. ..Ne7

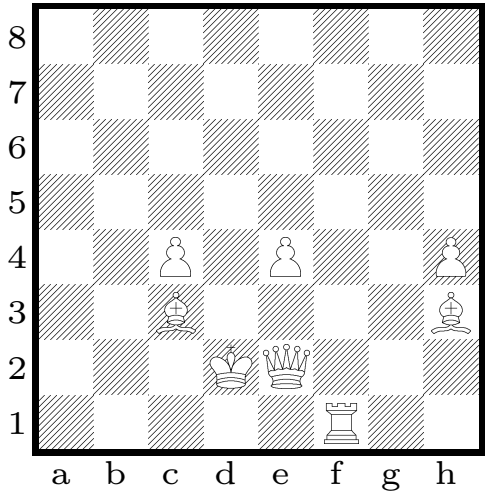


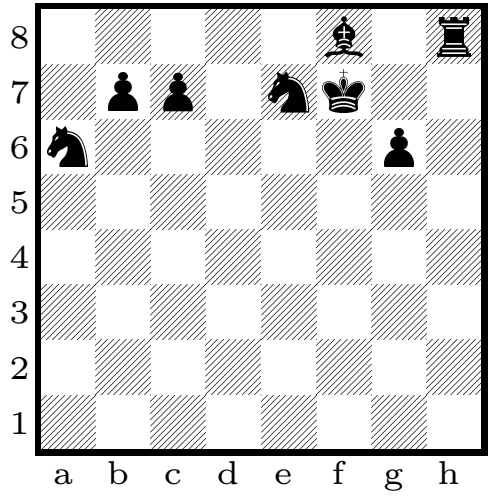


# PaoloC-Krieg, ICC Aprile 2003

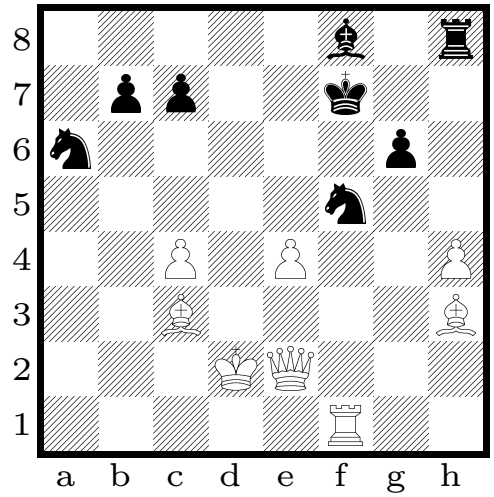


26.Rf1 check on file

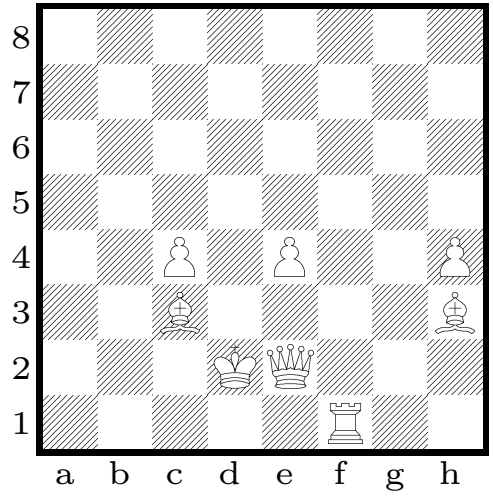


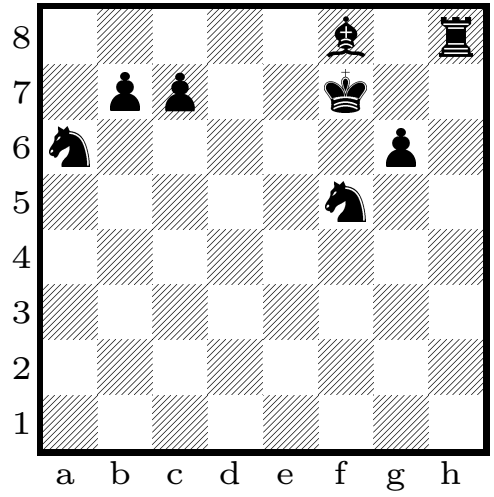


# PaoloC-Krieg, ICC Aprile 2003

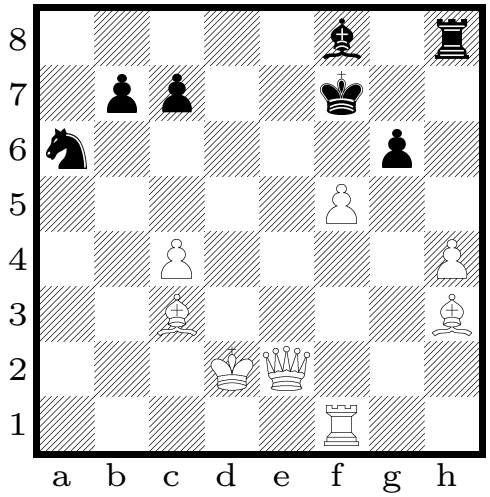


one pawn try 26. ..Nf5

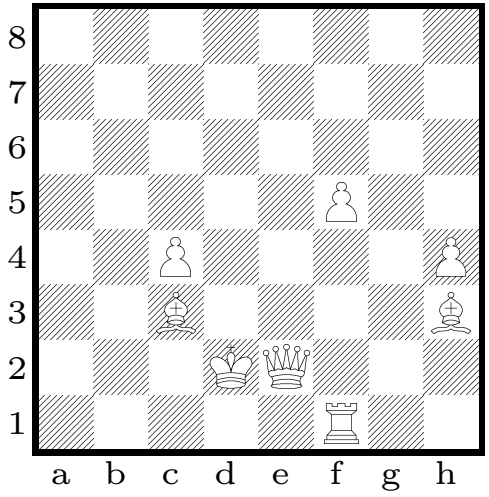


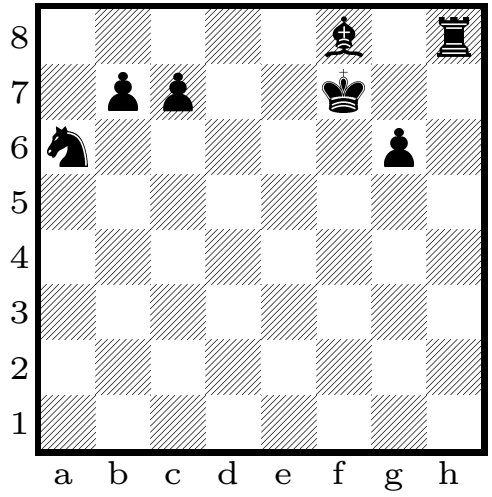


# PaoloC-Krieg, ICC Aprile 2003

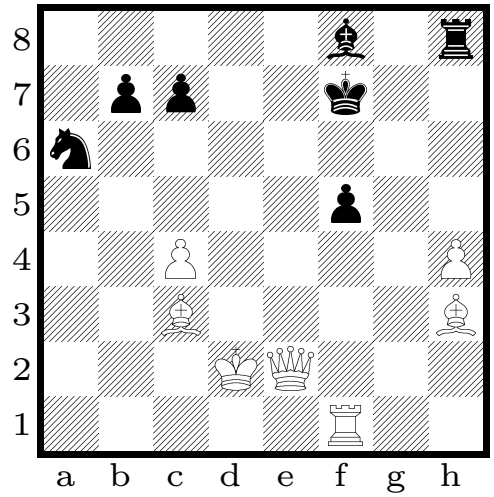


27.e×f5 piece capture in f5

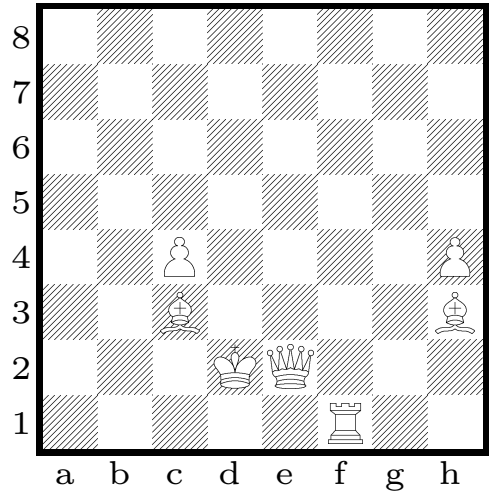


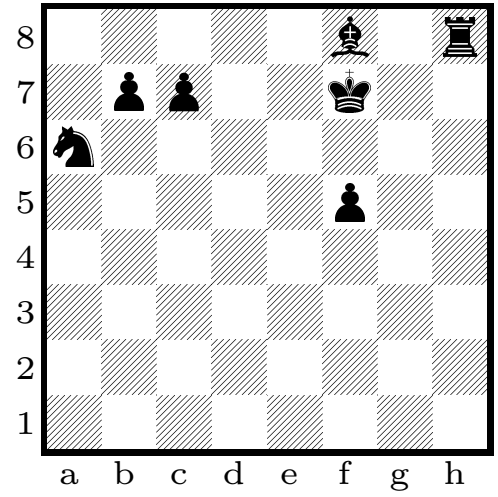


# PaoloC-Krieg, ICC Aprile 2003

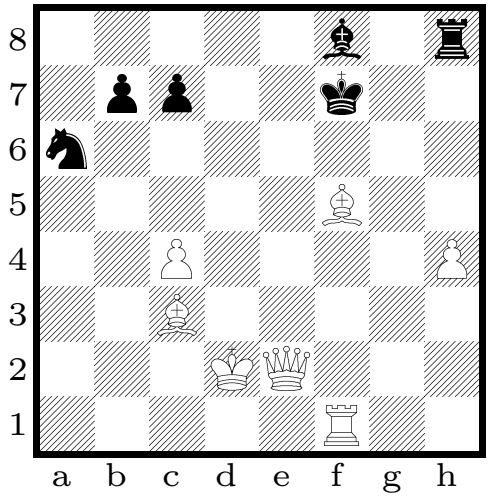


pawn capture in f5 27. ..g×f5

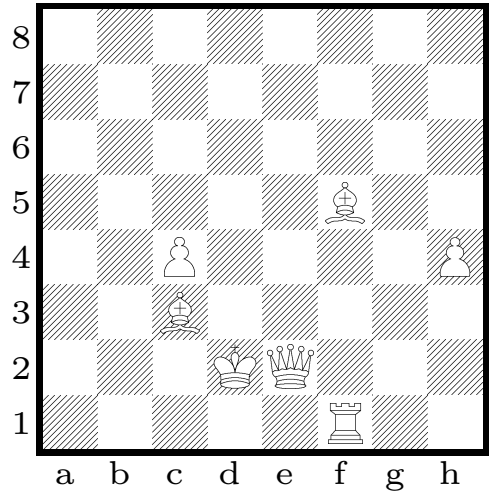


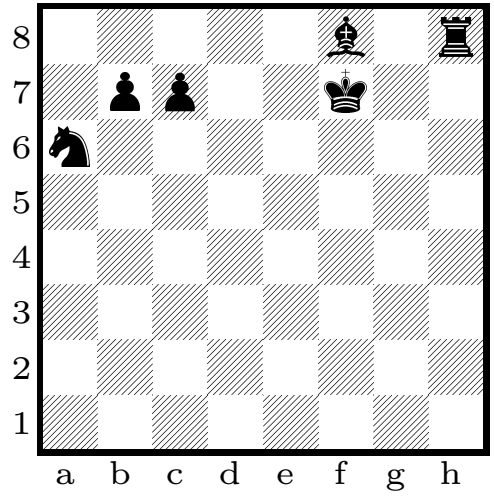


# PaoloC-Krieg, ICC Aprile 2003

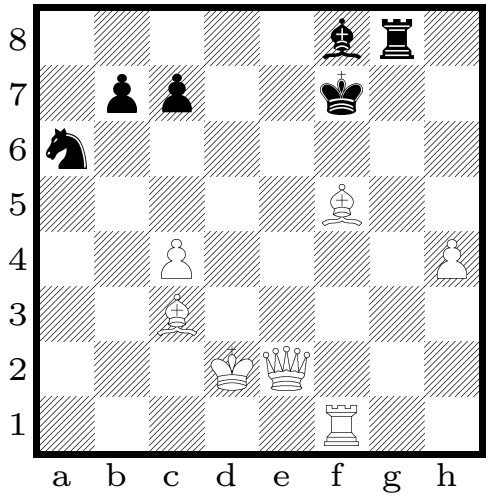


28.B×f5 pawn capture in f5

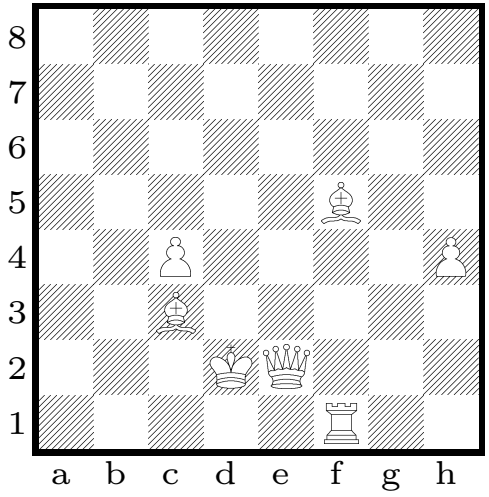


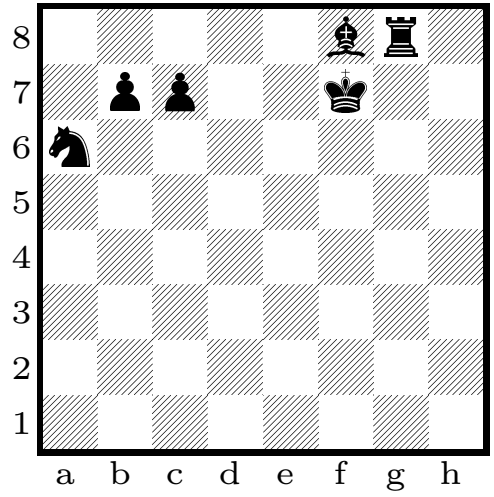


# PaoloC-Krieg, ICC Aprile 2003

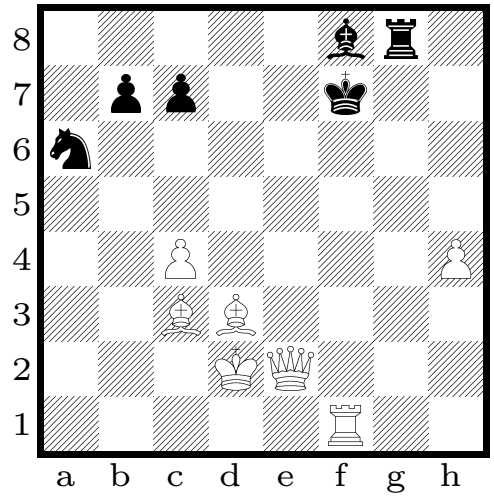


28. ..Rg8

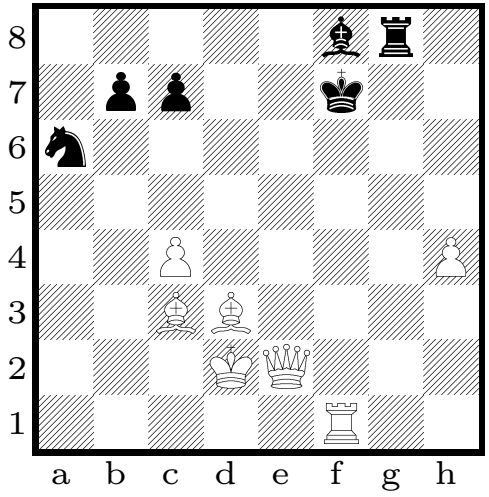


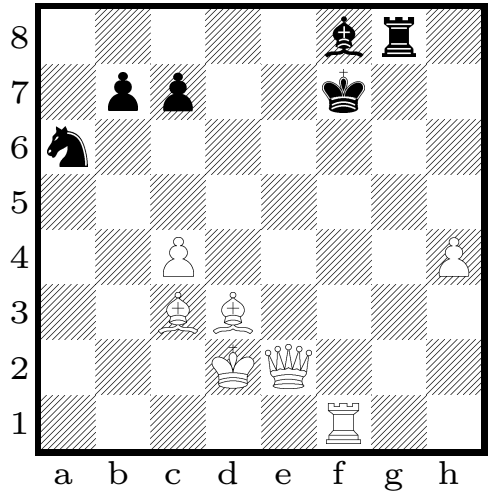


# PaoloC-Krieg, ICC Aprile 2003

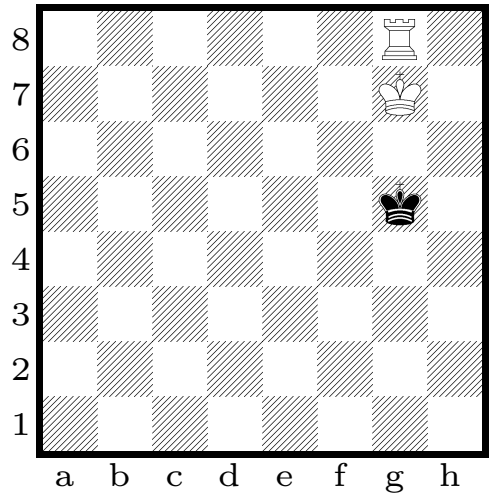


29.Bd3 # 1-0 checkmate 9. ..Qe7





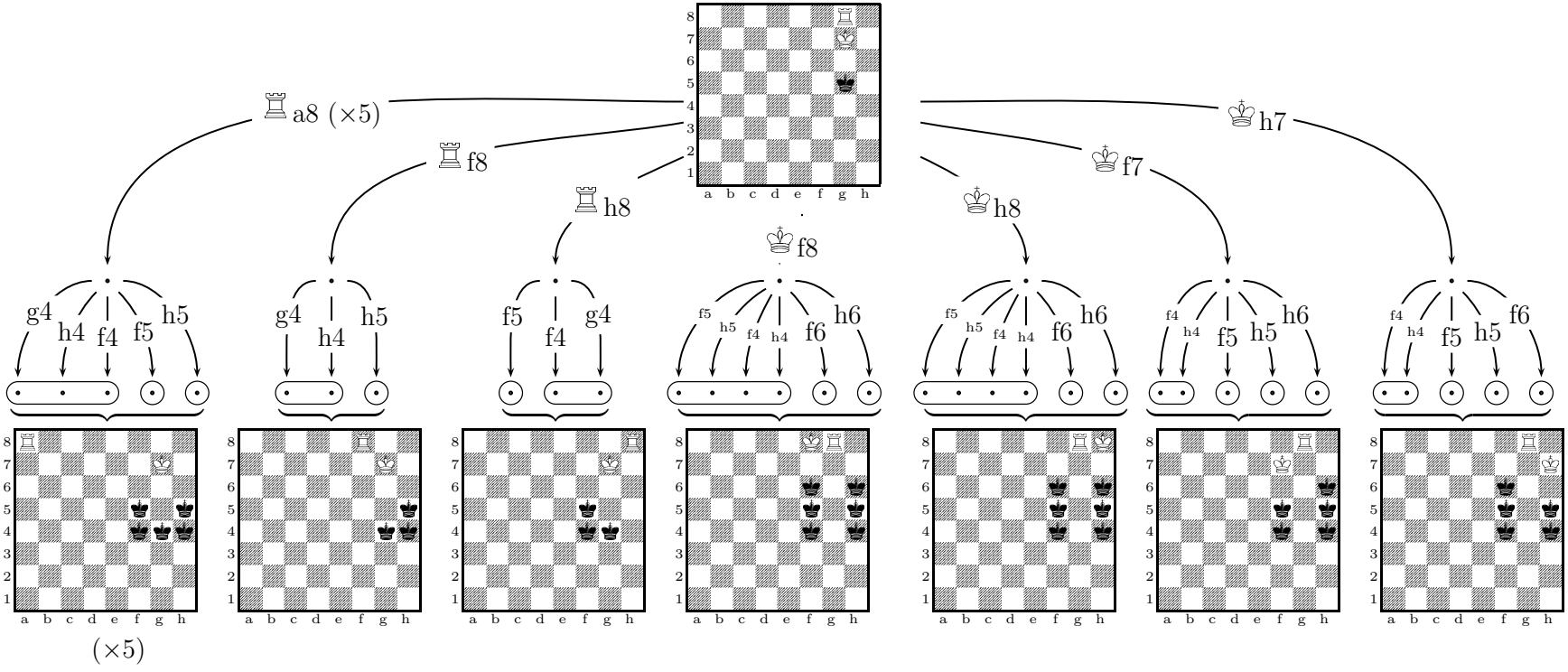
# Example of position (1/3)




- This diagram shows an example of position during a game of ♔♖♚ ending;
- the game tree has a branch of 11 moves, corresponding to the possible White's moves,
- plus the Black's ones.

# Example of position (2/3)

- Black's choices compose the *information sets* for White, who does not know where Black has moved his pieces;
- In the example seen before we have to deal with 33 *information sets* for White and 53 positions in total



# Example of position (3/3)

Considering White moves ♖f8:

- Black's possible moves are ♜h5, ♜h4 or ♜g4;
- White now faces two information sets of 2 and 1 elements respectively:
  - The former includes 21 possible moves,
  - the latter includes 19 moves
- the game tree branches out with

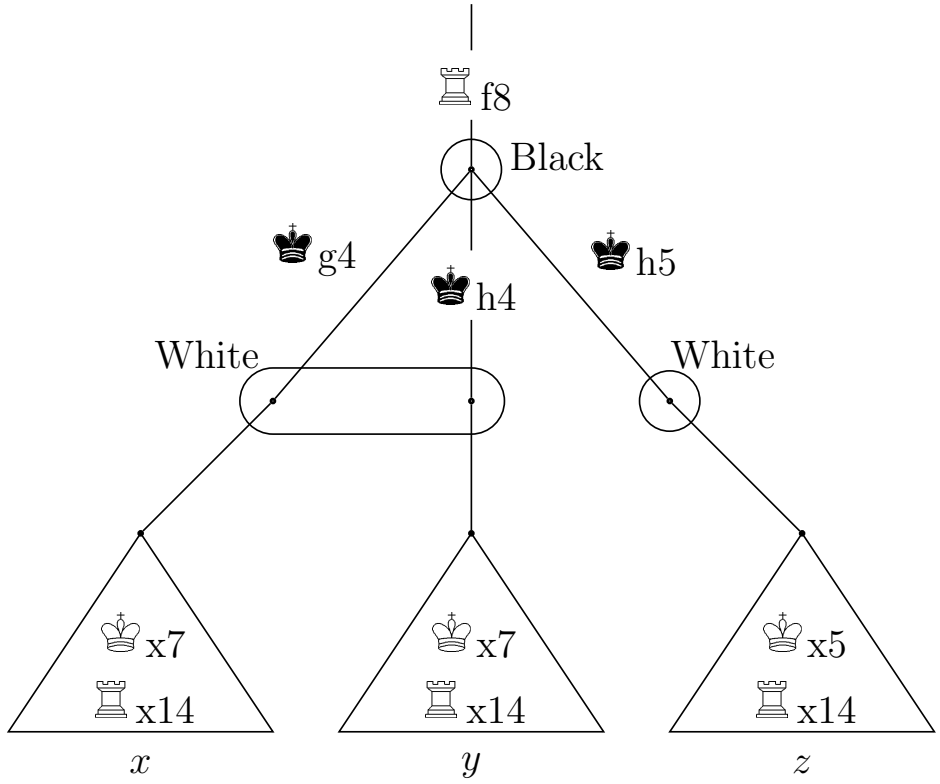
$$2 \cdot 21 + 19 = 61$$

  further nodes

We aim to obtain a tree without information sets with more than an element.

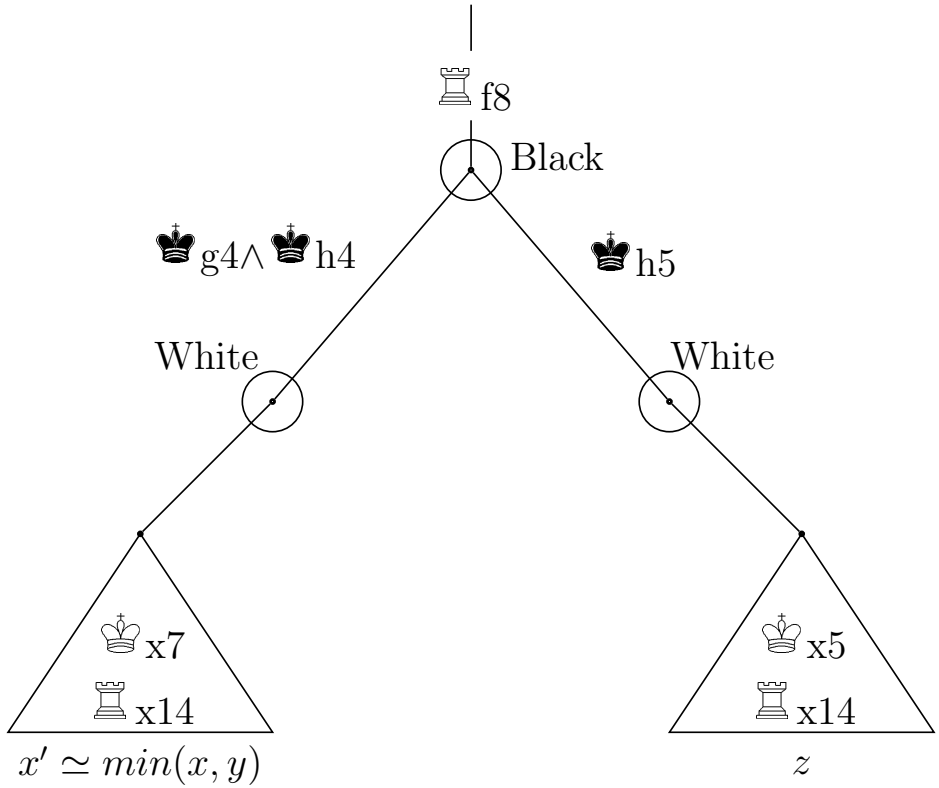
# Using metapositions

We could join the moves made by Black which lead to the same information set:



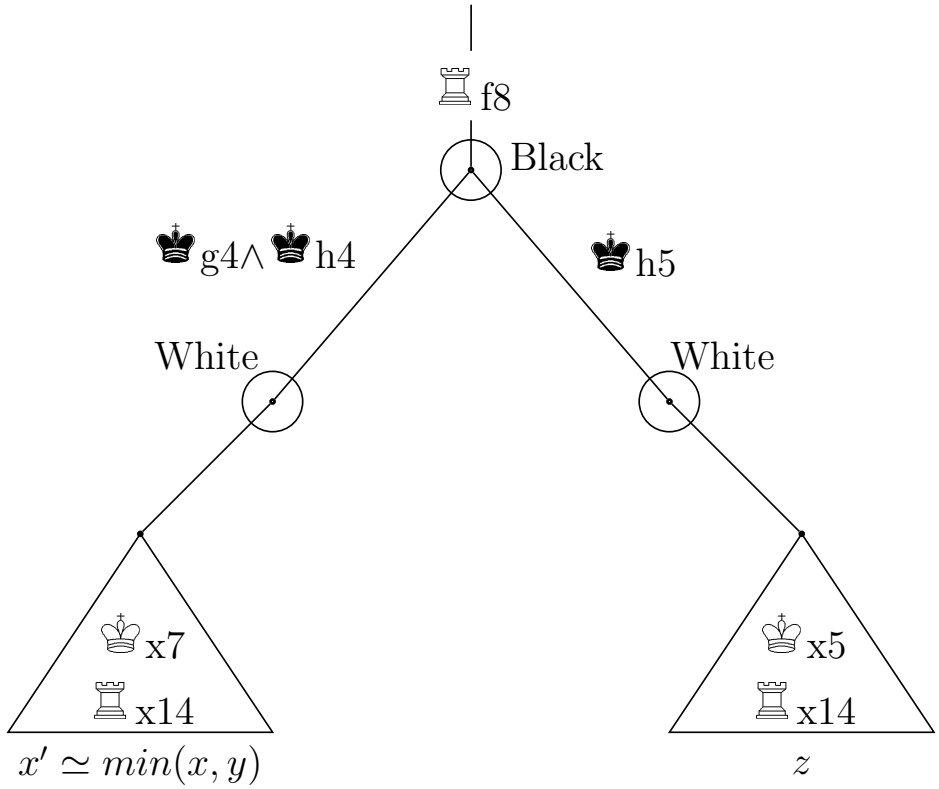
# Using metapositions

We could join the moves made by Black which lead to the same information set:



# Using metapositions

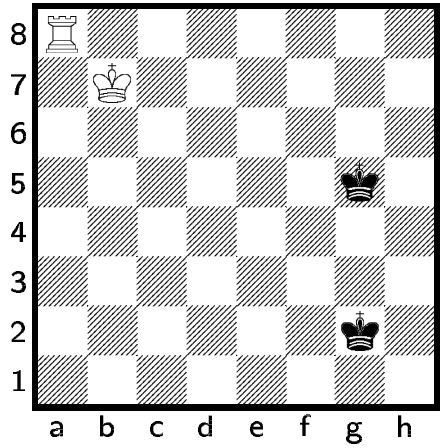
We could join the moves made by Black which lead to the same information set:




Because of the complexity of a procedure which distinguishes between Black's moves that lead to different information sets for White we define the game tree in a simpler but equivalent way.

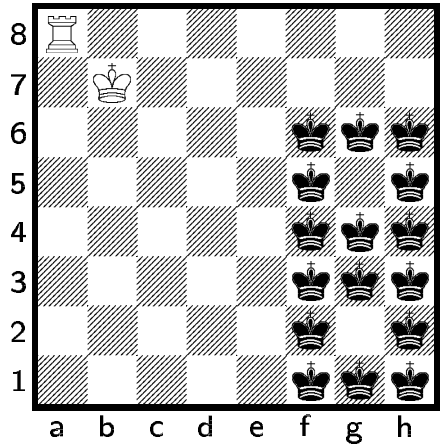
# Metapositions

- *Metaposition*: is a particular position which describes a set of positions.
- *Metamove*: a move by Black player from one metaposition to another.
- *Pseudomove*: a move by White player on a metaposition.



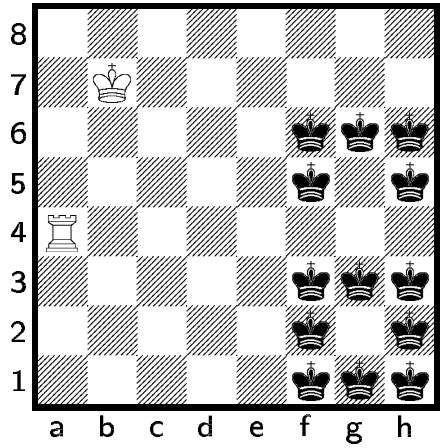
# Metapositions

- *Metaposition*: is a particular position which describes a set of positions.
- *Metamove*: a move by Black player from one metaposition to another.
- *Pseudomove*: a move by White player on a metaposition.



# Metapositions

- *Metaposition*: is a particular position which describes a set of positions.
- *Metamove*: a move by Black player from one metaposition to another.
- *Pseudomove*: a move by White player on a metaposition.



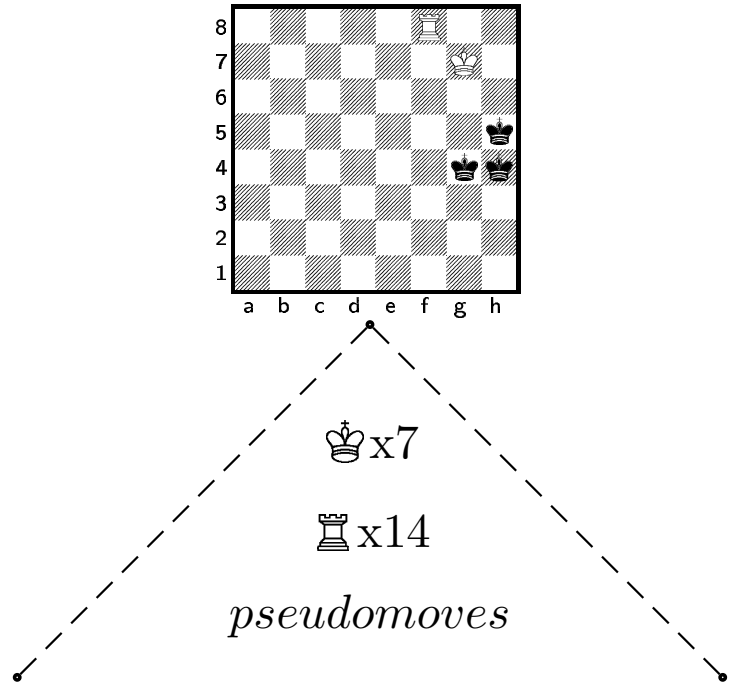
# Cardinality of pseudomoves

- The set $P$ of pseudomoves has a cardinality equal to the maximum cardinality of legal moves for each position.

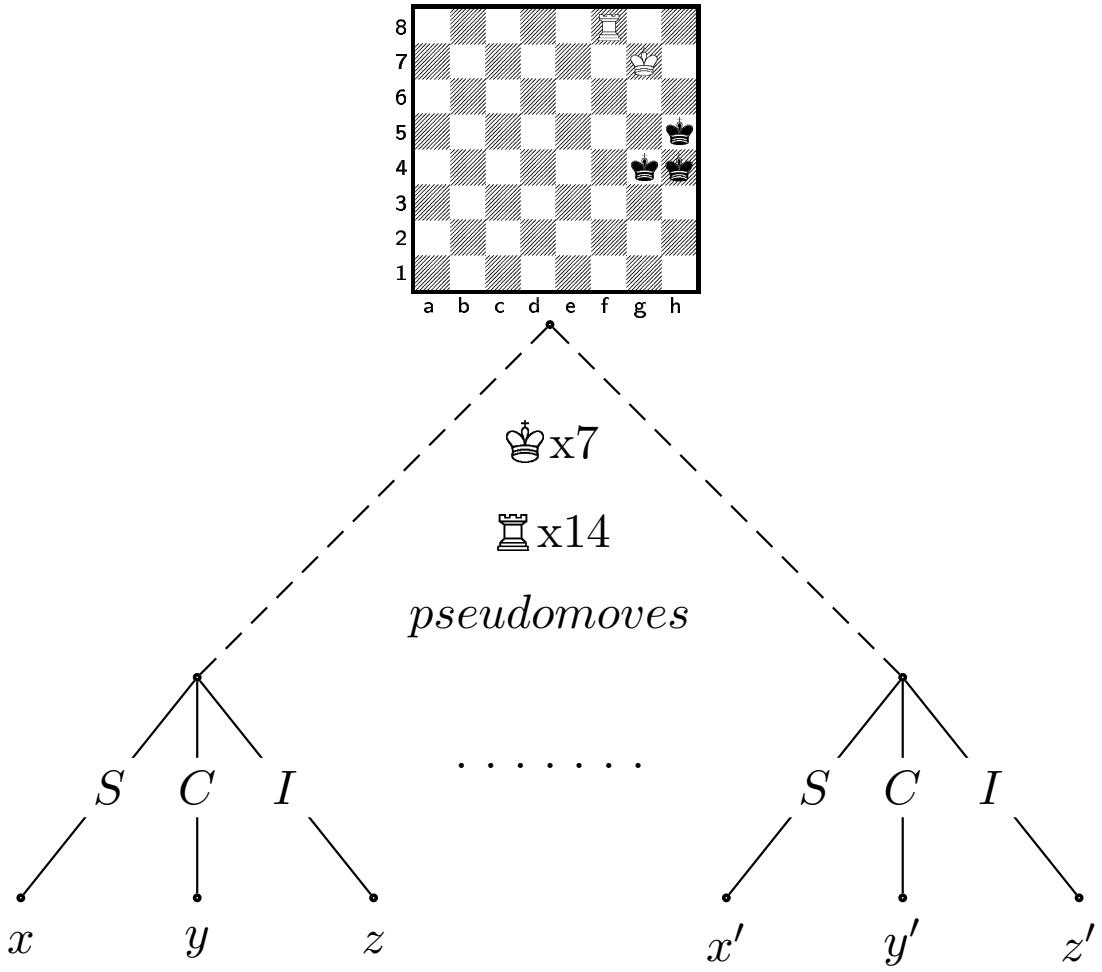
$$|P| = \max(|\text{legal moves}|)$$

- For the example seen before White always considers 21 pseudomoves, even if he is in the case with 19 legal moves, because he cannot distinguish between the two situations.

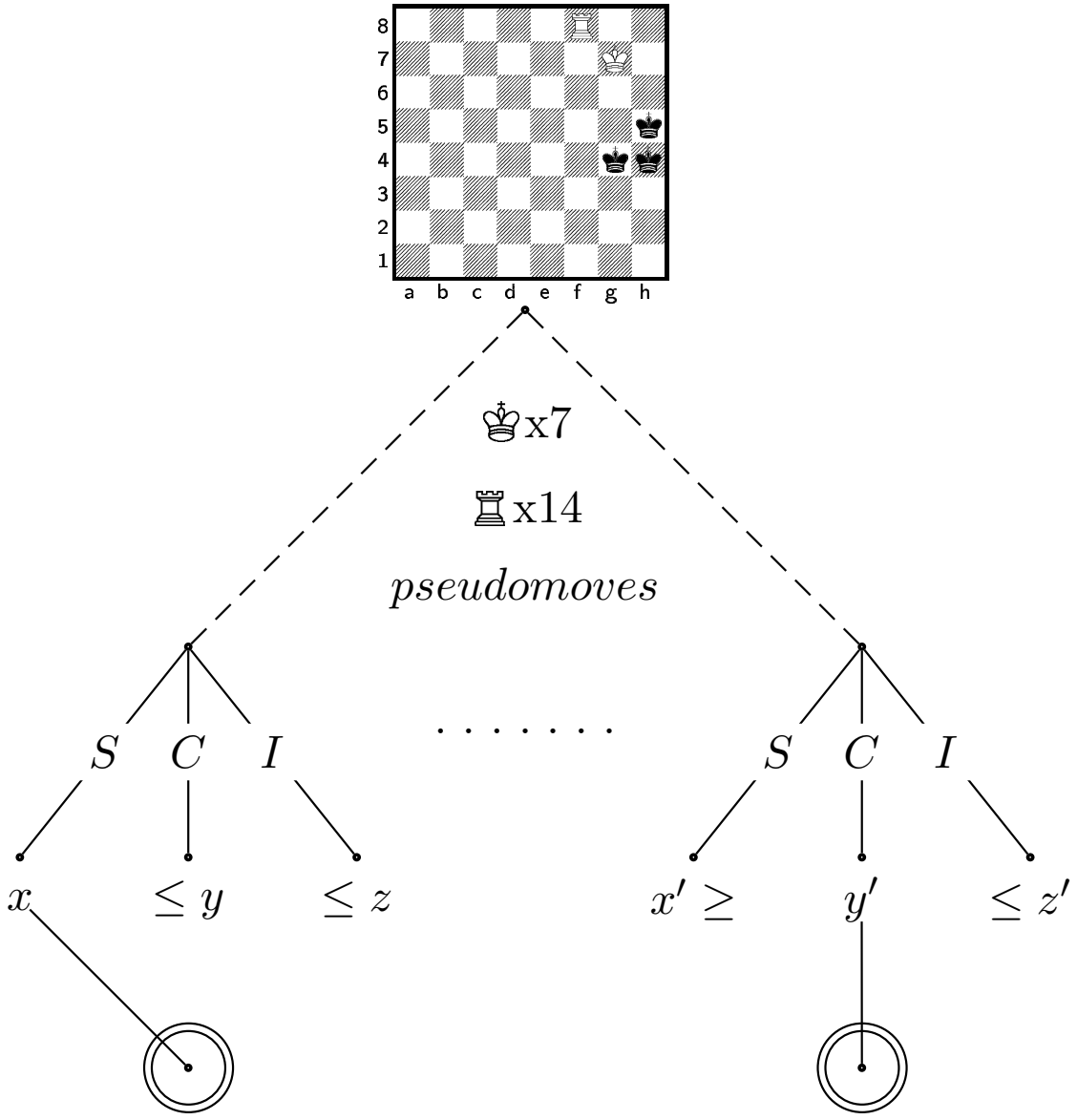
# The game tree



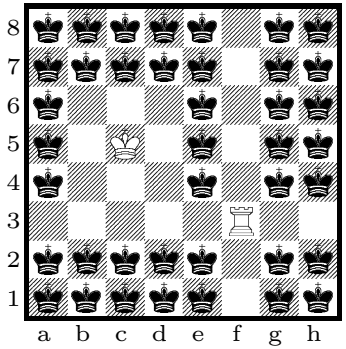
# The game tree



# The game tree

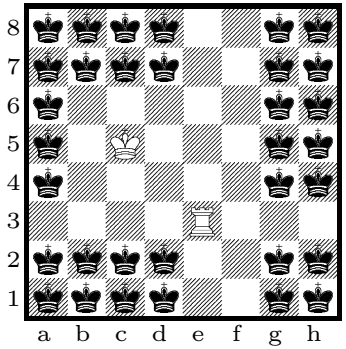


# Exploiting the referee's answers



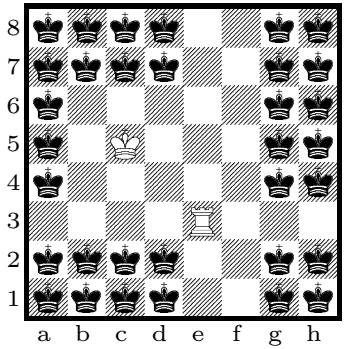

♖ **e3 silent:** we clean the squares around the ♔ and along ♖'s row and column

# Exploiting the referee's answers

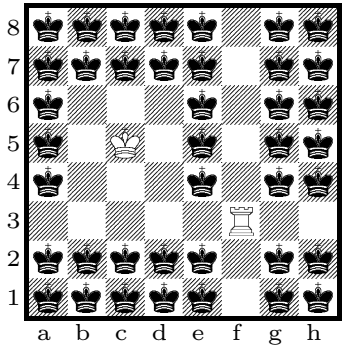



♖**e3 silent:** we clean the squares around the ♔ and along ♖'s row and column

# Exploiting the referee's answers

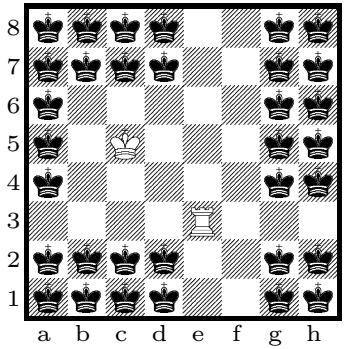


♖ **e3 silent:** we clean the squares around the ♔ and along ♖'s row and column

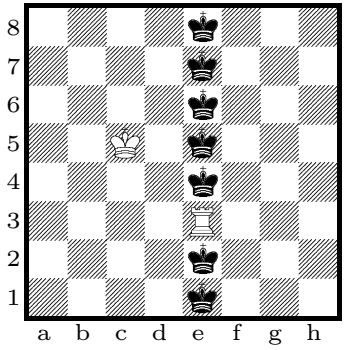


♖ **e3 check:** we clean all the squares and we assume that the ♛ is on the ♖'s row or column

# Exploiting the referee's answers

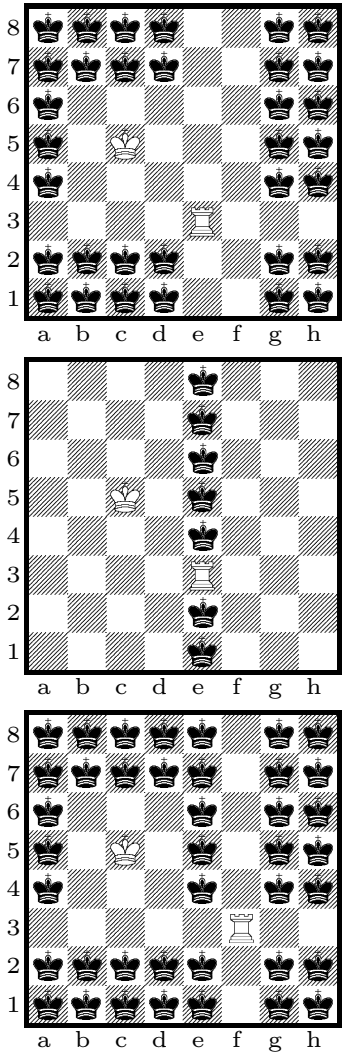


♖ **e3 silent:** we clean the squares around the ♔ and along ♖'s row and column



♖ **e3 check:** we clean all the squares and we assume that the ♛ is on the ♖'s row or column

# Exploiting the referee's answers

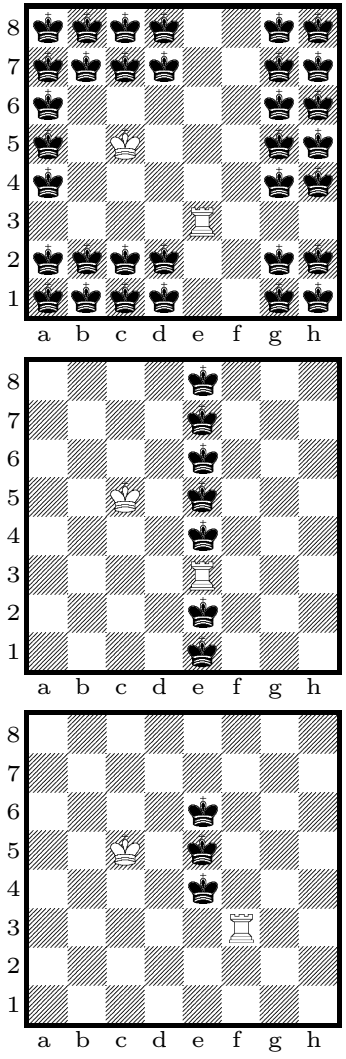


♖ **e3 silent:** we clean the squares around the ♔ and along ♖'s row and column

♖ **e3 check:** we clean all the squares and we assume that the ♚ is on the ♖'s row or column

♔ **d5 illegal:** we clean all the squares leaving those around the ♔

# Exploiting the referee's answers



♖ **e3 silent:** we clean the squares around the ♔ and along ♖'s row and column

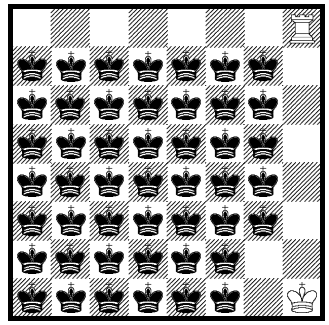
♖ **e3 check:** we clean all the squares and we assume that the ♛ is on the ♖'s row or column

♔ **d5 illegal:** we clean all the squares leaving those around the ♔

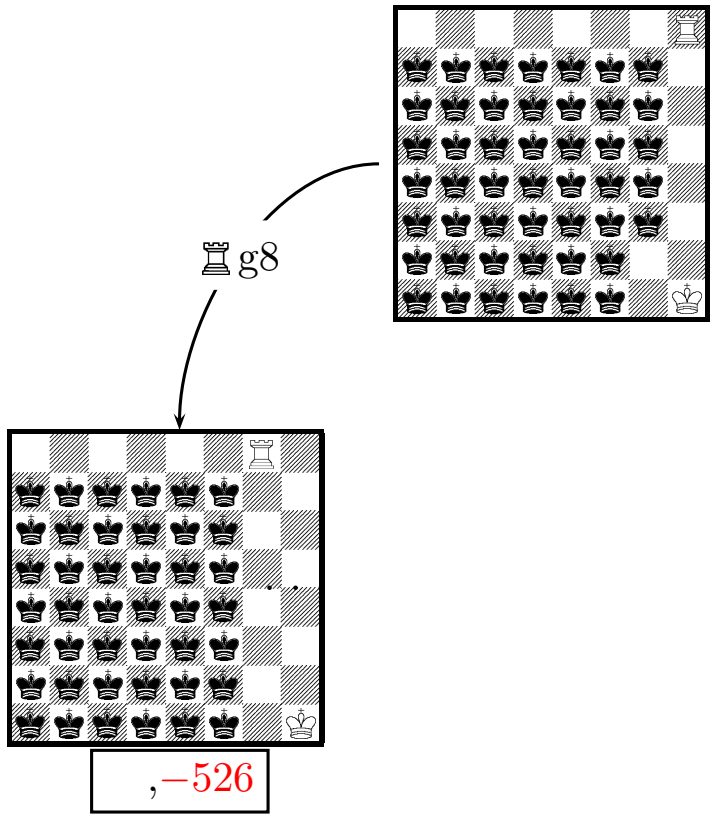
# The search algorithm

- Moves that jeopardize important pieces are pruned;
- recently-played moves are not played in direct sequence;
- the search algorithm uses:
  - a *static evaluation function*:
    - to enable the algorithm to prune bad moves;
    - to enable the algorithm to choose one of the three referee's answers;
  - a *recursive evaluation function*:
    - it is returned by the recursive call during the tree visit.
- The value of a metaposition is obtained by adding the recursive value to the static value.

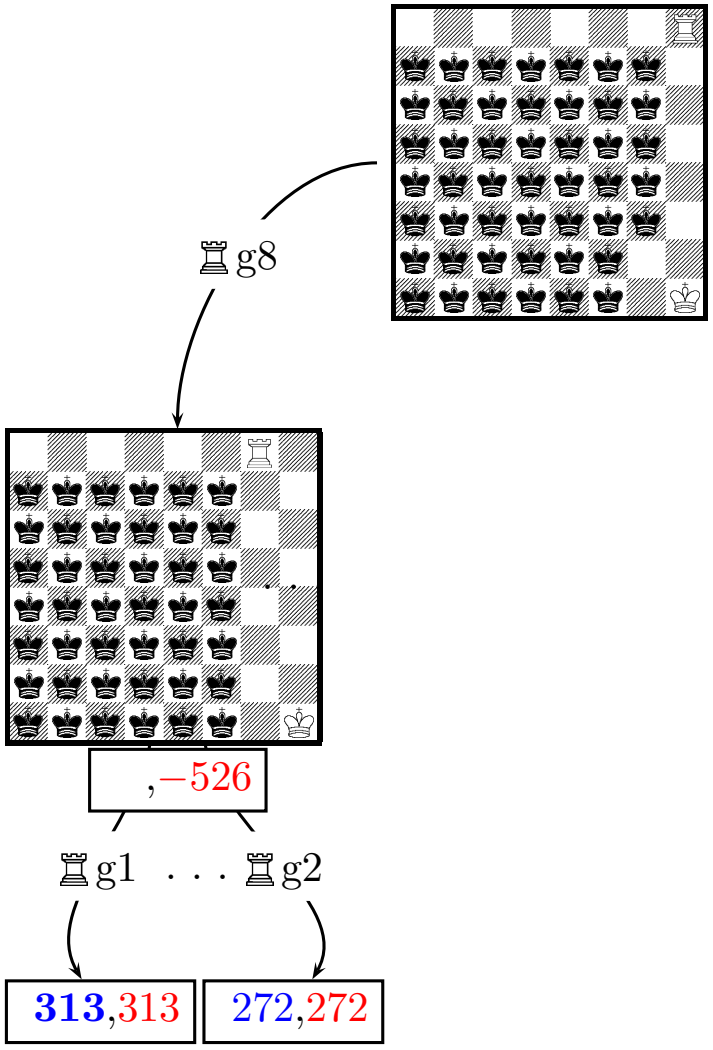
# Searching through metapositions



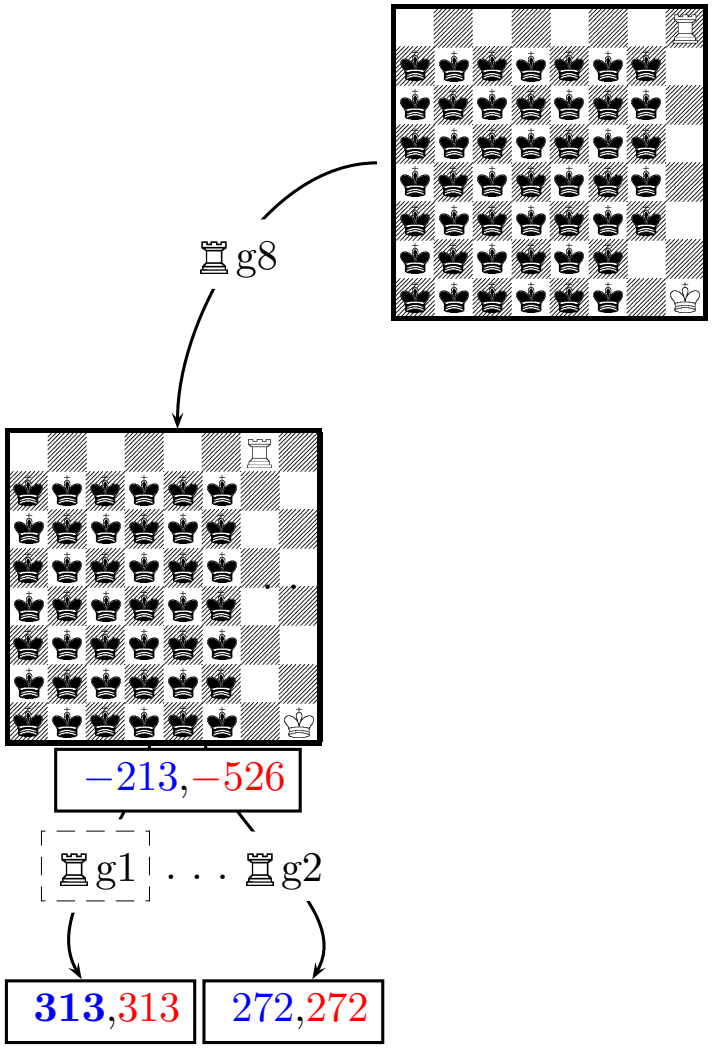
# Searching through metapositions



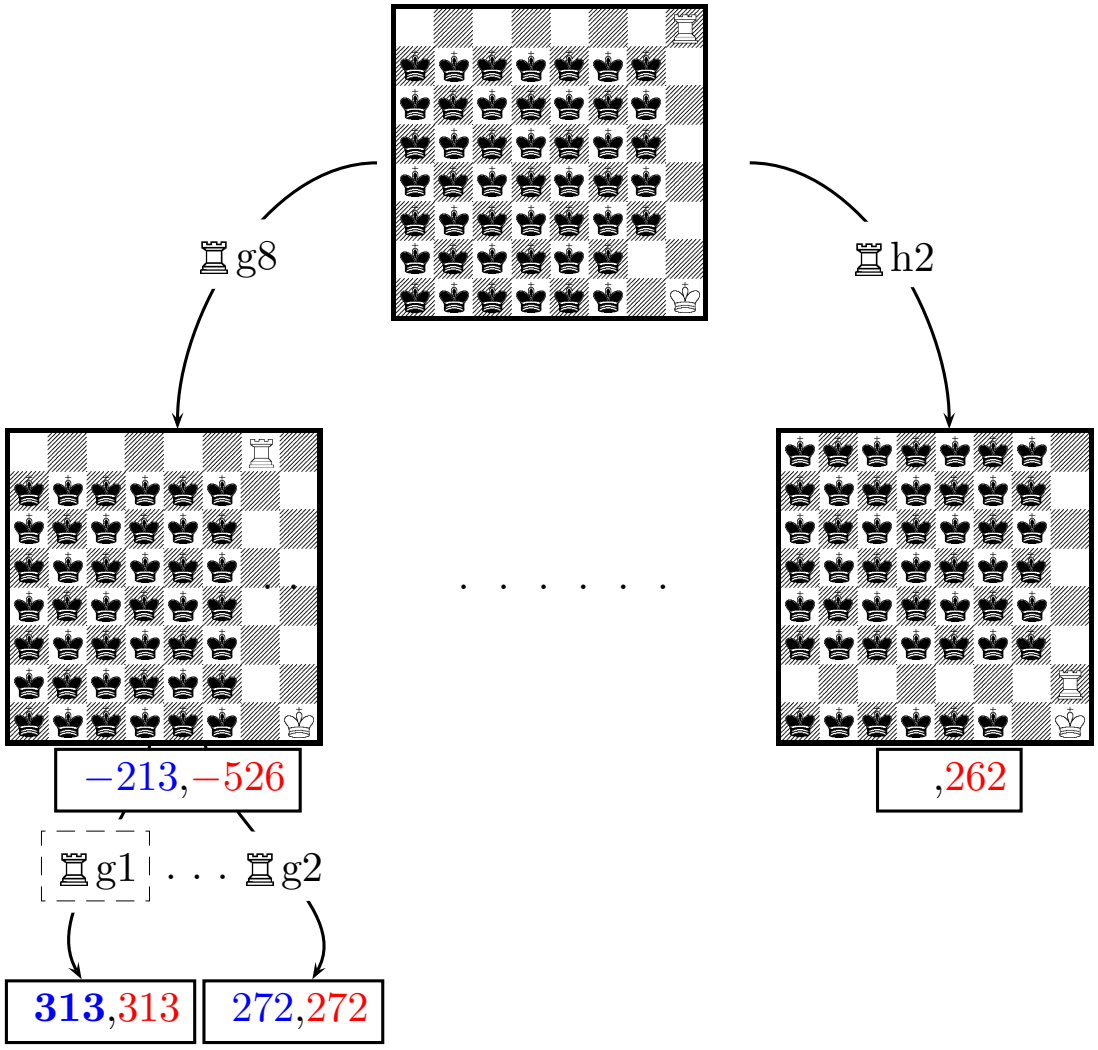
# Searching through metapositions



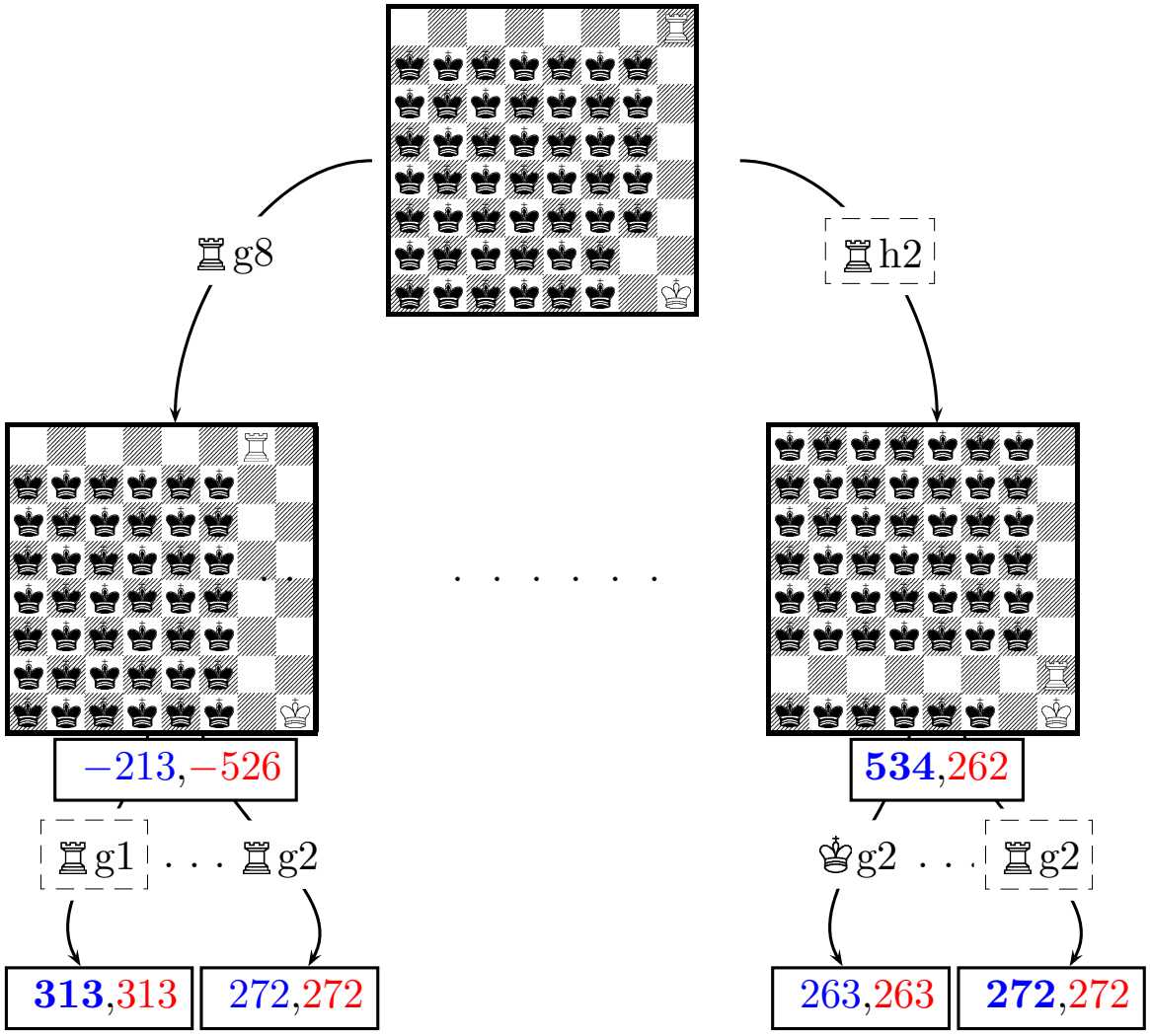
# Searching through metapositions



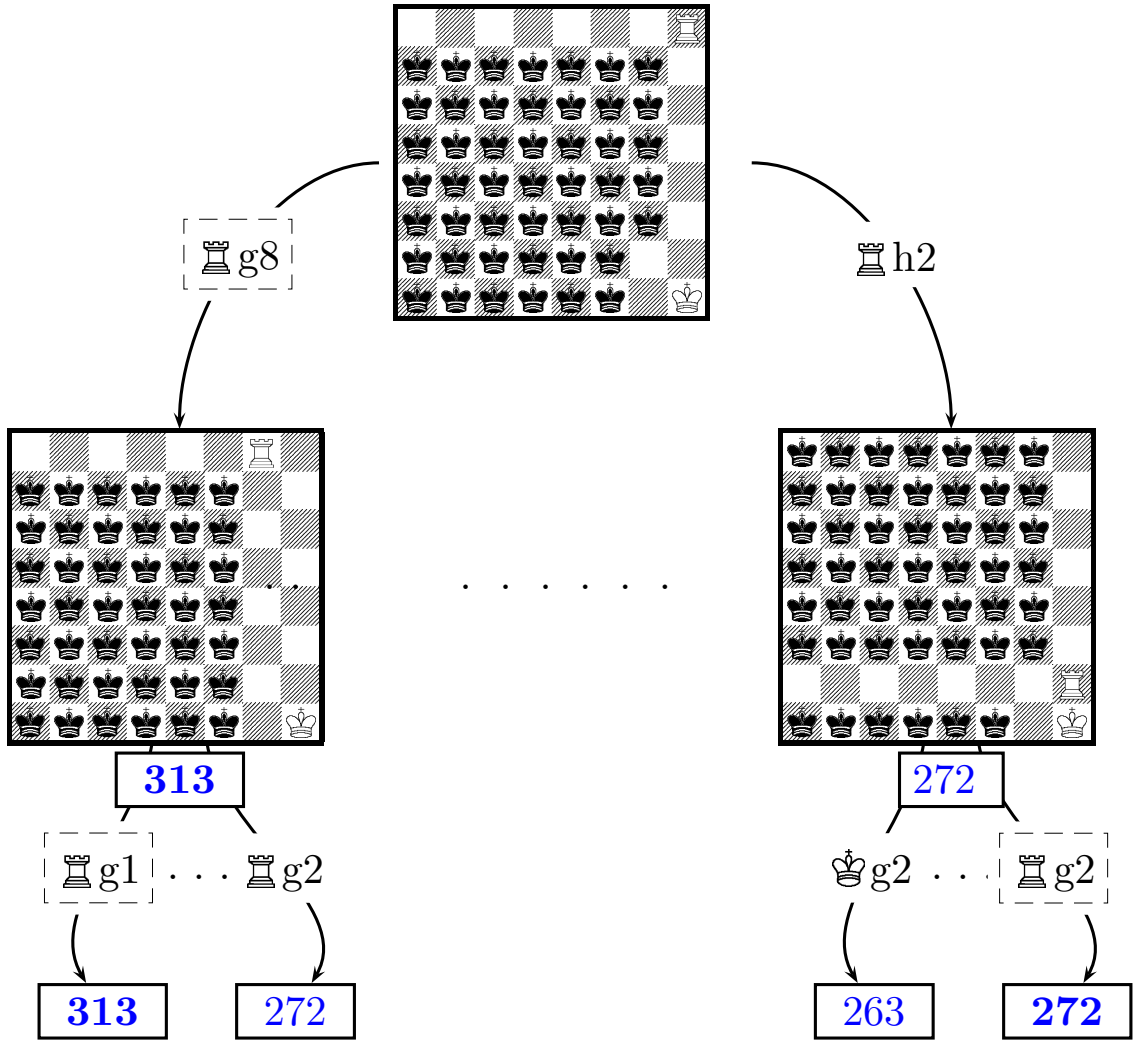
# Searching through metapositions



# Searching through metapositions



# Searching through metapositions



# The evaluation function

- contains the notion of *progress* which leads the player towards victory;
- is a linear weighted sum of features

$$\text{EVAL}(m) = w_1 f_1(m) + w_2 f_2(m) + ... + w_n f_n(m)$$

for example: $w_1 = -1, \quad f_1(s) = number\ of\ black\ kings$;

- returns the move which immediately gives *Checkmate*, if it exists;
- prunes moves that risk leading to a stalemate;
- is different according to each single ending.

# The rook ending (♔♖♚) (1/3)

The evaluation function for this ending considers $n = 6$ different features.

1. it avoids jeopardizing the Rook: $w_1 = -1000$ and $f_1$ is a boolean function which is true if white Rook is under attack;
2. it brings the two Kings closer: $w_2 = -1$ and $f_2$ returns the distance (number of squares) between the two Kings;
3. it reduces the number of black Kings on the quadrants of the board as seen from the Rook and it favors having the black Kings grouped together in as few quadrants as possible: $w_3 = -1$ and $f_3 = c \sum_{i=1}^{4} q_i$ where $c \in \{1, 2, 3, 4\}$ is a constant which counts the quadrants that contains a black King and $q_i$ counts the number of possible black Kings on $i^{th}$ quadrant;

# The rook ending (♔♖♚) (2/3)

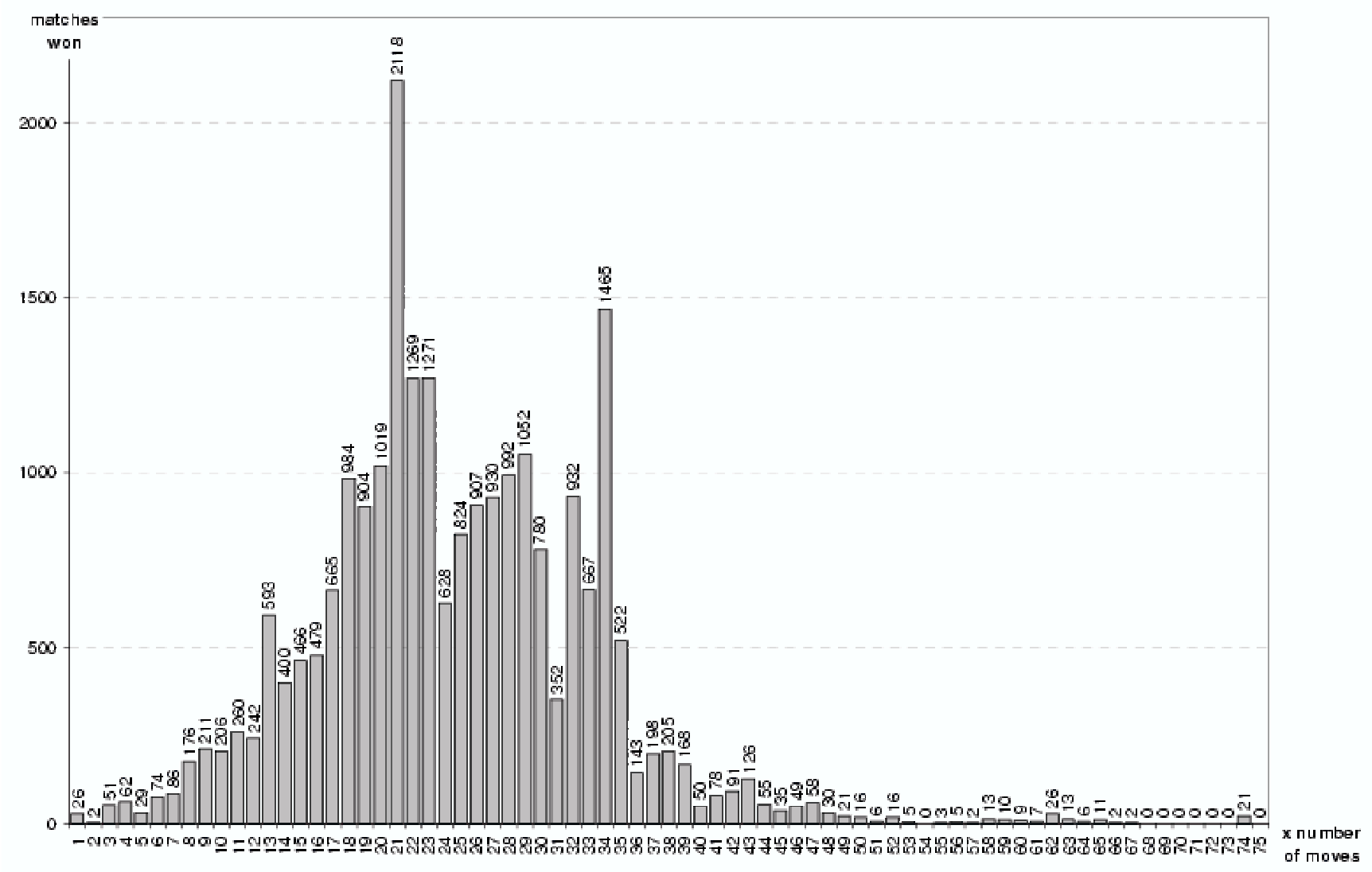
4. it avoids the black King to go between white Rook and white King: $w_5 = -500$ and $f_5$ is a boolean function which returns true if the black King is inside the rectangle formed by white King and white Rook on two opposite corners;
5. it keeps White pieces close to each other: $w_5 = +1$ and $f_5$ is a boolean function which returns true if the Rook is adjacent to the King;
6. it pushes the black King toward the corner of the board: $w_6 = +1$ and $f_6 = \sum_{i=0}^{63} v[i]$, where $v$ is a numerical 64-element vector, shown in the following slide, that returns a grade for each squares which possibly holds the black King or returns 0 otherwise.

# The rook ending (♔♖♚) (3/3)

The simple numerical matrix $v[]$ used in the $6^{th}$ features:

$$\begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & -2 & -4 & -4 & -2 & 0 & 0 \\ 0 & 0 & -4 & -4 & -4 & -4 & 0 & 0 \\ 0 & 0 & -4 & -4 & -4 & -4 & 0 & 0 \\ 0 & 0 & -2 & -4 & -4 & -2 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 \end{pmatrix}$$

# Histogram of ♔♖♚ ending




This histogram represents the number of moves needed to win each game, starting from metapositions with greatest uncertainty.

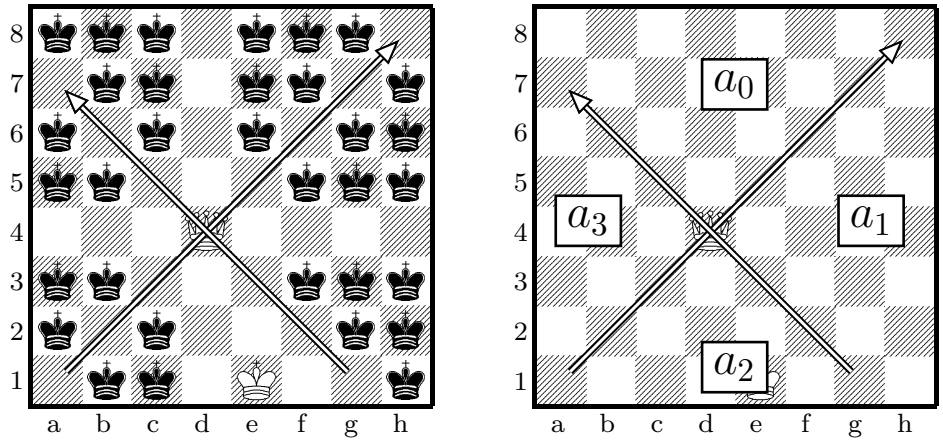
# The queen ending (♔♕♚)

7. it avoids metapositions where Queen risks to be captured: $w_7 = -100$ and $f_7$ is a boolean function that returns true if Queen is under attack;
8. it penalizes those metapositions with a big number of black Kings: $w_8 = -1$ and $f_8$ is equal to the number of black Kings on White's reference board;
9. it reduces the number of black Kings on the areas traced by the Queen's diagonals: $w_9 = -1$ and $f_9 = evalRhombArea(S)$ where

$$(1) \quad evalRhombArea(S) = c \cdot (a_0 + a_1 + a_2 + a_3)$$

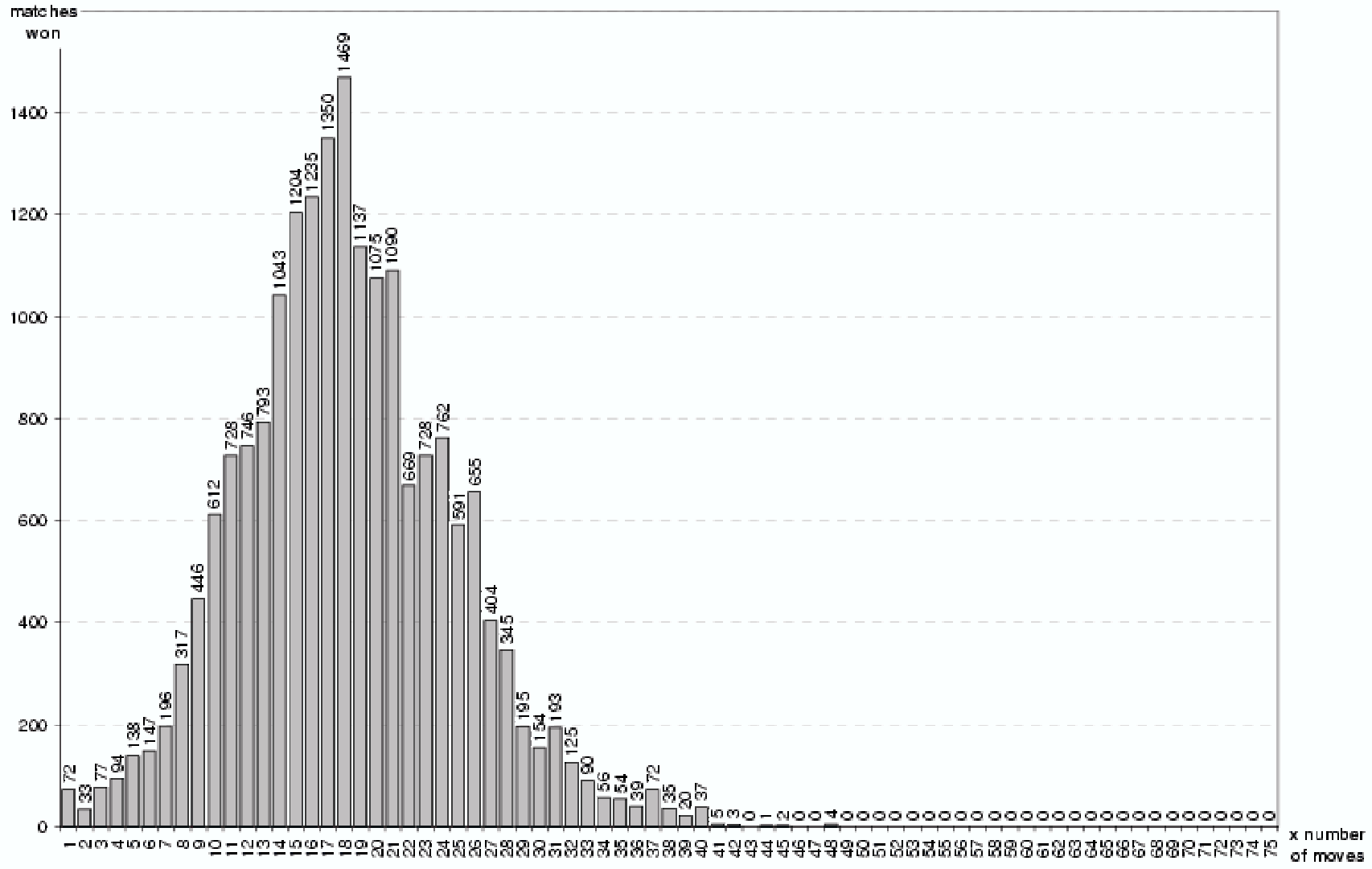
and $c \in \{1, 2, 3, 4\}$ is a constant which counts the areas that possibly contains a black King and $a_i$ $(i = 0, ..., 3)$ counts the number of possible black Kings on $i^{th}$ area.

# Description of evalRhombArea()




- Miniature on the right shows the 4 different areas if the queen is on d4;
- miniature on the left represents a metaposition with greatest uncertainty: in this case the function returns $4 \cdot (12 + 12 + 3 + 6) = 132$.

# Histogram of ♔♕♚ ending




This histogram represents the number of moves needed to win each game, starting from metapositions with greatest uncertainty.

# The ending with two Bishops (♔♗♗♚)

The evaluation function exploits the same subfunctions previously analyzed, but it assigns different weights:

1. it avoids jeopardizing the Bishop: $w_1 = -1000$ and $f_1$ is a boolean function which is true if white Bishop is under attack;
2. it brings the two Kings closer: $w_2 = -1$ and $f_2$ returns the distance (number of squares) between the two Kings;
3. it avoids the black King to "pass through" the border controlled by the Bishops: $w_3 = -500$ and $f_3$ is a boolean function which returns true if the black King is inside the rectangle formed by King and Bishop row or King and Bishop column;
4. it keeps close the white Bishops: $w_4 = +2$ and $f_4$ is a boolean function which returns true if the Bishops are adjacent to each other;

# The ending with two Bishops (♔♗♗♚)

5. it pushes the black King toward the corner of the board: $w_5 = +1$ and $f_5 = \sum_{i=0}^{63} b[i]$, where $b$ is a numerical 64-element vector, shown in figure, that returns a grade for each squares which possibly holds the black King or returns 0 otherwise.

$$\begin{pmatrix} 0 & -10 & -50 & -100 & -100 & -50 & -10 & 0 \\ -10 & -10 & -40 & -40 & -40 & -40 & -10 & -10 \\ -50 & -40 & -40 & -40 & -40 & -40 & -40 & -50 \\ -100 & -40 & -40 & -50 & -50 & -40 & -40 & -100 \\ -100 & -40 & -40 & -50 & -50 & -40 & -40 & -100 \\ -50 & -40 & -40 & -40 & -40 & -40 & -40 & -50 \\ -10 & -10 & -40 & -40 & -40 & -40 & -10 & -10 \\ 0 & -10 & -50 & -100 & -100 & -50 & -10 & 0 \end{pmatrix}$$

# The ending with two Bishops (♔♗♗♚)

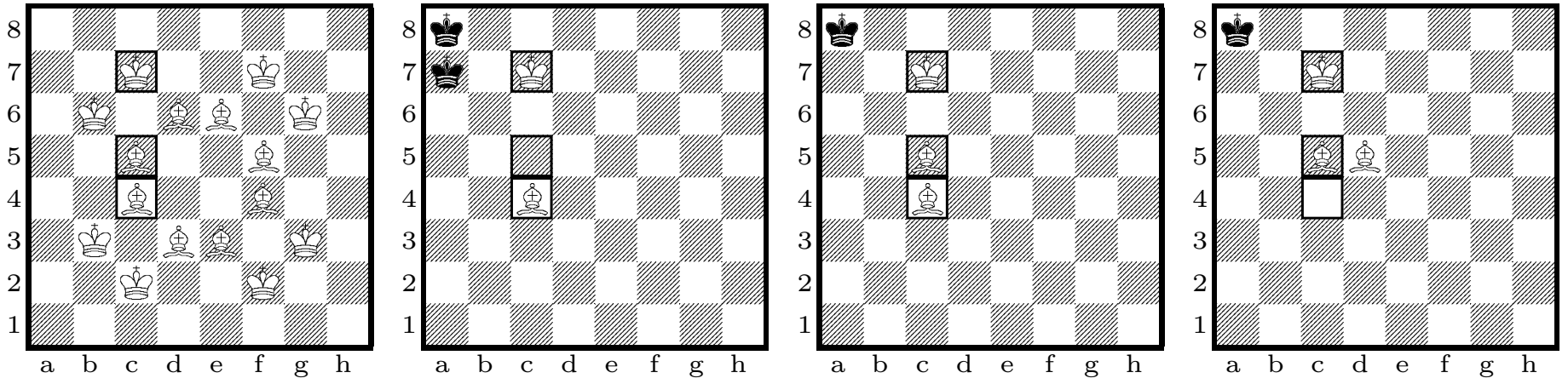
6. it keeps white King on the Bishop's row or column: $w_6 = +1$and $f_6$ is a boolean function which returns true if the King and the Bishop are on the same row or column;
7. it penalizes the metapositions where the Bishop risks to be captured: $w_7 = -100$ and $f_7$ is a boolean function that returns true if Bishops are under attack;
8. it penalizes the metapositions with a big number of black Kings: $w_8 = -1$ and $f_8$ is equal to the number of black Kings on White's reference board;
9. it reduces the number of black Kings on the areas traced by the Bishop's diagonals: $f_9 = evalRhombArea(m)$, and

   **if** $evalRhombArea(m) < -600$ $w_9 = -4$;

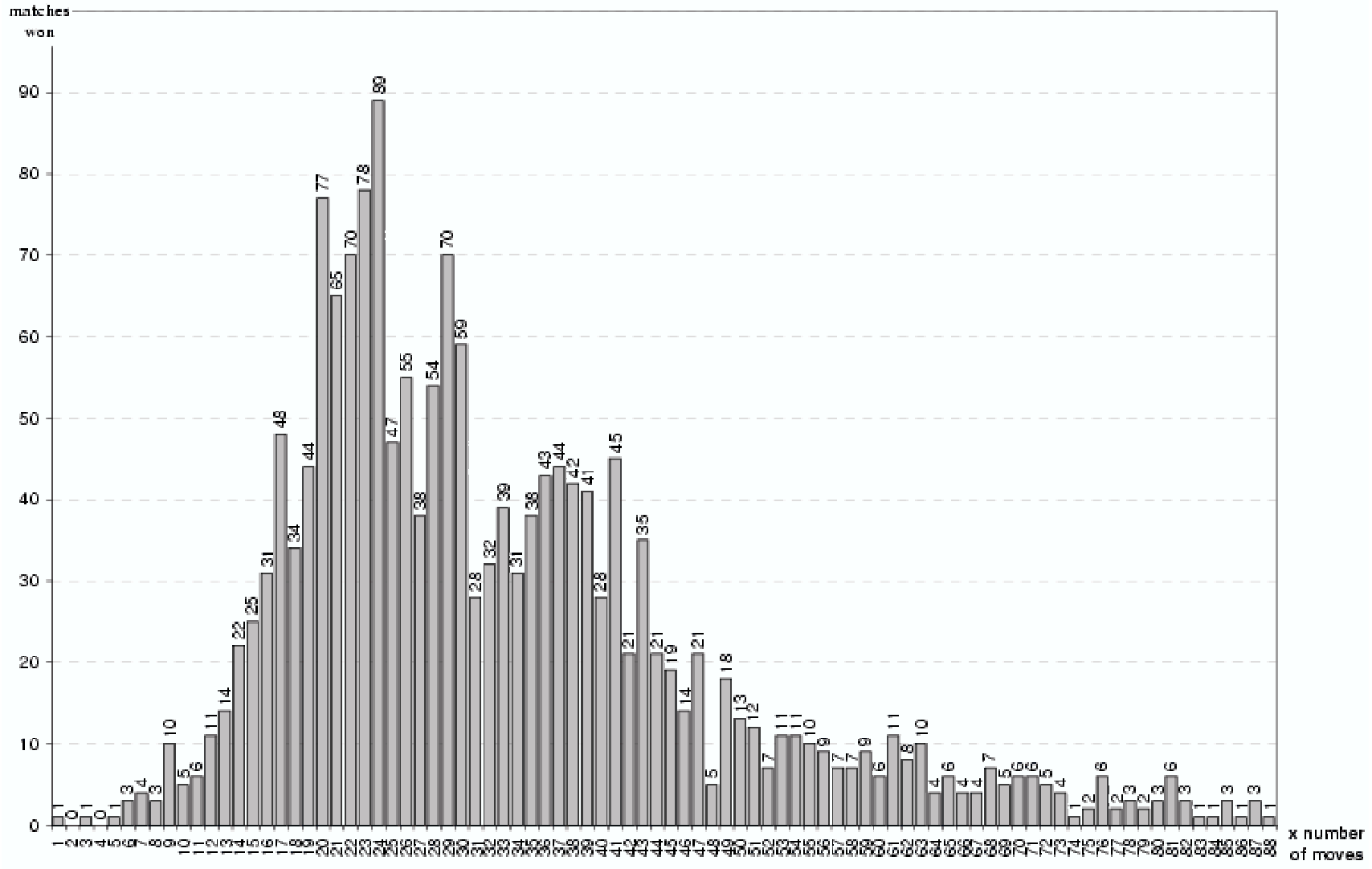
   **otherwise** $w_9 = \frac{1}{6}$

# The ending with two Bishops (♔♗♗♚)

10. it prefers some particular positioning (we will refer to with the term *key Bishops' positions*) for the white King and Bishops, highlighted in figure; for example ♔c7, ♗c4 and ♗c5. Therefore $w_1 0 = +30$ and $f_1 0$ is a boolean function which is true if the Bishops and the King are arranged in one of the key positions.



# Histogram of ♔♗♗♚ ending

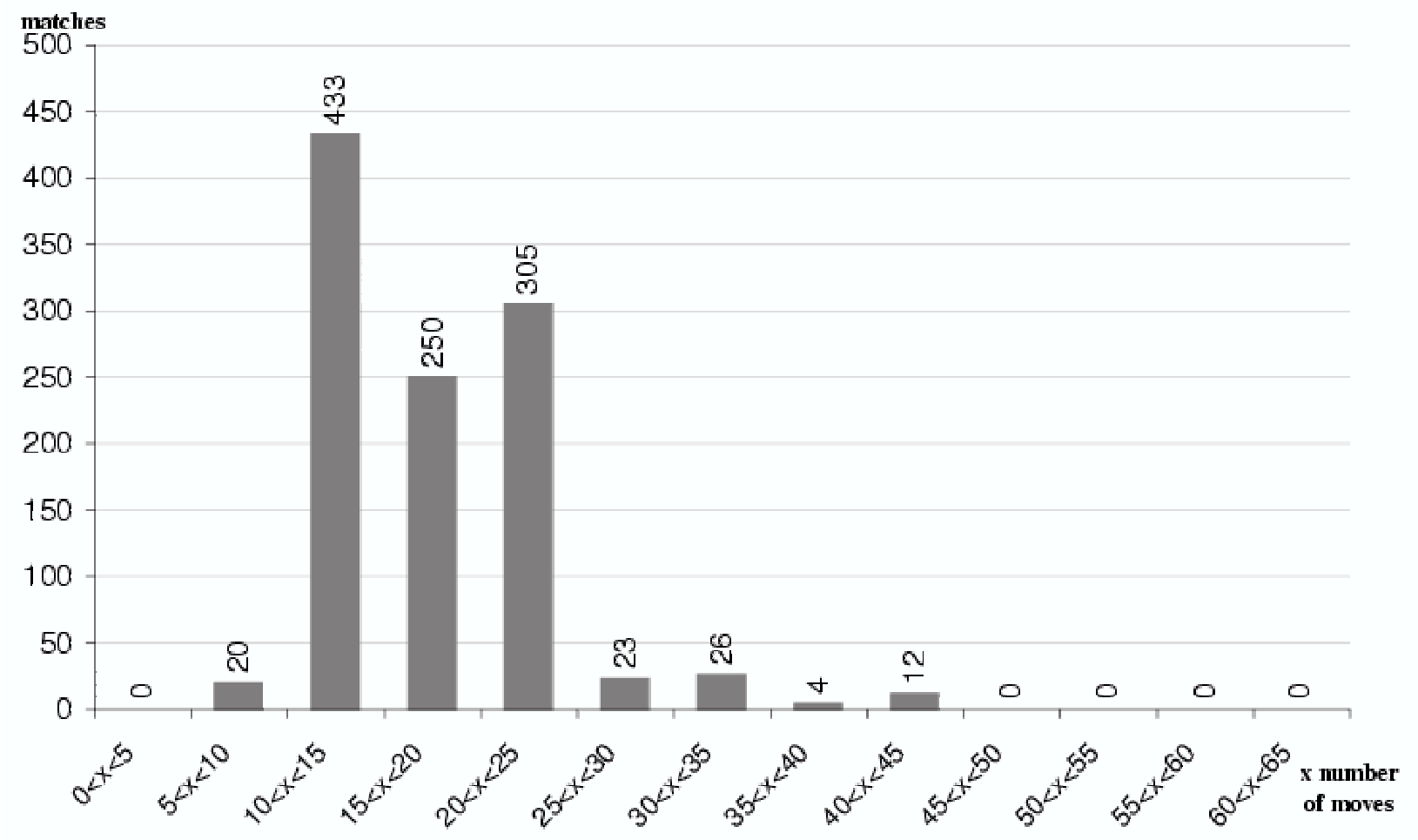



This histogram represents the number of moves needed to win each game, starting from random metapositions with greatest uncertainty.

# The pawn ending (♔♙♚)

1. it brings the Pawn adjacent to the King: $w_1 = -1$ and $f_1$ calculates the distance between King and Pawn;
2. it pushes the Pawn: $w_2 = +1$ and $f_2 = 2 \cdot (Pawn'srow)$;
3. it let the King above the Pawn: $w_3 = +1$ and $f_3 = (King'srow) - (Pawn'srow)$;
4. **If** $(Pawn's\ row == seventh\ row)$
   **if** $(Pawn's\ row) > (King's\ row)$ $w_4 = -1000$;
   **otherwise** $w_4 = +100$
5. **If** $(Pawn's\ row == sixth\ row)$
   **if** $(King\ is\ on\ the\ right\ of\ the\ Pawn)$
   $w_5 = +5 + rand()$;
   **if** $(King\ is\ on\ the\ left\ of\ the\ Pawn)$
   $w_5 = +5 + rand()$;

# Histogram of ♔♙♚ ending




This histogram represents the number of moves needed to win each game, starting from random metapositions with greatest uncertainty.

# Conclusion

- In our knowledge this is the first time that an evaluation function including a notion of progress has been defined for Kriegspiel.
- We have devoted special care to implement progress inside such an evaluation function.
- We have tested such a function on some simple endings, with good results except for the KBN vs K case.
- Future work will lead us to adapt the program to more complex endings, where both players have a larger number of pieces on the board.
- Our aim consists in writing a complete program for the whole game of Kriegspiel.