Universiteit Utrecht

Utrecht University

The Faculty of Science

The Department of Artificial Intelligence

Using computational argumentation for explainable chess AI

Thesis submitted in partial fulfilment of the requirements

for the Master of Sciences degree

Stefan van der Sman

Under the supervision of **Henry Prakken**

March 2024

Universiteit Utrecht

Utrecht University

The Faculty of Science

The Department of Artificial Intelligence

Using computational argumentation for explainable chess AI

Thesis submitted in partial fulfilment of the requirements

for the Master of Sciences degree

Stefan van der Sman

Under the supervision of **Henry Prakken**

Signature of student: _____ Date: _____

Signature of supervisor: _____ Date: _____

Signature of chairperson of the

committee for graduate studies: _____ Date: _____

March 2024

Using computational argumentation for explainable chess AI

Stefan van der Sman

Master of Sciences Thesis

Utrecht University

2024

Abstract

This thesis examines what role the combination of explainable artificial intelligence (XAI) and argumentation can play in the explanation of chess endgames. In the twenty-first century, the focus of the development of chess engines has been on their playing strength, meaning the engine's ability to win a game against other players (and later against other engines). With this focus, engines have achieved a skill level that is far beyond the reach of humans. On the highest levels, and even at the amateur level, ideas generated by engines are heavily influencing the game of chess. However, there is difficulty in interpreting the ideas behind engine generated moves. Using the ASPIC+ framework, this thesis tries to enable reasoning about chess endgame positions, specifically king and pawn versus king (KPvK) endgames. This is done by constructing a set of rules covering the entire space of this

specific endgame. First by approaching all positions with a pawn on the 7th rank, and then by expanding the model to include all other relevant positions. The explanations generated by the model are compared to the Chess XAI product DecodeChess, performing better for KPvK endgames. This makes it an interesting avenue for further research. It is possible to explain a specific subset (KPvK endgames) of all endgames using ASPIC+ as a base. However, it remains to be seen how this can be placed in the context of more complex endgames.

Acknowledgements

I am grateful for the excellent guidance and clear feedback of my supervisor, Prof. dr. mr. Henry Prakken. I also could not have undertaken this journey without the valuable feedback of my second examiner, Dr. Silja Renooij. I also want to mention Fay for proofreading and the great support in the final stretch.

Contents

List of Figures

Chapter 1

Introduction

After the worldwide successful Netflix show The Queen's Gambit, a fictional show about a prodigal chess player, Chess.com notes that the number of games played per day has doubled since the show's release in October 2020 [7].

When faced with a challenging chess position, finding the best move can prove to be a daunting task for players. However, once the underlying ideas behind the moves are understood, identifying the optimal move becomes much easier. This newfound knowledge can be applied to similar positions, improving a player's overall performance. Computers, on the other hand, approach chess problems differently, relying heavily on deep calculations. The resulting complex variations often fail to show the underlying knowledge in a way that helps humans comprehend them.

Chess enthusiasts often rely on materials like books and videos to study the game. These resources, presented by accomplished titled players, cover var-

ious topics such as opening theory, middle games, and endgames. While these materials enable in depth studying of the game, they fail to provide interaction with the student's questions. Students can, for questions about their own games, ask feedback from a teacher. Chess engines are another frequently used tool for feedback. With careful use, these engines can provide great insight in the position, but usage can also lead to a false sense of understanding. This false sense of understanding can be prevented when the ideas behind the moves are explained in depth. This is where explainable artificial intelligence (XAI) could be an interesting tool to enhance the communication between computer and player.

The main focus of this thesis is to discover how chess endgames can be explained using computational argumentation. Argumentation could be used to provide a bridge between computers and human understanding. Chess players discuss positions in a way that is similar to argumentation. One player proposes an idea and an explanation for why it works, this idea is then either accepted or refuted by a counterargument. When computers use structured argumentation to find a move it may be easier to explain to humans. Endgames are well suited because the endgame is the most concrete part of the game, yet they are still complex for humans to comprehend.

1.1 Research Questions

The main research question of this thesis is: Can computational argumentation be used for the explanation of chess endgames?

This question will focus on how current chess learning tools can be complemented with computational argumentation. For this purpose, only king and pawn versus king (KPvK) endgames will be discussed. This has three reasons. First, these endgames rely heavily on knowledge and ideas, making the moves of an engine difficult to interpret without an explanation. Secondly, positions in KPvK endgames lead to a clear result, it is either a draw or a win for the player with the extra pawn. This avoids the complexity that comes with positions where one player is slightly better. And finally, these type of endgames can be discussed without the requirement for a large amount of chess knowledge, leading to clearer examples. This clarity would be harder to guarantee with rook endgames, for example. The argumentation framework ASPIC+ will be used. ASPIC+ is used to build abstract argumentation frameworks. [15] The resulting argumentation frameworks correspond to the argumentation frameworks defined by Dung. [8] Such frameworks can be described as a directed graph where arguments refer to other arguments with an attack or defeat relation. This graph can then be interpreted to generate sets of acceptable arguments. ASPIC+ enables freedom in the definition of these aspects, making it applicable in many argumentation related contexts. A model that implements the framework in the desired context should be able to provide ways to formulate chess related logic and reasoning in a structured and reproducible way. This makes ASPIC+ well suited. The main question will be broken down into five sub questions:

1: How can argumentation for chess XAI be formalized using ASPIC+?
For a good model, arguments should be structured in a formalized fashion.

Formalizing the model enables it to be both reproducible and testable. This is attempted using ASPIC+. The knowledge base can be constructed using the available board position. The position on the board can then be used as cornerstone for the rules. Chess endgames are usually very black and white, this should ensure that clear rules are available in this context. Another point of interest is whether rule preferences are required. A rule for promotion, for example, should be preferred over other rules, but it also makes sense that such a rule is strict.

2: How detailed should arguments be?

The level of detail is an important aspect of an explanation. Two areas require attention. Firstly the depth of an explanation, this is where the explanation stops. One could consider pawn endgames in chess winning once a checkmate is on the board, but the game's result is clear many moves prior. Explanation to the checkmate seems redundant, but a clear cut-off point is required. The secondary part that should be discussed is the size of the steps in the explanation. Skipping steps can still lead to a logical argumentation for why a move is good, but it may omit important information required for a good explanation. Another added benefit may be increased performance if the approach turns out to be computationally heavy, resulting in a quicker generation of explanations.

3: What defines a good explanation for chess endgames?

Before discussing the application of the model to generate an explanation, some kind of metric to evaluate explanations needs to be discussed. The

metrics for XAI discussed by Hoffman [13] provide a good framework. Complimentary to this, the philosophy of Mark Dvoretsky in his book "Endgame Manual" [9] provides insight in how information should be structured for the studying of concepts in chess. The key emphasis here is on the ability to figure out the best move in any position that these concepts apply to, in opposition to simply understanding the discussed position.

4: Is argumentation using the defined approach sufficient for a good explanation?

With an argumentation model and a metric for explanation quality, it should be possible to generate an explanation and confirm the quality. In addition to addressing whether the explanation conforms to the metric, a comparison with an existing product. Decode Chess, a website that automatically generates explanations for chess games, is currently the only tool available in the realm of chess XAI. The software is not specifically tailored for endgames, but a shallow comparison is interesting nonetheless. This could give insight into whether the argumentation model could add value to already existing software. This all will be addressed with the following sub questions.

4.1: How does the model generate an explanation?

First, the generation of explanations should be explored. The most obvious solution is a direct mapping of applicable rules to an explanation. This should then result in a structured description. Another point of interest is whether all information is required and all required information is extracted from the model. One potential factor that could result in missing information

is inverse logic. Humans may make a choice because some concept does not apply to the position, for example, the defending king is not in a position to capture the pawn. Statements like this may or may not be required in the generating of an explanation.

4.2: How does a generated explanation get evaluated by the established metrics?
This question will be answered by comparing the model to the established metrics in question 3.

4.3: How does the model perform on those metrics in comparison to the software Decode Chess?
A comparison will be made to Decode Chess to see what the argumentation model has to offer as a complement to commercially available chess XAI. This comparison may not be completely fair because Decode Chess is built as a product that gives insight into all stages of the game. The explanations may therefore be less detailed. The goal of this question, however, is to figure out whether the argumentation approach improves on available tech.

1.2 Structure of this thesis

Section 1.1 introduced the research questions.

In chapter 2 the related work is discussed. Here the relevant chess XAI and argumentation literature will be introduced in section 2.1. Furthermore, the chess XAI product Decode Chess will be discussed in section 2.2. Although this is not strictly literature it is a product that is relevant for this research. The relevance of these items for this thesis will then be briefly discussed. After this, the important background on argumentation will be discussed in section 2.3.

In chapter 3, the explanation model will be discussed. First, the underlying logic model will be addressed. This logic will be split into pawns on the 7th rank and pawns in general. This is because some slightly different concepts apply to pawns on the 7th rank, and it is a clear and consistent domain to discuss before approaching the entire set of endgames.

Chapter 4 addresses how the model can be used to construct explanations. This is done by discussing several positions worked out using the model established in chapter 3.

In chapter 5, the quality of an explanation will be discussed. Following this, the established metrics in this chapter will be used to evaluate the explanations generated in chapter 4. After this, a brief comparison with an explanation from DecodeChess is discussed.

Finally, in chapter 6 the conclusions are drawn and limitations and further

research are discussed.

Chapter 2

Related Work

No research has been done on using argumentation for explainable AI (XAI) for chess endgames. There are several studies on the implementation of XAI for chess in general. Other relevant literature covers computational argumentation and chess endgame books (they give insight in traditional chess teaching). This section gives a summary of related work and previous research.

2.1 Chess XAI

The general topic of this thesis is the application of XAI for chess. There are several relevant papers in this category. First, there is a paper by Guid et al. [12], that focuses on the implementation of a tutoring system for chess endgames. This paper describes the components required for such a system, a domain model that contains expert knowledge on the specific endgame: a

tutoring model with pedagogical knowledge about the teaching strategies for the endgame, and a student model with knowledge about the progress of the student. The first two models are relevant for this thesis. If an endgame is solved with argumentation, the domain model will be used to generate these arguments. In the paper, the domain model contains rules that describe a goal that is achievable in a set number of moves. This approach is interesting for complex endgames, as move sequences are usually the only way to figure out how to make progress in a position. In more straightforward endgames, however, it may benefit explanation detail to approach positional concepts in greater depth. Furthermore, the domain model discussed in the paper depends partly on interactivity with the student. In chess endgames, however, the calculation of long sequences cannot entirely be avoided even if the endgame mainly relies on theoretical endgame knowledge, for rules in relation to calculation the paper provides good guidance. The explanation of the rules in the domain is encapsulated by the tutoring model in the paper. The tutoring model compares student input with the domain model. The student model they describe is used to give feedback on incomplete knowledge, for this to be useful it needs to be part of an interactive system. The model described in this thesis will not be interactive, but attempt to generate a full explanation from the provided position, therefore the student model discussed in the paper will not be taken into account.

In another paper, an endgame tutor is discussed by Gadwal et al. [10]. This paper discusses the use of higher level chess concepts such as plans or goals

for explaining a position. Such goals can for example entail winning a piece, but more useful are motifs like pinning a piece to the king (which then leads to winning the piece). Having such motifs incorporated allows for searching them, simplifying the search drastically. The tutor is focused on bishop pawn endgames and makes plans in a structured way. The plan is presented as an object with slots that represent aspects of the plan: side to play (black or white), type of plan (expert or student), applicability predicate, feasibility predicate, better-goals, holding-goals, move constraints for both sides, and decidability of the plan. The applicability predicate consists of board features, for example, pawn structure (blocked or passed pawns), that can be applied for selecting a plan relevant to the position. Feasibility predicates indicate how likely a plan is to succeed. The criterion of success or the purpose of the plan is described by its better-goals. The criterion of failure of the plan is defined by its holding-goals. For example, if a side has a passed pawn (applicability) and is not controlled by the bishop (feasibility) then the plan can be to 'queen the pawn' (better-goal), and at the same time the pawn should be safe (holding-goal), and the success of the plan decides the game (decidability). The move-constraints are represented as Mcx and Mcy, where Mcx defines the constraints on the moves for the side to play and Mcy defines the constraints on the opponent's moves in order to satisfy the goals (better-goals and holding-goals) of the plan. Both correct and incorrect solution strategies can be modelled by the system. The feedback given by the system is conceptual, making use of plans and goals instead of only showing the next best move. The system allows students to explore incorrect strategies and provides counter-strategies. This paper approaches positions in a way

that is relevant for argumentation. A goal based approach in the build-up of argumentation rules should allow for clear and concise explanations. All current research is focused on a specific position where certain rules apply. The paper uses something comparable to argumentation. Counterarguments are essential to form an explanation that looks like a human dialogue. The endgame tutor is, however, only applied to bishop endgames. This makes the knowledge discussed in the paper less relevant to the endgames discussed in this thesis. Furthermore, it is important to note that the tutor does not provide full textual explanations but in some way depends on interaction with the student. As this is an approach used in more papers, it is interesting to see whether this is an actual requirement, or whether a full explanation can be generated with only an initial query.

The paper from Bratko et al. [11] states: "Our results also clearly confirm one aspect of classical De Groot's model of chess thinking. Namely, that satisfactory calculation is not possible without detection of motifs.". The paper discusses the importance of correct calculation and the knowledge of motives for finding the best move. The conclusion is that knowledge of the motifs in the position is required for correct calculation. From this one could infer that giving concrete variations of moves does little on explaining a chess position, for an explanation the focus should be on the motives. Motives in the endgame can be fairly advanced, like corresponding squares and distant opposition for example, this is where current attempts of chess XAI fail to form a clear and concrete explanation. As stated earlier the goal of this

thesis is not to approach endgames using long complex sequences of moves, this paper confirms that conceptual knowledge is important. Implementing a model using chess knowledge from the endgame manual by Dvoretsky [9] as motives and goals should help in bridging this gap.

The research from Das and Chernova [6] is focused on leveraging rationales to improve human task performance. They make use of a rationale generating algorithm (RGA) which converts the utility function of Stockfish (Chess AI program) to a rationale. The importance of aspects in the user interface is addressed, for example, highlighting specific squares and the piece that should be moved. The paper addresses a highlight function, this can be used to highlight areas of interest on the board, but sometimes gives away too much information and enables finding the move without enough understanding of the position by the player. This function highlights that it is important to not give away too much information, as this will not help the student in understanding the position. This is similar to the advice many chess teachers give, looking at the computer before analysing the game yourself gives a false sense of understanding. The research is focused on translating the reasoning of an algorithm to human reasoning. Humans, however, mainly make use of motives to decide what moves have potential in a position, as stated in Bratko et al.[5]. The RGA does have the capability of generating some motives based on the parameters of stockfish, but more complex motives that are required for understanding of the endgame can often not be derived from the parameters of a chess engine and either require deep calculation of more lines than humanly possible or a way of mapping complex motives

Figure 2.1: A complex pawn endgame analysed by DecodeChess

onto the position. It is interesting to look at the balance between textual explanation and highlighting square. An explanation model may benefit from highlighting essential squares in some motifs.

2.2 Chess XAI products

There are little chess XAI products on the market. The most notable, and only program explicitly marketed as chess XAI, is Decode Chess [2], a software program on a website that makes use of a strong chess engine (stockfish) and unknown underlying algorithms to explain chess positions in a rich way, making use of intuitive language. Even though the software has some limitations, it would be interesting to use it as a comparison to see whether an argumentation-based approach can complement existing software. In the example in the image in figure 2.1 one could argue that the explanation is not rich and not making use of intuitive language. A good human-like explanation in this position would be, that Kg2 allows white to get a favourable position because whatever black does, white can get the distant opposition

or outflank on the left side of the board.

Decode Chess seems to work well for tactical sequences, as these patterns are used in the natural language of the program. Complex plans and concepts like distant opposition, however, cannot be explained by the program, and therefore the use of the program is limited. The patterns that appear in the opening and middle game are for the most part tactical or to prevent tactical plans by the opponent. The main lack of explanation involves endgame concepts as these are an entirely different area of chess, patterns can be very concrete but also hard to conceptualize, especially for computers that mostly rely on a brute force approach, which Decode Chess seemingly relies on often. This is the reason to focus on explainable AI for endgames.

2.3 Argumentation

In this thesis, computational argumentation is used to provide a framework to work within. Argumentation can be used to evaluate statements by considering arguments for and against the statement. A well-constructed argument makes use of enough information for a sensible conclusion. This richness in information makes argumentation inherently suitable for explanation.

The basis of an explanation will rest on whether a justified argument can be constructed for a position is winning or drawn.

Random statements about a chess endgame on their own are not very useful, it is the relations between them that enable structured argumentation. To provide structure, this thesis tries to prove that chess endgames, with em-

phasis on king pawn versus king endgames, can be structured and explained with the ASPIC+ framework.

2.3.1 Defeat relations

In argumentation a conflict between two arguments can be described in several ways, in this document the word 'defeat' will be used. The notion of defeat can be used to describe the relative strength between arguments.

2.3.2 Abstract Argumentation Frameworks

ASPIC+ is used to generate abstract argumentation frameworks. An abstract argumentation framework (AF) as defined by Dung [8] is a fully abstract framework where both the structure of arguments and the grounds for defeat are left unspecified. The abstract nature of the framework allows it to be applied in various settings. An AF can be defined the following way:

Definition 1 (Abstract argumentation frameworks.) *[16]*

1. *An abstract argumentation framework (AF) is a pair $\mathcal{A}, \mathcal{D}$, where $\mathcal{A}$ is a set of arguments, and $\mathcal{D}$ a binary relation of defeat on $\mathcal{A}$.*

2. *We say that a set S of arguments defeats an argument A iff some argument in S defeats A; and S defeats a set S' of arguments iff it defeats a member of S'.*

Using this definition, a directed graph can be constructed where arguments are related to other arguments using the defeat relations. This graph can be used to determine the set of accepted arguments.

To determine this set, it is of importance to decide the semantics. The arguments in the graph can be labelled in three ways: in, out, or undecided. An argument is labelled as in if and only if all attackers are out. An argument is out if and only if some argument that attacks it is in. All other arguments are labelled undecided. ASPIC+ uses semantics to determine extensions. These semantics provide the criteria under which arguments are accepted. For this specific use case, it is suitable to use grounded semantics. The reason for this is that grounded semantics minimizes the arguments labelled in. This is helpful for explaining chess positions because there is no room for alternative ways of labelling the arguments, preventing ambiguity. Grounded status assignments can be formally defined the following way, using status assignments:

Definition 2 (Grounded status assignments.) *[16] Let $AF = (\mathcal{A}, \mathcal{D})$ be an abstract argumentation framework and In and Out two subsets of $\mathcal{A}$. Then (In, Out) is a status assignment on the basis of AF iff $In \cup Out = \emptyset$ and for all $A \in \mathcal{A}$ it holds that:*

1. *A is in (that is, $A \in In$) iff all arguments defeating A (if any) are out.*
2. *A is out (that is, $A \in Out$) iff A is defeated by an argument that is in.*

A status assignment S = (In, Out) is grounded iff there is no status assignment S ' = (In', Out') such that $In' \subset In$.

In grounded semantics, an argument is justified on the basis of AF iff it is in the grounded extension of AF.

2.3.3 G-Game

With the semantics defined, it is possible to create the set of accepted arguments. This does, however, provide little information about who is correct in a dispute, because the status of individual arguments is not clear. To determine the status of individual arguments, the argument game will be used. In an argument game, a proponent and opponent take turns to make statements. The proponent starts with an argument, and all other moves consist of players responding with an argument that defeats the previous one. Prakken defines the argument game in the following way:

Definition 3 (Moves, disputes, and protocols.) *[16] Given an argumentation framework $AF = (\mathcal{A}, \mathcal{D})$ we define the following notions.*

- *The set M of moves consists of all pairs (p, A) such that $p \in \{P, O\}$ and $A \in \mathcal{A}$;*
  for any move (p, A) in M we denote p by $pl(m)$ and A by $s(m)$.

- *The set of $M \leq \infty$ of disputes is the set of all sequences from M and the set $M < \infty$ of finite disputes is the set of all finite sequences from M.*

- *A protocol is a function that specifies the legal moves at each stage of a dispute. Formally, protocol is a function Pr with domain a nonempty subset $\mathcal{D}$ of $M < \infty$ taking subsets of M as values. That is:*

$Pr : D \longrightarrow Pow(M)$ such that $D \subseteq M < \infty$. The elements of D are called the legal finite disputes. The elements of $Pr(d)$ are called the moves allowed after d. If d is a legal dispute and $Pr(d) = \emptyset$, then d is said to be a terminated dispute. Pr must satisfy the following conditions for all finite disputes d and moves m:

1. *$d \in D$ and $m \in Pr(d)$ iff $d, m \in D$;*
2. *if $m \in Pr(d)$ then $pl(m) = P$ if d is of even length, otherwise $pl(m) = O$.*

- *A winning function is a partial function of type $W : D \longrightarrow \{P, O\}$*

The exact implementation of the game depends on the semantics. It needs to be addressed if moves need to be strictly defeating, if backtracking is allowed, whether moves may be repeated, and if players may make a move that is defeated or defeats an earlier statement.The G-Game is a version of the argument game for grounded semantics. The G-Game can be defined the following way:

Definition 4 (Proof theory for grounded semantics.) *[16] A dispute satisfies the G-game protocol iff it satisfies the following conditions.*

1. *Moves are legal iff in addition to Definition 5.1.1 they satisfy the following conditions.*
   a) *Proponent does not repeat his moves; and*
   b) *Proponent's moves (except the first) strictly defeat opponent's last move; and*

c) Opponent's moves defeat proponent's last move.

2. *A player wins a dispute iff the other player has no legal moves*

With these definitions, argument games can be played, but the perfect strategy for either player has not yet been addressed. To ensure a strategy is winning, the proponent should have a reply ready for all responses of the opponent, or formally:

Definition 5 (Strategies.) *[16]*

1. *A strategy for player p is a tree of disputes only branching after p's moves, and containing all legal replies of* $\bar{p}$

2. *A strategy for p is winning iff p wins all disputes in the strategy.*

For the proponent to have a winning strategy, all the leaves of the strategy's tree should be a move by the proponent. Once a winning strategy is found for a g-game, the arguments used by the proponent can be classified as in and the arguments used by the opponent are labelled out. This directly follows from the fact that leave nodes are not defeated by any argument. This gives a grounded labelling that can be used as a cornerstone for an explanation.

When talking about chess endgame positions, the simplest approach is to find a winning strategy that supports the objective evaluation of the position. These objective evaluations are known for KPvK endgames and can be acquired from endgame databases or chess engines. This makes generating a winning strategy simpler as for a winning position one can simply look for

a winning strategy where the position is winning. For a drawn position, the winning strategy for a draw is used instead.

2.3.4 ASPIC+ Framework

The ASPIC+ framework is a framework that is used for structured argumentation. ASPIC+ enables generating abstract argumentation frameworks as described in [8].

One important aspect of ASPIC+ is inference rules. A rule of inference is a rule that takes premises and returns a conclusion. For example, from "if p then q" ,"p" you can conclude q. As stated by Prakken [17], an argument can be either strict or defeasible. A strict rule in a knowledge base always holds. A defeasible rule generally holds, but it can be incorrect given a certain scenario. If an inference makes use of a defeasible rule, the conclusion is also defeasible.

ASPIC+ makes use of two types of inference rules, strict and defeasible rules. When the conditions of a strict rule are met, the outcome is definite and leaves no room for ambiguity. This is in opposition to defeasible rules, here the rule is generally the case, but exceptions may occur. An example is, birds fly. This tends to be the case, but there are counterexamples possible, like a penguin for example. Generally, the rule is still correct.

To enable the use of ASPIC+ the language $\mathcal{L}$ has to be defined. The sets of strict ($\mathcal{R}_s$) and defeasible ($\mathcal{R}_d$) rules have to be specified. These sets can be empty.

This leads to the following formal definition as described by Prakken [17]:

Definition 6 *(argumentation system) An argumentation system is a tuple* $AS = (\mathcal{L}, ^-, \mathcal{R}, n)$

- $\mathcal{L}$ *is a nonempty logical language e with a unary negation symbol* $\neg$*,*
- $^-$ *is a contrariness function from* $\mathcal{L}$ *to* $2^{\mathcal{L}}$*,*
- $\mathcal{R} = \mathcal{R}_s \cup \mathcal{R}_d$ *is a set of strict* ($\mathcal{R}_s$)*, and defeasible* ($\mathcal{R}_d$) *inference rules such that* $\mathcal{R}_s \cap \mathcal{R}_d = \emptyset$
- n *is a partial function such that* $n : \mathcal{R}_d \rightarrow \mathcal{L}$*.*

To be able to use ASPIC+ it is also required to specify a knowledge base. A knowledge base contains information about the premises. These are split up into axioms, which cannot be attacked, and ordinary premises, which are not certain and can therefore be attacked. Below is a formal definition.

Definition 7 *(knowledge base) A knowledge base in an* $AS = (\mathcal{L}, ^-, \mathcal{R}, n)$ *is a set* $\mathcal{K} \subseteq \mathcal{L}$ *consisting of two disjoint subsets* $\mathcal{K}_n$ *(the axioms) and* $\mathcal{K}_p$ *(the ordinary premises)*

Another concept that requires a definition is contrariness. Contrariness means that two propositions P and Q cannot both be true in the same interpretation.

Definition 8 *(logical language) [17] Let* $\mathcal{L}$*, a set, be a logical language and* $^-$ *a contrariness function from* $\mathcal{L}$ *to* $2^{\mathcal{L}}$*. If* $\phi \in \bar{\psi}$ *then if* $\psi \notin \bar{\phi}$*, then* ϕ *is called a contrary of* ψ*, otherwise* ϕ *and* ψ *are called contradictory.*

Combining a knowledge base and an argumentation gives an argumentation theory.

Definition 9 *(argumentation theory) An argumentation theory is a tuple* $AT = (AS, \mathcal{K})$ *where* AS *is an argumentation system and* $\mathcal{K}$ *is a knowledge base in* AS

The arguments that can be constructed from a knowledge base are defined next as described by Prakken [17]. Arguments can be constructed by combining inference rules into trees. Arguments are built using subarguments, that support intermediate conclusions. With the function, $Prem$ all formulas (premises) in $\mathcal{K}$ are returned that are used to construct the argument. The function Sub returns all its subarguments. The function $Conc$ returns the conclusion of an argument. The function $DefRules$ returns all defeasible rules. The function $TopRule$ returns the last inference rule used in the argument. Prakken [17] gives the following formal definition for this:

Definition 10 *(argument) An argument* A *on the basis of a knowledge base* $\mathcal{K}$ *in an* $AS = (\mathcal{L}, ^-, \mathcal{R}, n)$ *is*

(1) ϕ *if* $\phi \in \mathcal{K}$ *with*

$Prem(A) = \{\phi\}$,

$Conc(A) = \phi$,

$Sub(A) = \{\phi\}$,

$DefRules(A) = \emptyset$,

$TopRule(A) = undefined$.

(2) $A_1,\ldots,A_n \rightarrow \psi$ *if* $A_1,\ldots,A_n$ *are arguments such that there exists a*

strict rule

$Conc(A_1), \ldots, Conc(A_n) \rightarrow \psi$ *in* $\mathcal{R}_s$,

$Prem(A) = Prem(A_1) \cup \ldots \cup Prem(A_n)$,

$Conc(A) = \phi$,

$Sub(A) = Sub(A_1) \cup \ldots \cup Sub(A_n) \cup A$,

$DefRules(A) = DefRules(A_1) \cup \ldots \cup DefRules(A_n)$,

$TopRule(A) = Conc(A_1), \ldots, Conc(A_n) \rightarrow \psi$.

(3) $A_1, \ldots, A_n \Rightarrow \phi$ *if* $A_1, \ldots, A_n$ *are arguments such that there exists a defeasible rule*

$Conc(A_1), \ldots, Conc(A_n) \Rightarrow \psi$ *in* $\mathcal{R}_d$,

$Prem(A) = Prem(A_1) \cup \ldots \cup Prem(A_n)$,

$Conc(A) = \phi$,

$Sub(A) = Sub(A_1) \cup \ldots \cup Sub(A_n) \cup A$,

$DefRules(A) = DefRules(A_1) \cup \ldots \cup DefRules(A_n) \cup Conc(A_1), \ldots, Conc(A_n) \Rightarrow \psi$,

$TopRule(A) = Conc(A_1), \ldots, Conc(A_n) \Rightarrow \psi$.

Definition 11 (Argument properties) *[16] An argument A is strict if* $DefRules(A) = \emptyset$*; defeasible if* $DefRules(A) \neq \emptyset$*; firm if* $Prem(A) \subseteq K_n$*; plausible if* $Prem(A) \cap Kp \neq \emptyset$ *. We write* $S \vdash \phi$ *if there exists a strict argument for* ϕ *with all premises taken from S, and* $S \sim \phi$ *if there exists a defeasible argument for* ϕ *with all premises taken from S.*

2.3.4.1 Attack and defeat

To enable the generating of Dung-style abstract argumentation frameworks using ASPIC+ attack and defeat relations should be defined.

There are three ways in which arguments in ASPIC+ can conflict, and therefore three types of attack. There are attacks on conclusions (rebutting attacks), attacks on inference steps (undercutting attacks), and attacks on an ordinary premise (undermining attacks). Arguments cannot be attacked on strict inferences, conclusions of strict inferences and axioms. Attacks can be formally defined in the following way:

Definition 12 (attacks) *[16] A attacks B iff A undercuts, rebuts or undermines B, where:*

- *A undercuts argument B (on B') iff Conc(A) = -n(r) for some B' ∈ Sub(B)such that the top rule r of B' is defeasible.*
- *A rebuts argument B (on B') iff Conc(A) = -ϕ for some B' ∈ Sub(B) of the form B''_1 , . . . , $B''_n \Rightarrow \phi$.*
- *Argument A undermines B (on ϕ) iff Conc(A) = -ϕ for an ordinary premise ϕ of B.*

An attack relation gives information about which arguments are in conflict with each other. Defeat relations are used to specify which attack is successful.

In ASPIC+ the user needs to specify the relative strength of arguments. This can be done by providing a binary ordering $\preceq$ on the set of all arguments that can be constructed on the basis of an argumentation theory. Then if A $\preceq$ B and B $\not\preceq$ A then B is strictly preferred to A (denoted A $\prec$ B). Also, if A $\preceq$ B and B $\preceq$ A then A $\approx$ B.

Definition 13 (Successful rebuttal, undermining and defeat) *[16]*

- *A successfully rebuts B if A rebuts B on B' and $A \not\prec B'$.*
- *A successfully undermines B if A undermines B on ϕ and $A \not\prec \phi$.*
- *A defeats B iff A undercuts or successfully rebuts or successfully undermines B.*

When two arguments A and B are of equal strength, or when ordering is not defined, and both arguments rebut one another, both rebuts are successful. In other words, A defeats B and B defeats A.

2.3.4.2 Generating argumentation frameworks

We are now fully equipped to instantiate a Dung framework that incorporates ASPIC+ arguments and the ASPIC+ defeat relation.[16]

Definition 14 (structured argumentation framework) *[16] Let AT be an argumentation theory (AS, KB). A structured argumentation framework SAF defined by AT is a triple (A, C, $\preceq$) where*

- *In a SAF, $\mathcal{A}$ is the set of all arguments on the basis of KB in AS;*
- *$\preceq$ is a preference ordering on A;*
- *(X, Y) $\in \mathcal{C}$ iff X attacks Y.*

Definition 15 (Argumentation frameworks) *[16] An abstract argumentation framework (AF) corresponding to a SAF = (A, C, $\preceq$) is a pair (A, D) such that D is the defeat relation on A determined by (A, C, $\preceq$).*

Based on the framework specified in chapter 2.3.2, we can define how to determine which arguments are justified in grounded semantics. Below is a definition as specified by Prakken[16] that is applicable to all semantics.

Definition 16 *[The status of conclusions] [16] For grounded semantics and for every structured argumentation framework SAF with corresponding abstract argumentation framework AF, and every formula $\phi \in \mathcal{L}_{AT}$:*

1. *ϕ is justified in SAF if and only if there exists a justified argument on the basis of AF with conclusion ϕ;*
2. *ϕ is defensible in SAF if and only if ϕ is not justified in SAF and there exists a defensible argument on the basis of AF with conclusion ϕ;*
3. *ϕ is overruled in SAF if and only if it is not justified or defensible in SAF and there exists an overruled argument on the basis of AF with conclusion ϕ*

2.3.5 Application

No research has been done on the use of argumentation for chess XAI. For this application, the argumentation framework ASPIC+ seems like a suitable choice because ASPIC+ is a framework that can be used to give structure to argumentation. Structured argumentation about chess endgames should allow for to the point explanations with a logical structure. An implementation of the ASPIC+ framework for the use of chess XAI will be explored in this thesis. As a reference for the structured argumentation using the ASPIC+ framework, the article by Modgil et al. will be used [15].

Chapter 3

The implementation of an argumentation model

For the generation of explanations, a model with an innate understanding of the concepts that apply to chess positions is required, as these concepts are the cornerstones of a good explanation. Humans try to figure out whether a move is strong or whether there is a move that refutes it. This is similar to argumentation. Therefore, a model that uses argumentation should be able to provide a good base for generating explanations.

In addition, the model needs information about the board position and a set of rules that can be applied to the information harnessed from this position.

3.1 Language

Predicates will be denoted as words starting with an uppercase letter, with their relevant variables in brackets. The variable p expresses the position on the chessboard. The position has a lot of properties that depend on the intricacies in the position. For clarity the position p refers to will be displayed in a diagram, an example is shown in figure 3.1. The diagrams contain relevant information like which player is to move, the colour in the square of the diagram indicates which player has to move. For variables, it is common to use small letters. Constants will be denoted with capital letters. For example, the predicate $IsCapture(m)$ is used to represent that a property of move m is it being a capture.

Moves are indicated with the variable m. Using predicates like $IsKPK(p)$ helps to dedicate the model to specific endgames. This predicate indicates that the current position has access to the properties in king and pawn versus king endgames. Another such predicate is $IsPawn7th(p)$, which indicates the pawn is on the seventh rank. A pawn being on the seventh rank heavily impacts the options that have to be considered by both players.

For the scope of a KPK endgame, it is possible to make an exhaustive list of predicates required to solve it. The following predicates are required for KPK games with a pawn on the 7th rank:

- $IsPromotion(p)$: This predicate is true if promotion is available in position p,
- $IsPromoted(p)$: This predicate is true if the pawn in a KPK endgame

is promoted in position p, the position is technically no longer a KPK endgame, but this predicate can be used for reasoning about positions where the pawn can be moved to the 8th rank

- $IsCapture(p)$: This predicate is true if a capture is available in position p,
- $IsInvalidMaterial(p)$: This predicate is true if there is not enough material to end the game with a decisive result in position p,
- $IsDraw(p)$: This predicate is true if position p is always drawn if the defending player responds correctly regardless of what the attacking player plays,
- $IsStalemate(p)$: This predicate is true if the player to move has no available moves in position p (in KPK endgames only applies for the defending player),
- $IsWin(p)$: This predicate is true if position p leads to a decisive result for the attacking player, regardless of the response of the defender,
- $IsKPK(p)$: This predicate is true if position p has no pieces and at most one pawn,
- $IsFlankPawn(p)$: This predicate is true when the pawn in position p is a flank pawn (narrows the endgame down to a subsection of KPK endgames),
- $IsPromotionDefended(p)$: This predicate is true when the defending king has control of the promotion square in position p,

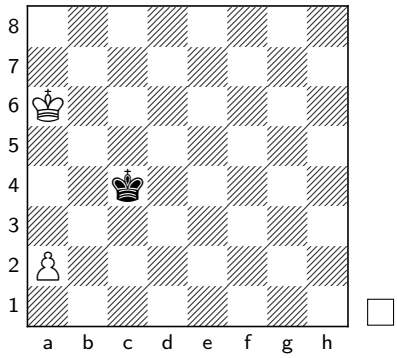


Figure 3.1: Visual representation of a specific position p on the chessboard

- $IsPromotionAttacked(p)$: This predicate is true when the attacking king has control of the promotion square in position p,
- $IsPawn7th(p)$: This predicate is true when the pawn is on the 7th rank in position p (narrows the endgame down to a subsection of KPK endgames),
- $IsBlocked(p)$: This predicate is true when the defending king is directly in front of the pawn in position p,
- $IsPawnDefended(p)$: This predicate true when the pawn is defended by the king in position p,
- $IsMove(p)$: This predicate is true when the attacking player has to move in position p,

The language will make use of the function symbol $MakeMove(m, p)$ to discuss transformations from the original position after making the move m.

The rules that apply to a chess position can be separated in three categories: rules that apply on the attacking player's turn, rules that apply on

the defending player's turn and rules that apply in general. For simplicity purposes, and because reasoning about a losing position from the defendants side provides no value, the used diagrams and rules are used from the attacking player's perspective. Moves are described with the constant M, with as subscript the played move, for example: $Capture(M_{kd6}) \rightarrow Draw$, here the model checks whether there is a capture available after kd6, the position is then evaluated as draw if it is. Checking whether a statement is accepted or rejected by the model is called a query. When executing a query, the statement will be validated using the grounded argument game to determine if there is an acceptable set of arguments for the statement. Having both rules with and without variables allows multiple kinds of queries. The query Draw will only check for general rules applying to the position, where a query like $Draw(MakeMove(M_{kd6}, p))$ will check whether the position is a draw after the move is played. To enable checking positions, in addition to the initial position, rules contain a variable in the output, for example $Stalemate(p) \rightarrow Draw(p)$. If position p occurs after a move from the original query, no conclusion will be drawn about the original position until all other variations are evaluated as well.

To enable conflicts in the model, contrariness should be defined. As long as only a win or a draw is possible for the current player, a draw is the contrary of a win.

Contrariness:

- $IsDraw(p) - IsWin(p)$

- $IsWin(p) - IsDraw(p)$

Note that this makes IsDraw(p) and IsWin(p) contradictory. Also note that only two results are considered in the subdomain of KPvK endgames. The position is always better for the player with the pawn, never resulting in a loss.

There are no standards yet for chess board representation in logical language. Chessboard positions will be represented as diagrams in this document. Whenever relevant, a position p will be accompanied by such a diagram.

3.2 Knowledge Base

The knowledge base $\mathcal{K}$ contains all relevant statements applicable to the chessboard position. This includes the truth values of every defined predicate, except for $IsDraw(p)$ and $IsWin(p)$.

3.3 Rules

The inference rules consist of two disjoint subsets, $\mathcal{R} = \langle \mathcal{R}_s, \mathcal{R}_d \rangle$ where $\mathcal{R}_s$ is a set of strict rules and $\mathcal{R}_d$ is a set of defeasible inference rules. Some rules will be included in $\mathcal{R}_n$, it is for example always a draw if the position is stalemate or if a capture leads to insufficient material. Some liberties have been taken by considering a king and queen versus a lone king as a win, as it is technically possible to blunder the queen or blunder stalemate, however,

explaining these positions adds low value to an explanation of the initial pawn endgame. Most rules belong in $\mathcal{R}_d$.

There are two types of conditions, conditions that apply on the current move and conditions that can be evaluated after a move is played. For the second type of condition, the played move is input as variable. This allows queries on specific moves, for example, $IsWin(MakeMove(M_{Kd4}, p))$ checks whether the move Kd4 leads to a position that is winning. The moves are always evaluated from the attacking player's perspective in the model, this enforces that moves like capturing can only be executed if it is known what move a rule is responding to. A model containing rules that expect to be evaluated on the defending player's turn could also be made, but these positions could also be approached using the attacking player's model after one move is played.

To create a set of rules, a vast amount of chess knowledge is required. There are several places to acquire this knowledge. This can either be gained from an expert opinion, e.g. Kd4 is the best move in the position according to a chess grandmaster because this leads to the most issues for the opponent. Or it can be gained from an analogy, for example: Kd4 is the best move in the position because in a specific similar position Kd4 is also the best move. Or it can be gained from generalisations, as found in for example a chess theory book. For example: Kd4 is the best move because it follows the general rule for king pawn endgames to keep the attacking king in front of the pawn. Expert opinion is a good source when discussing equal positions, as grandmasters are good at making practical choices that limit the options of their opponents. It is, however, not reasonable to get an expert opinion about ev-

ery position, and in most positions not required. Analogies may be suitable for the interpretation of chess programs if the positions that neural networks use to base their decisions on are known, but as these programs work currently this is not the case. The best fitting solution is using generalisations. Chess is a game with a lot of knowledge and expertise built over the years. The many books and articles written about chess endgames where grandmasters have shared their knowledge and expertise could be a valuable resource to analyse and understand chess endgames. What these books not provide, however, is a consistent list of the rules that can be applied. It is therefore not simple to come to a minimal list of generalisations that can be used to answer how to approach all king and pawn endgames. Another source that may be more fitting is the material written by chess-programmers, programmers building chess engines. Most programs use tablebases like the Syzygy tablebase [3] (solved databases of all chess endgames with 7 pieces or fewer) to make decisions about chess endgames. But the chess programming wiki [4] contains some general rules that can be used as a base to approach king and pawn endgames.

3.4 7th rank

Instead of trying to address all king pawn endgames immediately, it is simpler to isolate a subsection. When the pawn is one square away from promotion, the options are limited. Firstly, it is clear that once the pawn promotes, the endgame is won. If the pawn is captured, the endgame results in a draw. Pawn capture takes priority over promotion of the pawn in evaluating

the position, as there would be no material to checkmate. If the attacking king defends the promotion square, the pawn can always be safely promoted without being captured. This information is almost enough to address all endgames where the pawn is on the 7th rank. There are, however, some exceptions that need to be addressed. Foremost, the endgame is a draw if the pawn is a flank pawn and the defending king controls the promotion square. Furthermore, stalemates on the current move, and on the next move, need to be handled by the model. Then there is one case left where the defending king controls the promotion square, this endgame is winning for the attacker if the pawn can be defended with the next move without this resulting in a stalemate.

3.4.1 Rules for 7th rank pawns

The rules listed below cover all the 7th rank pawn cases described earlier. They should therefore be able to give a complete answer to queries about king and pawn endgames with pawns on the 7th rank, as long as all relevant information is extracted from the position and used in the knowledge base of the model or as query input.

Rules:

- $[r_*]IsMove(p), IsWin(MakeMove(m,p)) \rightarrow IsWin(p)$ If a winning move can be made in a position, then the original position is winning.
- $[r_0]IsMove(p), IsPromotion(MakeMove(m,p)) \Rightarrow IsWin(MakeMove(m,p))$ Making a move that promotes a pawn in the position is generally win-

ning.

- $[r_1]\neg IsMove(p), IsCapture(p) \rightarrow IsDraw(p)$

  If it is the turn of the opponent and the pawn can be captured, the position is a draw.

- $[r_2]IsMove(p), IsCapture(MakeMove(m,p)) \rightarrow IsDraw((MakeMove(m,p))$

  If the attacking player makes a move that allows capture by the opponent, the move leads to a position that is drawn.

- $[r_3]IsFlankPawn(p), IsPromotionDefended(p) \rightarrow IsDraw(p)$

  When the defending king has control of the promotion square of a flank pawn, the position is drawn because it is impossible to force the defending king away. Therefore, promotion of the pawn is prevented.

- $[r_4]IsStalemate(MakeMove(m,p)) \rightarrow IsDraw(MakeMove(m,p))$

  The move m leads to stalemate, therefore the move leads to a draw.

- $[r_5]IsStalemate(p) \rightarrow IsDraw(p)$

  There are no moves for the defending player left in the position, this means the position is drawn.

- $[r_6]IsPawn7th(p) \wedge IsPromotionDefended(p) \Rightarrow IsDraw(p)$

  If the defending king has control of the promotion square and the pawn is on the seventh rank, promotion of the pawn is usually not possible, making the position a draw.

- $[r_7]IsPromotionAttacked(p) \wedge IsPawn7th(p) \rightarrow IsWin(p)$

  If the attacking king has control of the promotion square and the pawn

is one move from promoting it is impossible to stop promotion, therefore the position is winning for the attacker.

- $[r_8]IsMove(p) \wedge \neg IsPromotionDefended(p) \wedge IsPawn7th(p) \rightarrow IsWin(p)$
  If the promotion square is undefended, the pawn cannot be captured after promotion. In this position, promoting the pawn is winning for the attacker.

- $[r_9]\neg IsMove(p) \wedge IsPawn7th(p) \wedge IsPawnDefended(p) \Rightarrow IsWin(p)$
  When a pawn on the seventh rank is defended and the defender has to move (with the attacker to move the pawn tends to become undefended or stalemate becomes a risk) the position is generally winning.

- $[r_{10}]IsMove(p) \wedge IsPawn7th(p) \wedge IsPawnDefended(MakeMove(m, p)) \Rightarrow IsWin(p)$
  Defending the pawn on the seventh rank is generally a good idea and leads to a win as long as the defender has no stalemate tricks.

- $[r_{11}]IsPromotionDefended(p) \wedge \neg IsBlocked(p) \rightarrow \neg r_{10} \wedge \neg r_9$
  If the defending king can move to the promotion square, simply defending the pawn is not enough to win because the defending side can enforce a stalemate situation.

3.4.2 Strictness of rules

Some rules are considered strict, while others are not. The reason for rules being strict is that they cannot be refuted when the conditions are present. A good example is the first two rules. Those rules state that the game is

drawn upon the capture of the pawn. The game being drawn is in this case enforced by the game rules, as there is not enough material to win the game.

Rule r_3 is strict because, while technically it is possible for the defender to lose by cooperating with the opponent, it is physically impossible for the pawn to reach the promotion square.

Rules r_4 and r_5 are again a matter of applying the game rules, a stalemate is drawn on the spot.

Rule r_6 as can be seen in figures 3.2(a) and 3.5(a) is susceptible to attacks. It can be attacked by rules r_7 and r_{10}.

Rule r_7 always results in a win for the attacking player because promotion is imminent. It is important to note that it is not possible to create a situation on the board where the previously mentioned strict rules resulting in a draw are also applicable.

Rule r_8 similarly to r_7 is strict because promotion is unavoidable by the defending player.

Rule r_{10} is defeasible. The rule shows that the game is won if the pawn can be defended with the next move. There is, however, one exception, which is

represented with r_{11}.

Rule r_{11} is strict. There are situations where this rule is applicable, and the game is a win, but all this rule specifies is that the game is not won in accordance to r_{10} even when another rule shows a win.

3.4.3 Rule preferences

When a conflict between arguments occurs it should prioritize more important rules. ASPIC+ allows the user to specify this with rule preferences. With the way the set of rules is constructed, it is always the case that an argument that is both strict and firm is preferred over all other arguments. A firm argument is an argument that is constructed without ordinary premises, and there are no ordinary premises in the model, therefore all arguments in the current representation of the model are firm. Individual rule preferences could be useful in most types of endgames. If, for example, both players had a pawn close to promotion, then rule preferences could be used to give priority to rules that result in quicker promotion. In the context of KPvK endgames, there are no such pawn races and only one player is playing for a win. Because the endgames are one-sided the individual preferences between the rules do not require specification. In summary: Rules either support that an endgame is winning or that it is drawn, and strict-and-firm rules have a preference over all other rules.

3.5 Explanation architecture

As stated earlier, the G-Game will be used to determine whether arguments are in or out. Because it is clear that the attacking side cannot lose in KPvK endgames can be divided into two situations:

1. The position is winning.
2. The position is drawn.

The model is used solely for explanation of the position and not for solving it, therefore there is no problem in using the result of the position to simplify the model. Having this information, the winning strategy for the proponent in a winning position results in a tree where all the branches end in a win. The winning strategy in a drawn position argues for a draw instead.

In figure 3.2(a) a position is given where white to play is winning. There are a couple of notable aspects to the position. The attacking player has the move, the pawn can be promoted with the move b8=Q, and the promotion square is both attacked and defended.

Example 3.5.1 *The proponent wants to show that IsWin($P_{3.2a}$) is in the grounded extension. The following knowledge can be elicited from the position:*

$\mathcal{K}_n : \{IsKPK(P_{3.2a}), IsPawn7th(P_{3.2a}), IsMove(P_{3.2a}),$
$IsPromotion(MakeMove(M_{b8=Q}, P_{3.2a}), IsPromotionDefended(P_{3.2a}),$
$IsPromotionAttacked(P_{3.2a})\}$

Using the knowledge, three arguments can be constructed:

(A)

- *A1:*$IsPawn7th(P_{3.2a})$
- *A2:*$IsMove(P_{3.2a})$
- *A3:*$IsPromotion(MakeMove(M_{b8=Q}, P_{3.2a}))$
- *A4:* $A1, A2, A3 \Rightarrow IsWin(P_{3.2a})$ *(applying* r_0*)*

The move b8=Q playable in the position in the diagram promotes the pawn. Making a move that promotes a pawn in the position is generally winning.

(B)

- *B1:*$IsPawn7th(P_{3.2a})$
- *B2:*$IsPromotionDefended(P_{3.2a})$
- *B3:* $B1, B2 \Rightarrow IsDraw(P_{3.2a})$ *(applying* r_6*)*

The defending king has control of the promotion square with the pawn on the 7th rank, this generally results in a draw because the pawn is unable to be promoted.

(C)

- *C1:*$IsPawn7th(P_{3.2a})$
- *C2:*$IsPromotionAttacked(P_{3.2a})$
- *C3:* $C1, C2 \rightarrow IsWin(P_{3.2a})$ *(applying* r_7*)*

The attacking king has control of the promotion square and the pawn is one move from promoting making it impossible to stop promotion, therefore the position is winning for the attacker.

Of these arguments, arguments A4 and B3 support that the position defeasibly is a win or a draw respectively. This can be concluded because the arguments

respectively are formed using rules r_0 and r_7. Because priorities are not specified, A4 and B3 successfully rebut each other. Therefore, it can be concluded that A and B defeat each other. Argument C3 provides a strict (and firm) argument for the endgame being a win because it is based on rule r_7. C3 being strict and firm gives it preference over B3. Argument C3 rebuts B3 and therefore defeats it.

A G-Dispute, with the proponent trying to show that IsWin($P_{3.2a}$) is admissible, can be argued in the following two ways:

- *P_1: A4, O_1: B3, P_2: C3,*
- *P_1: C3.*

The proponent has the winning strategy of responding to argument B3 with argument C3 (Or starting with C3, leaving no options for the opponent). This approach is valid against whatever the opponent does.

The example discussed above can be mapped to a natural language explanation. Either strategy, resulting in C3 labelled in or {A4,C3} in and B3 out, can be chosen. The natural text for an explanation using only argument C3 would be based on the description of rule r_7: "The attacking king has control of the promotion square and the pawn is one move from promoting. Promotion is impossible to stop making the position a win for the attacker."

3.6 Checking all 7th rank positions

Before expanding the model, it is essential to confirm that all positions with the pawn on the 7th rank are evaluated correctly. In this section, the different 7th rank positions will be defined. The number of positions that can occur with a pawn on the 7th rank is huge, but the positions can be greatly reduced as most are similar enough to be eliminated. First, the total positions can be halved because their mirrored alternative will be explored. This results in only having to check positions with the pawn on the a, b, c and d file.

Rules r_4-r_{10} are sufficient for all positions with pawns on the seventh rank. The first position to evaluate has all pieces in the upper left of the board, with the pawn on a7 and the kings on a6 and a8. Then the black king is moved to the right until all positions have been explored. Then the same process is repeated with the white king moved one square. Once all has been explored the same process will be repeated for a-, b-, c-, and d-pawn.

For the a pawn, figure 3.5(b) is a good example. The white king does not defend the promotion square and promotion is not even an option because the black king blocks the pawn. The moves Ka5 and Kb5 lead to a draw according to r_2. The only available move for white is Kb6, but applying r_5 shows the position is a stalemate after the move and therefore a draw. All other positions where the white king and pawn are on the same squares lead to a win for white because promotion is not defended as described in r_8. Promoting the pawn leads to a win in these positions, and therefore other white moves do not need to be considered. The only other position where promotion is not a win for white is when the pawn can be captured because

the black king is on b7, this can also be concluded from r_6. Promotion is defended, therefore promoting leads to capture of the promoted pawn. All the positions with the black king on b7 and the white pawn on a7 should be evaluated as a draw. This is easy to confirm with the model. Put the white king on any square and after promotion the model shows that the game is drawn after a capture using r_2. After any king move Kx the model also shows a draw after a capture using r_2. All other a-pawn positions have the defending king not in position to capture the pawn, and therefore lead to a win by applying r_8. The b-, c-, and d-pawn are similar to one-another with the only difference being how close the pieces are to the edge of the board. First, the b pawn should be addressed. With the white king on a6 and the defending king on b8, the position should lead to a draw. If the king moves away from the pawn, r_2 immediately leads to a draw. The only move that is still available because it does not lose the pawn is Kb6, but then r_4 can be applied, therefore this position is a draw. With the white king on c6, a similar reasoning applies. With the white king starting on b6 the pawn can be defended without leading to stalemate, the model should show a win here. Again, any move that loses the pawn is shown as a draw. This leaves Ka6 and Kc6. In both positions r_{10} applies as the moves defend the pawn. The defending king has to make a move which results in a position where r_1 applies. The c and d pawn can be addressed in the same way. All other positions with the white or black king on any other square can be resolved in either a win if the pawn can be promoted or a draw if the pawn can be captured either immediately or after moving.

In figure 3.2(a) the application of rules r_6 and r_7 is shown. The pawn is on the seventh rank and the promotion square is defended by the black king, r_6 concludes that the position should therefore be a draw. This, however, is attacked by both r_7 and r_{10}. r_7 shows that the endgame is won if the promotion square is attacked when the pawn is on the seventh rank. Because r_7 is strict it defeats r_6. Therefore, the conclusion is that the endgame is won.

In figure 3.2(b) a position is given where white to play is winning. The notable aspects of the position are that the pawn is on the 7th rank and can promote, and that the promotion square is not defended.

The proponent wants to show that IsWin($P_{3.2b}$) is in the grounded extension. Using the information listed, a defeasible argument (A) that the position is winning can be created with the help of r_0. A strict argument (B) can be created for the position being winning using r_8. The opponent cannot defeat either argument, so starting with either is a winning strategy. This leads to either of the following explanations:

r_0 The pawn can simply be promoted, leading to a win for white.

r_8 The promotion square is undefended and the pawn cannot be captured after promoting. In this position, promoting the pawn is winning for the attacker.

In figure 3.3(a) The proponent wants to show that IsDraw($P_{3.3a}$) is in the grounded extension. This can be shown using [r_{12}], all moves lead to a draw, therefore the position is a draw. Using this rule in an argument, the

proponent has a winning strategy for the argument game. This does require that an argument for any move being a draw can be constructed. This is the case because the pawn can be captured after both Ka5 and Kb5 (a simple argument that can be constructed using $[r_1]$). The opponent has no arguments here. And the only move left is Kb6, leading to stalemate using r_4. Here the opponent has no arguments either. This leads to the explanation, all moves are a draw, $\{Ka5, Kb5\}$ lead to capture of the pawn, and Kb6 is stalemate.

In figure 3.3(b) the situation is similar to figure 3.3(a). The proponent has to argue that the position is drawn. For all moves $[r_{12}]$ rule r_6 applies, and the pawn can in no way be defended. r_2 simply shows that in all resulting positions, the pawn can be captured. r_3 also shows that the position is a drawn flank pawn position. Any of these rules can be used to construct an explanation because the opponent has no counterarguments that would show that a move would be winning.

In figure 3.4(a) the same reasoning applies.

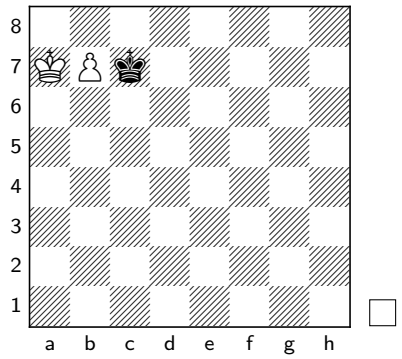
In figure 3.4(b) the situation becomes interesting. Here the proponent has to argue for a draw. All moves except Ka6 can be rejected using the same reasoning as in the previous example. In the resulting position, argument A can be generated that defeasibly claims that the position is a draw due to the defence of the promotion square using $[r_6]$. The opponent has counterargument B that can be constructed using $[r_{10}]$, the pawn is defended after Ka6, therefore the position should be winning. Argument B is, however, defeated by argument C constructed using r_{11}. The rule r_{10} is not applicable to the

position because the pawn is not blocked in the original position, enabling the defending king to go in front of the pawn and force a stalemate situation. The opponent has run out of moves in the g-game, therefore the opponent's claim on the position being a draw is in the grounded extension.

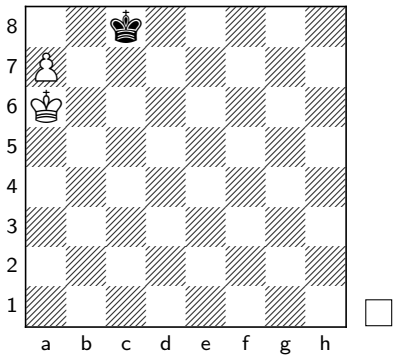
In figure 3.5(a) The same reasoning applies as to the previous example, but here the proponent is arguing for the winning side. The move Ka6 leaves the opponent only with the move Kc7, this does not block the pawn. This leads to a situation where the pawn cannot be blocked, therefore an argument A using r_{10} cannot be defeated using an argument that is built upon r_{11} leaving the opponent with no legal moves in the g-game after argument A.

In figure 3.5(b) the position is simply stalemate again.

In figure 3.6 the position is similar to figure 3.5(a).



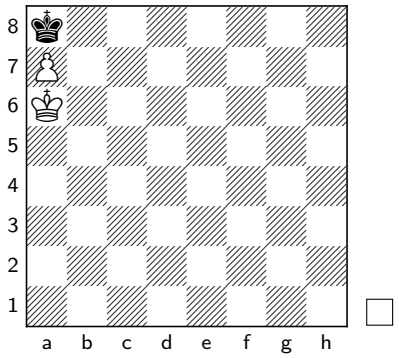
(a) Example r_6-r_7



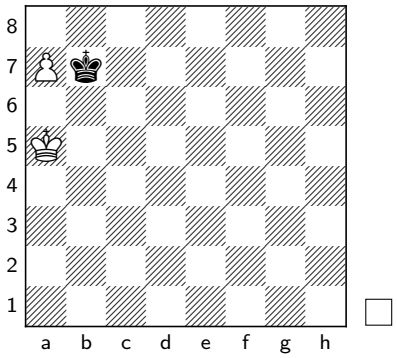
(b) Example r_8

Figure 3.2: Black cannot stop promotion of the pawn

Positions with the pawn on a different rank can be calculated down to positions with the pawn on the 7th rank, or be approached using the location of the pawn relative to the kings.

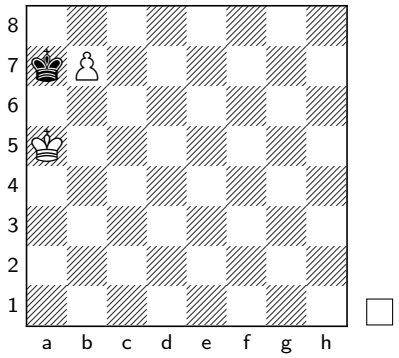


(a) Stalemate or pawn loss

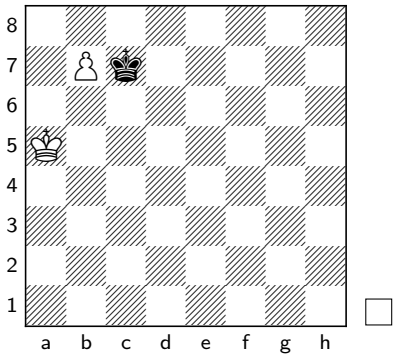


(b) The pawn is lost

Figure 3.3: White cannot promote the pawn



(a) Pawn is lost again

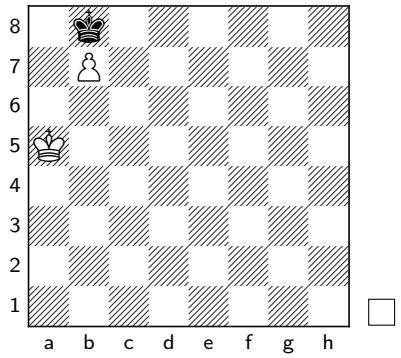


(b) Pawn can be defended, still a draw

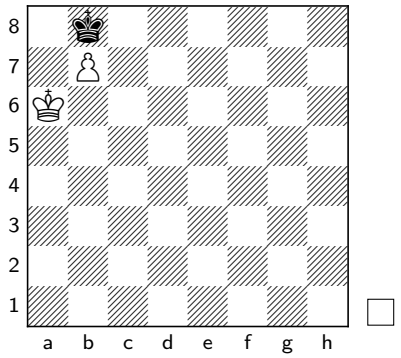
Figure 3.4: White cannot promote the pawn 2

3.7 Expanding the model to all king and pawn endgames

As explored in the previous section, positions with a pawn on the seventh rank are relatively simple to generalize. Therefore, all positions can be covered. Once complexity increases by adding material to the endgame or expanding the endgame to the entire board, it becomes increasingly more dif-



(a) After Ka6 black is in zugzwang



(b) Stalemate again

Figure 3.5: Stalemate situations

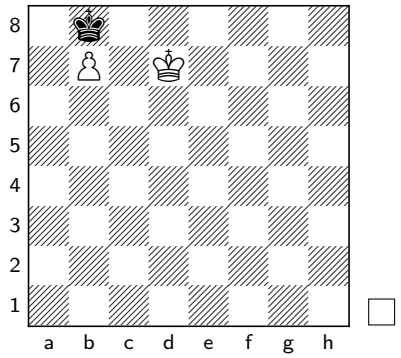


Figure 3.6: After Kc6, black is in zugzwang

ficult to show all positions can be covered. For human-like explanations, it is of importance that this complexity is not expressed in a way that chess engines address this, with deep calculation with long sequences of moves. It is therefore of importance that the model makes use of concepts instead of calculation. Of course, calculation cannot entirely be avoided in the game of chess, but all king and pawn endgames can be approached with shortcuts that remove the need of calculating them until the end.The chess programming wiki describes a set of heuristics that can achieve such a goal [4]. This is why

those shortcuts are used to explain these positions, as this allows humans to spend their thinking time elsewhere on the games. The model should take this into account.

3.7.1 Concepts applicable to KPK endgames

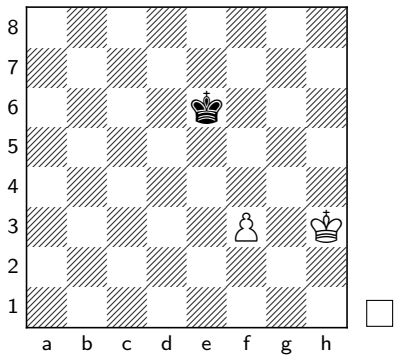
One of the factors that has impact on the complexity of the position is "zugzwang" which is German for "forced to move". If the defending player could pass every turn, a draw could simply be reached by having the defending king somewhere in front of the pawn. Due to the nature of the game, it is, however, possible to force the defending king away using zugzwang. To enable this, chess players make use of the concept of opposition. In Dvoretsky's endgame manual [9] opposition is defined as "the state of two kings standing on the same file with one square separating them." Opposition may be vertical, horizontal, or diagonal. To see how opposition can be applied to KPvK endgames, a set of endgames has been explored. A free online chess endgame trainer [1] is used as reference to elicit common patterns from. In the 45 KPvK endgames on the website, most fall under one of the three following patterns (after a move of the attacking side):

- 1. The kings are directly opposite of one another (on any file) and the attacking king is one rank in front of the pawn. Shown in figure 3.7(b). White wins because the player can keep putting the opponent in opposition if black tries to approach the pawn.

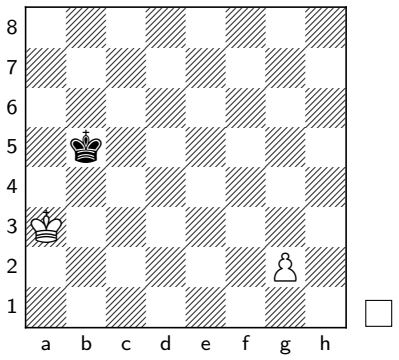
- 2. The kings are diagonally opposite of one another, and the defending

king is not on a file closer to the pawn than the attacking king. This is shown in figure 3.7(a). The white king either advances to g5 or opposes the black king after 1.Kg4 Kf6 2 Kf4.

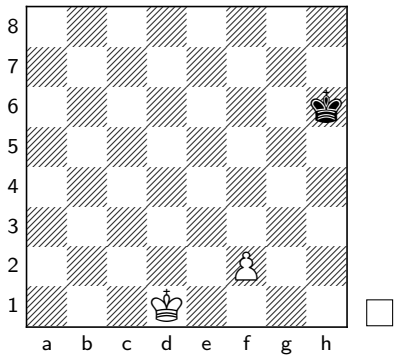
- 3. The defending king can move directly opposite of the attacking king, and the attacking king is one rank in front of the pawn. This is essentially the same as the attacking player not being able to give opposition.



(a) Diagonal opposition after Kg4



(b) Opposition after Kb3



(c) Diagonal position after 1.Ke2 Kg5 2.Ke3

Figure 3.7: Opposition plays a role in these endgames

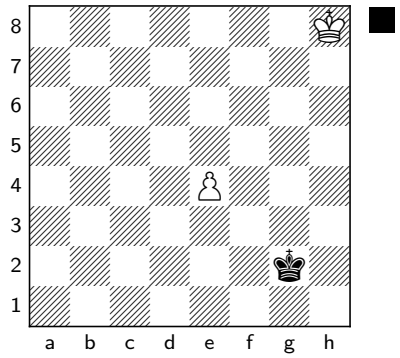


Figure 3.8: Blacks king can not move into the square, white wins

With opposition defined, it is now possible to make progress in all kinds of KPvK endgames.

Another important concept is what chess players know as the square rule. If the defending king is outside the area to catch the pawn before promotion, the endgame is won by simply pushing the pawn up the board. This concept is usually explained with an imaginary square extending from the pawn to the promotion-square, including the diagonal on the side of the defending king. This is illustrated in figure 3.8.

Another concept that applies to KPvK endgames is key squares. As described by Dvoretsky [9] key squares are squares whose occupation by the king assures victory. This is regardless of whose turn it is. Whenever a key square is discussed, there is a clear reason for it leading to a win. For simplicity' sake, this reason will be seen as a sufficient explanation. After reaching a key square, no further moves in the position require explanation. In some exceptional cases, e.g. the pawn being a flank pawn, the key square will not lead to a win. In KPvK endgames, the key squares can be described as the squares from which the king controls the first three squares in front of the

pawn. This allows the pawn to advance up the board while keeping a pawn move in reserve to create an opposition situation when necessary.

The following predicates are added:

- IsOpposition(p): This predicate is true if the kings are in opposition in position p,
- IsDiagonalOpposition(p): This predicate is true if the kings are in diagonal opposition in position p,
- IsInFront(p): This predicate is true if the attacking king is at least one row further up the board than the pawn in position p,
- IsInSquare(p): This predicate is true if the defending king is in the square of the pawn in position p,
- IsCloserKeySquare(p): This predicate is true if the attacking king can get to a key square faster than the defending king can defend it in position p,
- IsSameDistanceKeySquare(p): This predicate is true if the kings have the same distance to the key squares,
- IsFurtherKeySquare(p): This predicate is true if the attacking king is further away from the key square than the defending king in position p,
- IsCloserFlankPawnKeySquare(p): This predicate indicates whether the attacking king is closer to the key squares of the flank pawn in position

p. With a flank pawn, the key squares are the squares that defend the promotion square and are not in front of the pawn. For an a pawn, this would be b7 and b8.

3.7.2 Move sequences

There is another change compared to the endgames, where the pawn is on the 7th rank. Some positions require a sequence of multiple moves to reach a position that can be evaluated. The MakeMove function only evaluates one move at a time and allows the input of any move. If the move is bad, it will be shown by the result of the model. When inputting a sequence of moves into a similar function, it makes no sense to build sequences of all potential moves. This would evaluate situations where both players made suboptimal moves and not provide any useful information. Instead, only sequences with the best reply should be evaluated. When the position is considered a win, then the best move of the defending side should be considered once a suboptimal move leading to a draw has been played. When the position is considered a draw, the attacking side should play the best move. It is required to know the evaluation of the position. The evaluation can be acquired by checking the position with the syzygy table base [3]. This is an endgame tablebase containing evaluations of all endgame positions with 7 pieces or fewer. Using this evaluation, move sequences can be built in the following way:

- 1. A random move is selected for the attacking player
- 2. The resulting position is checked with the model, if it conforms to a

pattern specified in the rules and leads to a conclusion then stop

- 3. Else, the resulting position is checked with the syzygy database
- 4. If the resulting position evaluation is changed, the move is discarded and the next alternative is tried
- 5. If the evaluation stays the same the move is correct, then the best defensive move is selected (the move that makes the game last the longest) in case of a winning position. With a drawn position, any of the drawing moves is selected and added to the sequence. Evaluate the position like in step 2.
- 6. Go to step 1

This algorithm is represented as the function MakeMoves(m,p). Expanding on the set of rules with seventh rank pawns, all the previously listed predicates can be used to construct the following rules:

- $[r_{13}] IsCloserKeySquare(p) \Rightarrow IsWin(p)$; If the king is closer to a key square, the endgame is usually winning because the pawn can promote.
- $[r_{14}] IsMove(p) \wedge IsOpposition(MakeMove(m,p)) \wedge IsInFront(MakeMove(m,p)) \Rightarrow IsWin(p)$; Opposition can be converted to the king being on a key square if the attacking king is in front of the pawn.
- $[r_{15}] IsMove(p) \wedge IsDiagonalOpposition(MakeMove(m,p)) \wedge IsInFront(MakeMove(m,p)$ $IsCloserKeySquare(p)) \Rightarrow IsWin(p)$; Diagonal-opposition can be converted to normal opposition if the king is closer to the pawn and is therefore winning.

- $[r_{16}]\neg IsInSquare(p) \rightarrow IsWin(p)$; If the defending king is not in the square, it cannot catch the pawn. Therefore, pushing the pawn leads to a win.

- $[r_{17}]IsFlankPawn \wedge \neg IsCloserFlankPawnKeySquare(p) \Rightarrow IsDraw(p)$; If the defending king can get in front of the defending pawn, then promotion is not possible and the game is a draw.

- $[r_{18}]IsFlankPawn \wedge IsCloserFlankPawnKeySquare(p) \Rightarrow IsWin(p)$; If the attacking king can control the key squares of a flank pawn, then the pawn can promote, leading to a win.

- $[r_{19}]IsMove(p) \wedge IsDiagonalOpposition(MakeMoves(m,p)) \wedge IsInFront(MakeMoves(m,$ $IsCloserKeySquare(p)) \Rightarrow IsWin(p)$; There is a sequence of moves where the defender has no better option than to let the king get to a key square eventually, though diagonal opposition, the position is winning.

- $[r_{20}]IsCloserKeySquare(MakeMoves(m,p)) \Rightarrow IsWin(p)$; There is a sequence of moves where the defender has no better option than to let the king get to a key square eventually, the position is winning.

- $[r_{21}]\neg IsMove(p) \wedge IsOpposition(MakeMove(m,p)) \Rightarrow IsDraw(p)$; The opponent has opposition, making the game a draw.

- $[r_{22}]\neg IsMove(p) IsSameDistanceKeySquare(p) \Rightarrow IsDraw(p)$; The opponent can defend the key squares, and the game is therefore a draw.

- $[r_{23}]IsFurtherKeySquare(p) \Rightarrow IsDraw(p)$; The opponent can defend the key squares, and the game is therefore a draw.

3.7.3 Strictness of rules

The set of rules applying to pawns on the seventh rank contained a good number of strict rules. Releasing the restriction of the seventh rank results in more defeasible rules.

Rule r_{13}, describing whether the attacking king is closer to any of the key squares than the defending king, is defeasible. The most common exception to this rule is flank pawn positions.

Rule r_{14} shows that an endgame is winning when the attacker can take opposition after the current move while the attacker's king is in front of the pawn. Again, flank pawn positions are the most common exception to this rule.

Rule r_{15} is similar to r_{14}, except that it discusses diagonal opposition. Again, flank pawns are the most common exception.

Rule r_{16} shows whether the defending king is capable of catching the pawn if it is simply pushed for promotion. If the king cannot catch the pawn, it is always a win for the attacker. This rule is therefore strict.

Rules r_{17} and r_{18} discuss flank pawns and their key squares. These rules are defeasible because there are for example cases when the attacker has to

balance reaching a key square with preventing the capture of the pawn.

Rules r_{19} and r_{20} are applications of earlier rules applied after a sequence of moves. These rules are still defeasible.

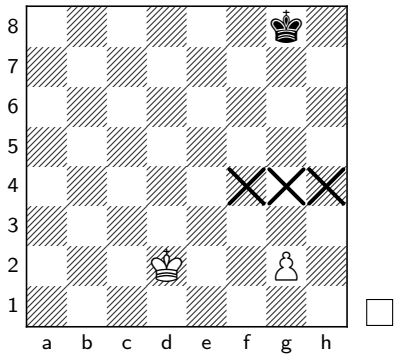


Figure 3.9: Simple king pawn endgame

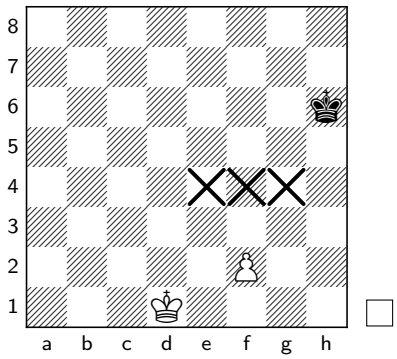


Figure 3.10: White can still get to a key square

3.7.4 Endgame trainer positions

The endgame trainer app [1] has 45 KPvK endgames. These are a diverse set of positions used to train the specific endgame. If these endgames can be solved and explained using the rules, then that would show that this set of

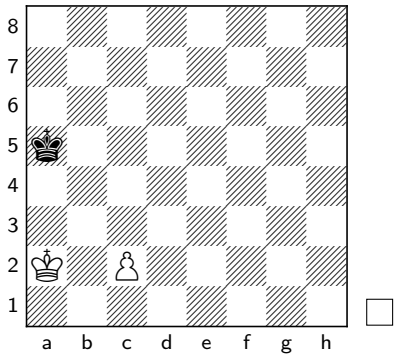


Figure 3.11: Application of r_{15}, Ka3 is winning

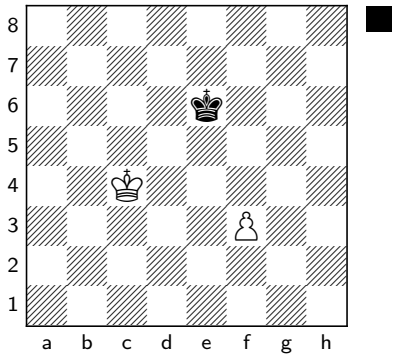


Figure 3.12: Diagonal opposition does not help

instructive endgames is covered. In the section 3.7.5 an attempt is made to prove the rules are sufficient for all KPvK endgames. Some endgames listed on the website have a lot of similarities, therefore only a subset is discussed here. Of the endgames listed on the website, some are selected to demonstrate the application of the rules, starting at 3.14(a). Under the images, a small description is added of why the position is winning.

Figure 3.14(a) is an example where r_{13} suggests that the key square can be reached by the white king because the predicate IsCloserKeySquare(p) ap-

plies to the position in the diagram. There are no conflicts because the pawn is not a flank pawn.

In figure 3.14(b) moving straight up to the e5 key square is too slow. The rule r_{13} does not apply to the key square g5, as both kings have the same distance. With r_{19}, it can be shown that this position is winning. The predicate IsMove(p) is applicable because it is the white player's turn. IsDiagonalOpposition is also applicable in the position after a move sequence generated with MakeMoves(m, p). The sequence m = 1.Kf2 Kd7 2.Kg3 Ke6 is generated using the MakeMoves function in the following way. 1. A random move is selected for the attacking player. For example Ke2. 2. The position after Ke2 does not directly correspond to one of the rules. 3. In the third step, the position after Ke2 is evaluated as a draw using the syzygy database. 4. The position before the move is evaluated as winning, therefore the move Ke2 is not correct. 1. All moves except for Kf2 reach the same conclusion. (Most moves conform to r_{22} and can be directly rejected in step 2) So now we consider the move Kf2. 2. The resulting position does not appear in the model (after any move except Kd7 and Kd8 it can be explained using r_{13} as the white king has a shorter distance to the g5 key square) 3. The syzygy database evaluates the position after Kf2 as winning 5. Both Kd7 and Kd8 could be selected as the best defensive move, as they both have a DTM (Depth to checkmate in half-moves) of 41. For this example, we consider Kd7. 6. Go to step 1 1. First, Ke3 should be considered 2. After Ke3 the position corresponds to r_{21}, after the move Ke7 black's king is always in a

position to get opposition. 1.Kf3 is the only reasonable move left (all other moves can be rejected using r_{22}) 2. The position after Kf3 does not correspond to any of the rules 3. Using syzygy, the position is again evaluated as winning 5. Both Ke7 and Ke6 have a DTM of 39. Ke7 is randomly selected. The resulting position corresponds to r_{15} using pseudo diagonal opposition. The move Kf4 leads to a win.

by and is the only sequence that directly prevents the white king from walking to the key square. Here, however, white can play 3.Kg4 gaining the diagonal opposition.

The diagram in figure 3.15(a) again has the white king in position to reach a key square first. Rule r_{13} is sufficient to explain this position. The white king is not only closer but on a key square, therefore the predicate IsCloserKeySquare is applicable to the position.

The diagram in figure 3.15(b) shows the same story. Again, the white king is not only closer but on a key square, therefore the predicate IsCloserKeySquare is applicable to the position. There is another option of playing c3 first, confirmed when applying the predicate IsOpposition with the MakeMove function for the move c3. The position has all predicates in relation to r_{14} present. This begs the question: "Is it sufficient to explain one solution if there are multiple solutions?". In this case, the answer to that question would be no, as both approaches are interesting and useful for a chess player to understand. Later in this section, an example of the opposite

will be given.

In figure 3.16(a) an example of the application of r_{14} is shown. After the attacking player plays the move Kb3 the predicate IsOpposition is applicable to the position. In the resulting position, the white king is a row in front of the pawn which is represented with the predicate IsInFront. Note that the defending king is in range to capture the pawn; although this does not play a role in evaluating the position as a win this information may be of interest to the player. Another important point to notice is that technically rule r_{20} is applicable to the position, with best play a sequence of moves will be played resulting in a position where the white king is closer to a key square. This sequence does, however, make use of the concepts in r_{14}. Using r_{20} as a building block for an explanation leaves out relevant information that is available, therefore it would in this case be sufficient to explain only a solution to the position using r_{14}.

In figure 3.16(b) one can see that the rules specified earlier are still required as building blocks for further endgames. Rule r_2 shows that the move h3 is forced. The move h3 is the only move that ensures that the predicate IsCapture is not applicable to the resulting position. After all other moves a capture leads to a draw. Even after the move h3 the white king is closer to the flank pawn key square g7. Rule r_{18} can therefore be used to show that this position is winning for white: this is simply because the predicates IsCloserFlankPawnKeySquare and IsFlankPawn are applicable in the original

position. This example shows that rule r_{18} does work even in this unusual position. The detail that h3 needs to be included may, however, be a relevant detail in a human-like explanation.

The diagram in figure 3.17(a) has the king on a key square. The predicate IsCloserKeySquare is therefore relevant to the position, resulting in the conclusion that the position is a win using r_{13}. This position will therefore be explained in the same way as every other position with a king on the key square.

The diagram in figure 3.17(b) can be explained in a similar fashion as figure 3.16(a).

The diagram in figure 3.18(a) cannot be explained with rules r_1 to r_{19}. Rule r_{20} shows that there is a sequence of moves that gets the king to a key square. The rule does, however, not capture the details of the position required for a good explanation. Rule r_{20} is a catch all that ensures an answer can be found, this answer should fit in a context that can be explained using other rules. In the domain of KPK endgames it is possible to add intermediate rules to expand on the details given in an explanation. Creating rules for a specific situation is easy, the difficulty lies in identifying details of a position not as artificial patterns that make sense for only one or very few positions. When scaling up to more complex endgames, this problem becomes more visible as the number of rules that should be added increases rapidly and so

do the nuances in these endgames. Therefore the choice is made to - instead of approaching a position with uncommon artificial rules - abstract these patterns as forced move sequences ending in a position where the other rules (r_1 to r_{19}) are applicable.

The diagram in figure 3.18(b) shows that r_{14} can also be applied one row further up the board. After the move Kd4 the predicate IsOpposition is applicable to the position, resulting in a winning position using r_{14}.

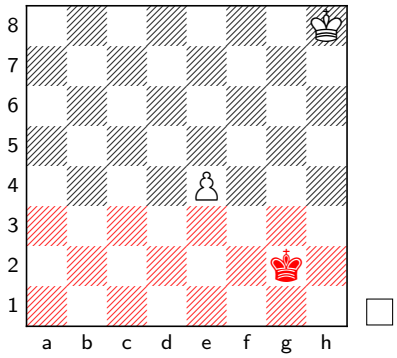
The diagram in figure 3.19(a) shows that the king once again is closer to a key square. Rule r_{13} is a simple rule, yet it is powerful enough to explain most KPvK endings.

The diagram in figure 3.19(b) is a funny example where the king is on a key square but will not be after moving. This position is still winning showing that r_{13} is correct even in this edge case. An explanation using r_{13} will, however, not show any hint about b4 being the winning move. This is not a problem because a player can simply find the best move by for example running a chess engine for half a second and looking at the first move it displays. Similarly, a software program using an explanation model can find moves using standard chess engines. The important part is that the explanation contains information about the concepts required to understand the position. After grasping the concepts the student should have enough insight into the position to find the move.

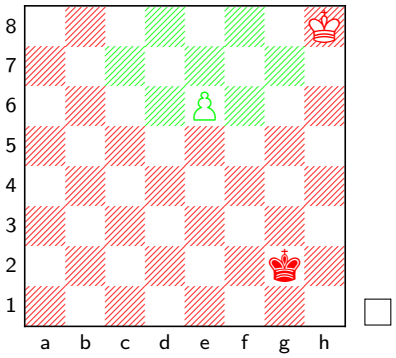
3.7.5 Are the rules sufficient for all KPvK endgames?

Whether the set of rules is sufficient to explain all KPvK endgames is an important factor to take into consideration. The set of rules is made using the limited information about KPvK endgames on the chess programming wiki in combination with general chess knowledge about endgames. Some adjustments are done to construct a set that should cover all KPvK endgames. For the 7th rank, it was relatively easy to prove the completeness of the set because of the symmetrical nature of the game. For endgames on the entire board, it becomes a bit more complicated. To make it simpler some illustrative diagrams are shown in figure 3.13 to show the impact of the square rule. The diagram shows that using the square rule results in a large number of positions being similar. The position of the white king does not matter in the diagram because it plays no role in promoting the pawn. All positions where the defending king is in range of the pawn can be further divided into two categories: positions where the attacking king is closer to a key square and positions where it is not. When the attacking king is closer to a key square the endgame is always winning. In the rules is defined that endgames are winning once a key square is reached. There is, however, one exception to this rule. This exception can be illustrated using the diagram in figure 5.3(a). As explained earlier the position is winning when playing Ka6. The position is also winning with Kc6, albeit through repeating the position. With this information, a position can be constructed that does not result in a win. If the move Kc6 would lead to a repetition the position is drawn. Furthermore, the position is drawn when trying to avoid repetition as it leads to a stalemate.

When it is not directly clear if a key square can be reached it either can be reached with the use of opposition, or with some other move sequence using r_{20}. Rule r_{20} conveniently ensures that the rules cover all scenarios. All other positions are simply a draw because a key square cannot be reached. The only positions not yet discussed are flank pawn positions, these are covered with rules r_{17} and r_{18}.

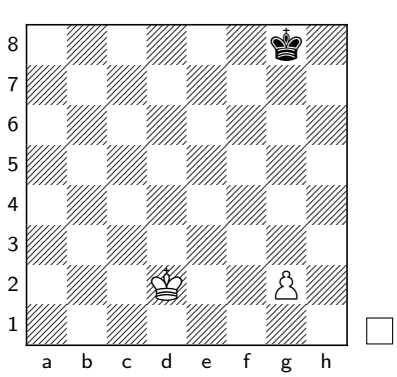


(a) Putting the king on any of the red squares leads to the same result

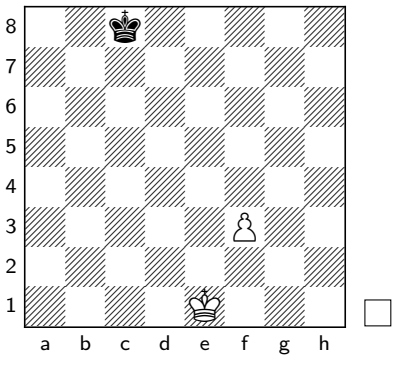


(b) The sides are out of range

Figure 3.13: More than half of the positions are eliminated using the square rule

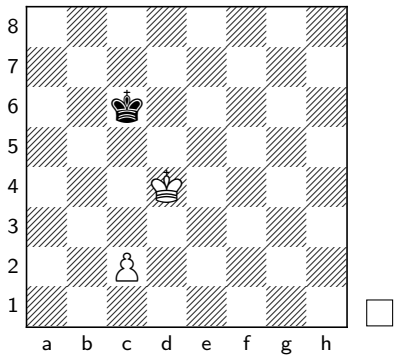


(a) Simple path towards key square f4

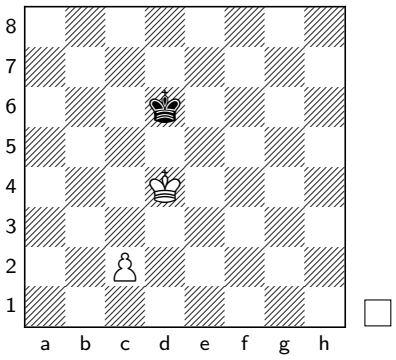


(b) Towards furthest key square and get opposition

Figure 3.14: Examples explainable using key squares

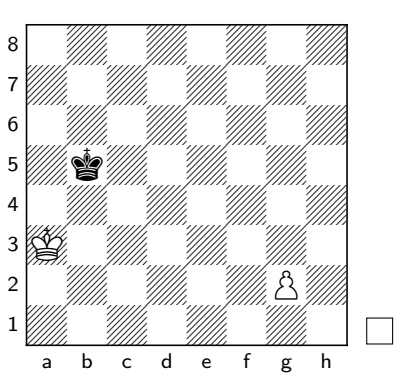


(a) Kc4 moves to a key square (and keeps opposition)

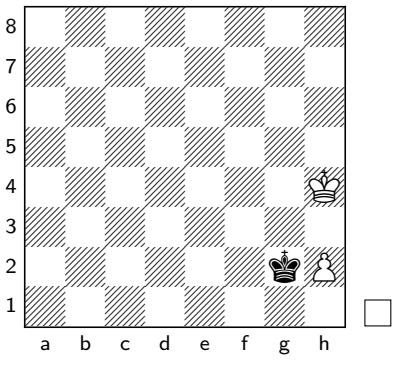


(b) Kc4 moves to a key square, c3 takes opposition

Figure 3.15: Examples where multiple concepts are applicable

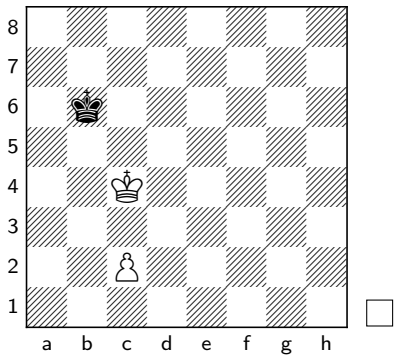


(a) Kb3 takes opposition

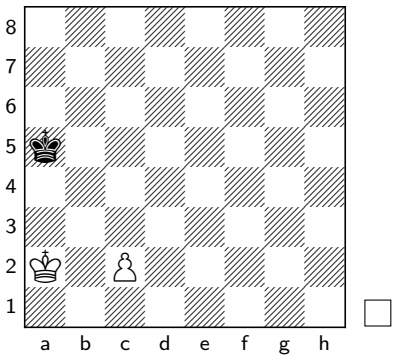


(b) The move h3 keeps the pawn and the white King will reach g7 first

Figure 3.16: Odd looking endgames with a straightforward explanation

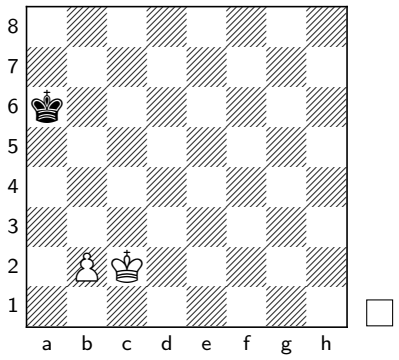


(a) The king is on a key square

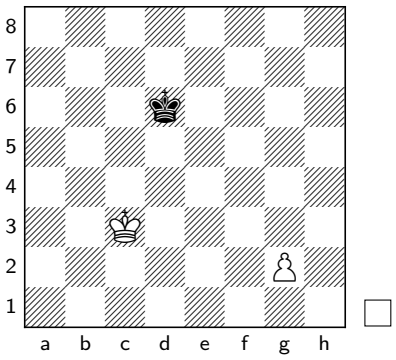


(b) Ka3 takes opposition

Figure 3.17: Two endgames that may misdirect players

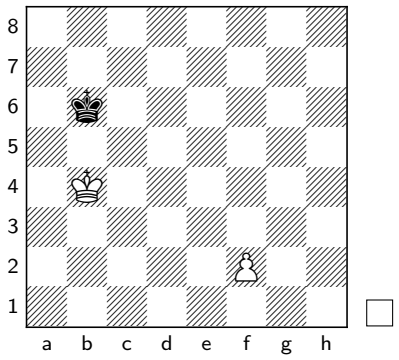


(a) Kc3 is the only move that wins, threatens moving to a key square and prevents opposition

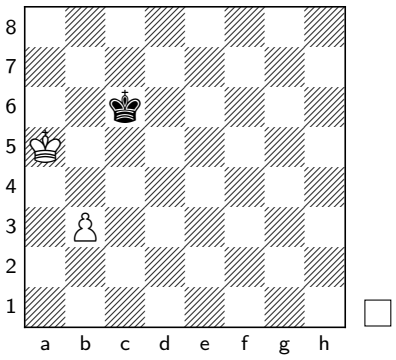


(b) Kd4 is the only move that wins

Figure 3.18: Advanced application of endgame motives



(a) Run to a key square or f3 for opposition (transposes)



(b) b4 enables white to get opposition

Figure 3.19: Two more instructive endgames

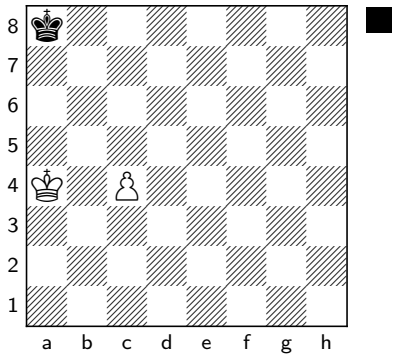


Figure 3.20: Black defends with Kb8

Chapter 4

Using the model to generate explanations

With the ASPIC+ model, specified explanations should be simple to generate. In this chapter, the construction of explanations is discussed. This is followed by several examples.

4.1 Constructing an explanation

In diagram 5.3(b) a position from the book Dvoretsky's endgame manual [9] is shown. This position is instructive, as only one of the king moves to a key square leads to a simple win, while the other move leads to a repetition of the position. According to the model, the position is winning. The knowledge base applicable to the position is as follows: $K_n = IsKPK(P_{5.1b})$, $IsPromotionDefended(P_{5.1b})$,

$IsPawnDefended(P_{5.1b})$,

$IsMove(P_{5.1b})$,

$IsInSquare(P_{5.1b})$,

$IsInFront(P_{5.1b})$,

$IsCloserKeySquare(P_{5.1b})$. Below in example 4.1.1 the arguments that can be constructed are discussed:

Example 4.1.1 *Using the knowledge base, two arguments can be constructed:*

(A)
- *A1:*$IsCloserKeySquare(P_{5.1b})$
- *A2:*$A1 \Rightarrow IsWin(P_{5.1b})$ *(applying* r_{13}*)*

(B)
- *B1:*$IsCloserKeySquare(MakeMoves($*1.Kc2 Ke7 2.Kb3 Kd6 3.Ka4 Kc6 4.Ka5 Kf7 5.Kb5* $, P_{5.1b}))$
- *B2:*$B1 \Rightarrow IsWin(P_{5.1b})$ *(applying* r_{20}*)*

Both arguments A2 and B2 defeasibly support that the position is winning. No arguments can be constructed for the position being drawn. A G-Dispute, with the proponent trying to show that $IsWin(P_{5.1b})$ *is admissible, can be argued in the following two ways:*

- P_1*: A2,*
- P_1*: B2.*

This shows that the proponent can either start with A2 or B2 and the opponent has no response.

In example 4.1.1 an argument for a winning position can be constructed in two ways. It is shown that there is an acceptable argument A2 where

$IsCloserKeySquare(p)$ supports that the endgame is winning for white. This information can be used to construct an explanation for the position. An explanation would look like: "The white king is closer to a key square (a6), therefore white is winning." Argument B2 can also be used to construct an explanation, and is perhaps even more useful in this case. It adds value by displaying an example set of moves where black approaches the key square: 1.Kc2 Ke7 2.Kb3 Kd6 3.Ka4 Kc6 4.Ka5 Kf7 5.Kb5. Other than this, an explanation would be the same.

To give more insight in how explanations are generated, a few more examples are discussed. The diagram in figure 3.16(a) provides a good example for a common KPvK endgame.

The human explanation for the position in the diagram would be something along these lines: "The white king can claim opposition with Kb3, this opposition will be converted to a position where white can get the king to a key square."

Generating an explanation using arguments would go the following way:

Example 4.1.2 *From the diagram the following knowledge can be extracted:*
$K_n = IsKPK(P_{3.16(a)})$,
$IsInFront(MakeMove(M_{Kb3}, P_{3.16(a)})$,
$IsMove(P_{3.16(a)})$,
$IsOpposition(MakeMove(M_{Kb3}, P_{3.16(a)}))$. *Using the knowledge base, the following arguments can be constructed:*

(A)

- *A1:*$IsMove(P_{3.16(a)})$
- *A2:*$IsOpposition(MakeMove(M_{Kb3}, P_{3.16(a)})$
- *A3:*$IsInFront(MakeMove(M_{Kb3}, P_{3.16(a)})$
- *A4:* $A1, A2, A3 \Rightarrow IsWin(P_{3.16(a)})$ *(*r_{14}*)*

(B)

- *B1:*$IsCloserKeySquare(MakeMoves($*1.Kb3 Kc5 2.Kc3 Kd5 3.Kd3 Ke5 4.Ke3 Kf5 5.Kf3 Kg5 6.Kg3 Kf5,* $P_{3.16(a)}))$
- *B2:*$B1 \Rightarrow IsWin(P_{3.16(a)})$ *(applying* r_{20}*)*

A4 defeasibly supports that the position is winning. B2 strictly supports that the position is winning. No arguments can be constructed for the position being drawn. A G-Dispute, with the proponent trying to show that $IsWin(P_{3.16(a)})$ *is admissible, can be argued in the following ways:*

- P_1*: A4,*
- P_1*: B2,*

This shows that the proponent wins the argument starting with either A4 or B2 because there is no valid response from the opponent.

Using the argument that followed out example 4.1.2 the following explanation can be constructed: "White is to move, the move Kb3 claims the opposition while also keeping the white king in front of the pawn. These positions can be converted to a win by using opposition to get to a key square. White is winning"

The example can also be explained using rule r_{20}. The corresponding sequence 1.Kb3 Kc5 2.Kc3 Kd5 3.Kd3 Ke5 4.Ke3 Kf5 5.Kf3 Kg5 6.Kg3 Kf5

does display how the position is converted. The applied rule does, however, mention nothing with regard to the applied concept opposition. The move sequence should not be the main part of the explanation. This is information that a user can just find by exploring the position with a chess engine. An explanation may, for example, look like this: "The sequence 1.Kb3 Kc5 2.Kc3 Kd5 3.Kd3 Ke5 4.Ke3 Kf5 5.Kf3 Kg5 6.Kg3 Kf5 is forced as deviation leads to a concession in the position, feel free to ask for an explanation of any potential deviation. If the sequence is followed by the players, the resulting position is winning for white because the king is closer to the key square h5". Options like rule priorities or a construction where rule r_{20} is only evaluated when no arguments for the position being winning can be made could resolve problems around this. But the main factor here would be the information desired in the explanation.

The position in figure 3.4(b) is another good example of how an explanation will be generated.

Example 4.1.3 *The knowledge base consists of the following information:*
$K_n : IsKPK(P_{3.4(b)}), IsMove(P_{3.4(b)}), IsPawn7th(P_{3.4(b)}),$
$IsPromotionDefended(P_{3.4(b)}), IsPawnDefended(MakeMove(M_{Ka6}, P_{3.4(b)}),$
$\neg IsBlocked(P_{3.4(b)}), IsCapture(MakeMove((M_{b8=Q}, P_{3.4(b)}),$
$IsPromotion(MakeMove(MakeMove(M_{b8=Q}, P_{3.4(b)}), IsFurtherKeySquare(P_{3.4(b)})$

(A)
- *A1:*$IsMove(P_{3.4(b)})$
- *A2:*$IsPromotion(MakeMove(M_{b8=Q}, P_{3.4(b)})$
- *A3:*$A1, A2 \Rightarrow IsWin(MakeMove(M_{b8=Q}, P_{3.4(b)})$ (r_0)

- *A4:*$A1, A3 \rightarrow IsWin(P_{3.4(b)})$ (r_*)

(B)

- *B1:*$IsMove(P_{3.4(b)})$
- *B2:*$IsCapture(MakeMove(M_{b8=Q}, P_{3.4(b)})$
- *B3:*$B1, B2 \rightarrow IsDraw(MakeMove(M_{b8=Q}, P_{3.4(b)})(r_2)$

(C)

- *C1:* $IsMove(P_{3.4(b)})$
- *C2:*$IsPawn7th(P_{3.4(b)})$
- *C3:*$IsPawnDefended(MakeMove(M_{Ka6}, P_{3.4(b)})$
- *C4:*$C1, C2, C3 \Rightarrow IsWin(P_{3.4(b)})(r_{10})$

(D)

- *D1:* $IsPromotionDefended(P_{3.4(b)})$
- *D2:*$\neg IsBlocked(P_{3.4(b)})$
- *D3:*$D1, D2 \rightarrow \neg C4(r_{11})$

(E)

- *E1:*$IsFurtherKeySquare(P_{3.4(b)})$
- *E2:*$E1 \Rightarrow IsDraw(P_{3.4(b)})(r_{23})$

The proponent starts with an argument for the position being a draw. The opponent can try two options (C4 and A4). Both can be refuted. C4 is undercut by D3. A4 is rebutted on A3 by B3.

- P_1*: E2,*O_1*: A4,* P_2*: B3 (Attacking A4 on the premise A3),*O_2*: C4,* P_3*: D3*
- P_1*: E2,*O_1*: C4,* P_2*: D3,*O_2*: A4,* P_3*: B3*

The example 4.1.3 shows that the proponent wins the argument starting with E2 because there is no strategy the opponent can follow to win. Both

strategies make use of the same arguments in a different order and therefore lead to the same explanation. The explanation based on the used arguments would look like this: "The attacking king is further from the key squares than the defending king, and promoting the pawn is not possible. Directly promoting the pawn leads to the capture of the pawn. Defending the pawn does not work because the defender can move in front of the pawn when stalemate is the only option left that defends the pawn."

As shown by the previous examples, it is relatively simple to generate an explanation in human language as long as the used predicates are expressible in human language and the used rules are justifiable with natural language.

4.1.1 Player level

Another variable that plays a role in how an explanation should be formulated is the level of the chess student. As seen in the previous section, some positions will be declared a winning position before the pawn is promoted. For stronger chess players, this is no problem at all because they have the required knowledge or calculation skills to do the rest of the work. For beginner level players, however, more detail may be required. An explanation should contain sufficient information to be able to discover the solution. This enables a student to win a position where the same ideas apply. For KPvK endgames the assumption will be made that the results of some positions, for example, the basic pattern in figure 4.1 with white to move, are known by the player. This prevents the need for additional rules that are not needed. Especially because this position specifically is a fundamental position in KPvK

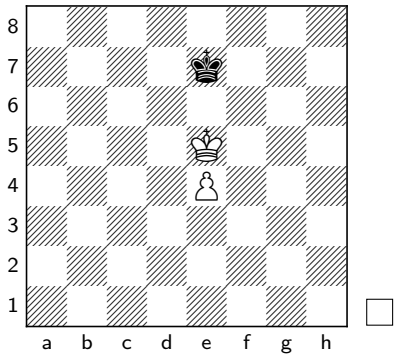


Figure 4.1: Basic position

endgames. A reference to a video or book involving a position with a similar pattern is sufficient.

Chapter 5

Evaluation of the model

In this chapter, the explanations are evaluated. In addition, some explanations generated by the model are compared to explanations generated by Decode Chess to give insight into how the argumentation model can contribute to better chess XAI.

5.1 Explanation metrics

Hoffman [13] discusses metrics that can be used to establish the quality of an explanation. The explanation goodness checklist mentioned can be used to get an idea of the quality of the explanation by simply answering the yes/no questions.

1. The explanation helps me understand how the [software, algorithm, tool] works.

2. The explanation of how the [software, algorithm, tool] works is satisfying.

3. The explanation of the [software, algorithm, tool] sufficiently detailed.

4. The explanation of how the [software, algorithm, tool] works is sufficiently complete.

5. The explanation is actionable, that is, it helps me know how to use the [software, algorithm, tool].

6. The explanation lets me know how accurate or reliable the [software, algorithm] is.

7. The explanation lets me know how trustworthy the [software, algorithm, tool] is.

Some points must be addressed before all these questions can be objectively answered with yes/no for a generated explanation. First of all the it should be established what the questions refer too. For this the algorithm that generates an explanation for why the position is winning/drawn will be used. Below the adjusted questions are listed.

1. The explanation helps me understand how the algorithm established that the position is [winning,drawing].

2. The explanation of how the algorithm established that the position is [winning,drawing] is satisfying.

3. The explanation how the algorithm established that the position is [winning,drawing] is sufficiently detailed (The moves that lead to the [win,draw] can be constructed after seeing the explanation).

4. The explanation of how the algorithm established that the position is [winning,drawing] is sufficiently complete (leaving no questions about plans and the possible replies of the opponent leading to the suggested result).

5. The explanation is actionable, that is, it helps me know how to use the plan showed by the algorithm in a similar position.

The last two questions are intentionally left out because the result of the position can be checked with an endgame tablebase to confirm that the conclusion about the position is correct.

5.2 Use of chess engines

When players require an explanation for a specific position, they usually (wrongly) use a chess engine. Chess engines are nowadays excellent at finding the best move in any given position. When analysing chess games clicking through engine lines, however, people tend to get a false sense of understanding. Engines show the way, but leave underlying chess concepts out of the equation. The goal of the explanation model is to uncover these underlying concepts. The purpose of the model is not to point out the best move, but simply enrich the information that a chess engine gives about strong moves

with conceptual justification.

5.3 Evaluation of the model

The explanations of several insightful positions discussed earlier are evaluated with the questions described in section 5.1. After this, an example will be compared to an explanation that was generated using DecodeChess before the update of April 2023. This may provide valuable insight for further research. It is important to note that DecodeChess is in no way specialized for endgames, it is meant as a generic one size fits all solution to chess XAI.

5.4 Tested positions

In example 4.1.1 the following explanation is generated: "The white king is closer to a key square (a6), therefore white is winning."

1. The explanation helps me understand how the algorithm established that the position is [winning, drawing]: Yes, the algorithm bases it on the position of the king in relation to the key square.

2. The explanation of how the algorithm established that the position is [winning, drawing] is satisfying: Yes, if familiar with the concept of key squares, but a move sequence might help as discussed after the example.

3. The explanation of how the algorithm established that the position is

[winning, drawing] is sufficiently detailed (The moves that lead to the [win, draw] can be constructed after seeing the explanation): No, chess knowledge is still required, but it does give the goal for the endgame.

4. The explanation of how the algorithm established that the position is [winning, drawing] is sufficiently complete (leaving no questions about plans and the possible replies of the opponent leading to the suggested result). Yes, although again an example sequence of moves might help, reaching the key square is the one important plan in the position.

5. The explanation is actionable, that is, it helps me know how to use the plan shown by the algorithm in a similar position. Yes, looking for key squares and how to reach them is important in such endgames and if spotted in a similar position converting the position to a win goes exactly the same way as in this position.

In example 4.1.2 the following explanation is generated: "White is to move, the move Kb3 claims the opposition while also keeping the white king in front of the pawn. These positions can be converted to a win by using opposition to get to a key square. White is winning"

1. The explanation helps me understand how the algorithm established that the position is [winning, drawing]: Yes, the algorithm bases it on opposition and the king being in front of the pawn.

2. The explanation of how the algorithm established that the position is [winning, drawing] is satisfying: Yes, it is clear how to convert the position to a key square position.

3. The explanation of how the algorithm established that the position is [winning, drawing] is sufficiently detailed (The moves that lead to the [win, draw] can be constructed after seeing the explanation): Yes, following the mentioned concepts key square and opposition converting the position is almost foolproof.

4. The explanation of how the algorithm established that the position is [winning, drawing] is sufficiently complete (leaving no questions about plans and the possible replies of the opponent leading to the suggested result). Yes, although again an example sequence of moves might help, reaching the key square is the one important plan in the position, which can be achieved using opposition.

5. The explanation is actionable, that is, it helps me know how to use the plan shown by the algorithm in a similar position. Yes, it is clear what move to play next and how to continue.

In example 4.1.3 the following explanation is generated: "The attacking king is further from the key squares than the defending king, and promoting the pawn is not possible. Directly promoting the pawn leads to the capture of the pawn. Defending the pawn does not work because the defender can move in front of the pawn when stalemate is the only option left that defends the pawn."

1. The explanation helps me understand how the algorithm established that the position is [winning, drawing]: Yes, the pawn cannot be promoted or defended because it will be lost or lead to a stalemate.

2. The explanation of how the algorithm established that the position is [winning, drawing] is satisfying: Yes, it is clear that nothing more than a draw can be reached and how to reach it as the defender.

3. The explanation of how the algorithm established that the position is [winning, drawing] is sufficiently detailed (The moves that lead to the [win, draw] can be constructed after seeing the explanation): Yes, the defending side simply gets in front of the pawn if it is defended.

4. The explanation of how the algorithm established that the position is [winning, drawing] is sufficiently complete (leaving no questions about plans and the possible replies of the opponent leading to the suggested result). Yes, all reasonable alternatives are considered.

5. The explanation is actionable, that is, it helps me know how to use the plan shown by the algorithm in a similar position. Yes, it is clear how to defend.

Of these three explanations, the third one checked the most boxes, overall the model provides good explanations.

5.5 Comparison to DecodeChess

A point of interest is whether the ASPIC+ based model expands on current technology. Therefore this section aims to give insight into the comparison between the ASPIC+ based model and DecodeChess [2], the market leader

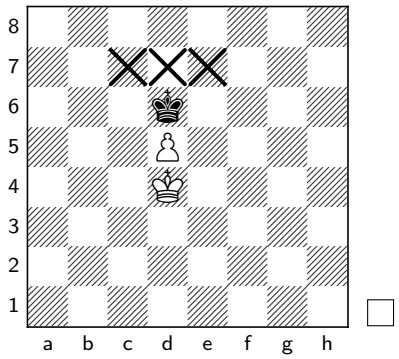


Figure 5.1: A fundamental chess endgame position, white to move draws and black to move wins

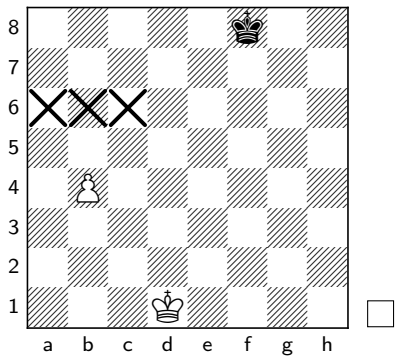


Figure 5.2: Highlighting the importance of finding the shortest distance to a key square

in explainable chess AI.

Below is a description of the elements DecodeChess uses for an explanation. The first thing that shows up in DecodeChess is the best move sequence in the position (Suggested by the newest version of the engine, Stockfish). This is accompanied by an explanation for the good moves. For the explanation, several topics can be manually selected in DecodeChess: a summary, the piece roles, the threats, good moves, plans, and concepts. The explanation for figure 5.1 is brief and mentions that Ke4 enables Kd4 in the next position. This seems like an odd plan, but it is somewhat understandable as the position is drawn. It does highlight the elementary understanding that the pawn needs to be defended, preventing the black player from capturing it. In an attempt to prevent stalemate, it demonstrates two seemingly random lines where the white player abandons the defence of the pawn instead of demonstrating the stalemate. This is annotated with the text: "The black king on d8 guards square d7". For this position, all other windows in DecodeChess show no relevant information to the position.

A good example to show where the ASPIC+ model could help is example 4.1.1. This example is also analysed by DecodeChess. The ASPIC+ based model concludes that the position is winning because there is a sequence that ensures that white reaches a key square with the king. This sequence is also highlighted by DecodeChess. Other than this, DecodeChess highlights that black threatens to play Ke8, which could be interpreted as: "If black were to move, the game is a draw." One interesting thing to note is what De-

codeChess mentions under the tab concepts. It highlights the b8-h2 diagonal and suggests that the move Kc2 uses this diagonal. This is an unorthodox way to describe the position, it highlights a point far in the future where the pawn is promoted and controls the diagonal as a queen. This is the checklist for the DecodeChess explanation:

1. The explanation helps me understand how the algorithm established that the position is [winning, drawing]: Yes, a sequence is shown that wins.

2. The explanation of how the algorithm established that the position is [winning, drawing] is satisfying: No, but with chess knowledge one can manage to figure it out.

3. The explanation of how the algorithm established that the position is [winning, drawing] is sufficiently detailed (The moves that lead to the [win, draw] can be constructed after seeing the explanation): No.

4. The explanation of how the algorithm established that the position is [winning, drawing] is sufficiently complete (leaving no questions about plans and the possible replies of the opponent leading to the suggested result). Yes, the move sequence is the most relevant.

5. The explanation is actionable, that is, it helps me know how to use the plan shown by the algorithm in a similar position. No, especially the comment controlling the diagonal is vague and confuses more than it helps.

For this position, the ASPIC+ generated explanation is better.

5.6 Comparison to literature

The endgame manual by Mark Dvoretsky [9] is the golden standard for chess endgames. The explanation quality is good, but the main feature of the book is the variety of theoretical endgames covered. All theoretical concepts required to play endgames at a high level are discussed in the book. The book is written like a manual, meaning that it does not explain positions word for word, but tries to provide all underlying concepts required to solve a position. It is assumed that the reader is capable of doing calculations of move sequences. For pawn endgames, the book explains that accurate calculation is an important part. The other important part is knowledge of "standard techniques"(in this document referred to as (positional) concepts) to make calculation easier. The text in the book is interwoven with move sequences, these sequences are interrupted by additional explanation where deemed necessary. The book uses standard signs and symbols:

- ! a strong move
- !! a brilliant or unobvious move
- ? a weak move, an error
- ?? a grave error
- !? a move worth consideration
- ?! a dubious move

Two players usually agree upon the evaluation of a move, but it is still somewhat subjective. Generating these symbols as annotation is occasionally done

with computers, but this fails to take human insight into account and only considers the size of the loss in computer evaluation of a move. The symbols are therefore not taken into account with the generation of an explanation, but still displayed for explanations from the book to give some extra context. For KPvK endgames specifically, here are two examples discussed in the book that give a good idea of what an explanation should look like, and how these symbols should be used:

In diagram 5.3(a) 1.Ka6! Ka8 2.b6 Kb8 3. B7 (White has a winning position) Note that 1.Kc6 is inaccurate in view of 1...Ka7, forcing white to repeat the initial position to prevent stalemate.

There are two things to note about this explanation. Firstly the explanation assumes that the concept of key squares is known as it is explained earlier in the book (the key squares are highlighted in the diagram), this assumption cannot be made in a standalone explanation. Secondly, the explanation considers a move that repeats the position inaccurate. A computer sees a repetition simply as a different, albeit longer, sequence. The computer will in this case consult the objective evaluation of the sequence, showing that the line is good (when choosing between the two options a computer will do whatever the settings specify, in some cases that means playing the shortest sequence, and in other cases like computer tournaments where it requires extra calculation time it will repeat the position).

The second example (diagram 5.3(b) is lead with the text: "The key squares are a6,b6 and c6. The sensible thing here is to head for the square farthest from the enemy king since that will be the one hardest to defend." Even

though this explanation is sufficient, it is still followed with a sequence of moves to show the idea in practice. 1.Kc2! Ke7 2.Kb3 Kd6 3.Ka4 (3.Kc4? Kc6) 3...Kc6 4.Ka5 (with the idea of Ka6) 4...Kb7 5.Kb5 (and white is winning).

Again, it is important to note that a computer will not annotate the exclamation mark as the valuation does not change (it could mark the move as only move). The explanation does not spend any textual description that 3.Kc4 does not achieve the goal of reaching a key square, this is assumed to be understood by the reader upon seeing the move sequence.

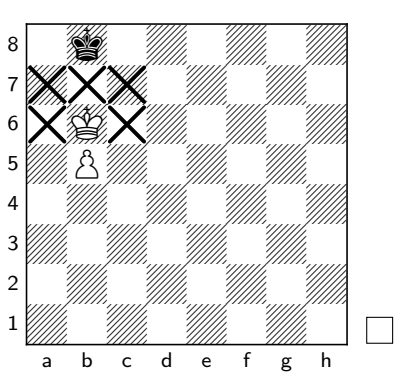
This is the explanation goodness checklist for the example in diagram 5.3(b):

1. The explanation helps me understand how the writer established that the position is [winning, drawing]: Yes, the explanation gives a sequence until the point where it is clear that the position is winning.

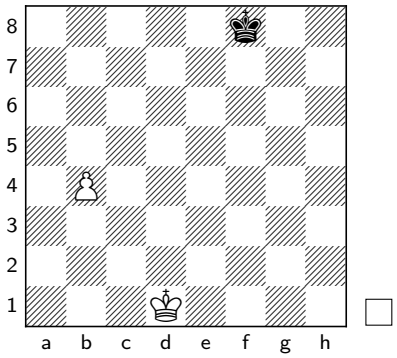
2. The explanation of how the writer established that the position is [winning, drawing] is satisfying: Yes, the explanation mentions alternatives and why they are not good.

3. The explanation of how the writer established that the position is [winning, drawing] is sufficiently detailed (The moves that lead to the [win, draw] can be constructed after seeing the explanation): Yes, one small side note is that the book skips over information mentioned earlier in the book.

4. The explanation of how the writer established that the position is [winning, drawing] is sufficiently complete (leaving no questions about plans



(a) 5th rank pawn example



(b) White heads for the square furthest from the opponent's king

Figure 5.3: Two fundamental endgames presented in Dvoretsky's endgame manual

and the possible replies of the opponent leading to the suggested result). Yes, all relevant information is either mentioned or discussed in earlier examples in the book.

5. The explanation is actionable, that is, it helps me know how to use the plan shown by the writer in a similar position. Yes, the described plan is clear and easy to execute.

The book checks all points. One important advantage the book has is that it can refer to earlier examples and build upon them in gradual steps. This allows for a detailed explanation without the need to go into unnecessary detail every time. In a book, the ASPIC+ generated explanations would miss some essential points. If the explanations are used solely to expand on chess engine lines or additional context is given by referencing a book, then the generated explanations are valuable.

Chapter 6

Discussion and Conclusions

The goal of this thesis was to get insight into whether computational argumentation can be used for the explanation of chess endgames. This has been explored by attempting to implement an argumentation model using ASPIC+. The level of detail in explanations has been briefly explored by comparing explanations to the book Endgame Manual by Dvoretsky [9] and the software DecodeChess. This gave insight into whether argumentation can be used to generate an explanation that is on par.

6.1 Results

The main research question of this thesis, "Can computational argumentation be used for the explanation of chess endgames?" has been addressed using several research questions:

1: How can argumentation for chess XAI be formalized using AS-

PIC+?

The first research question, regarding formalization using ASPIC+, has been addressed in chapter 3. In this chapter, A comprehensive set of 23 predicates and 23 rules is established to model KPvK endgames. The current board position can be used to construct a knowledge base. For KPvK endgames specifically, individual rule preferences do not need to be specified due to the simplicity of the endgame. For the established set of rules, it makes sense to give strict-and-firm rules priority over other rules.

2: How detailed should arguments be?

The second research question is used to discuss the level of detail required for arguments to translate into good explanations. The level of detail in arguments required for explanation is briefly discussed. Stopping explanations when a motif is present that guarantees a winning position according to chess literature is the used approach in this thesis. For KPVK endgames this entails stopping when a queen endgame is reached, for example as discussed in chapter 3.3. The rules in this chapter rely on chess concepts to decide whether a position is winning. For example, the concept of key squares is widely used in all chess literature and is a good stopping point for an explanation. The importance of sequences for explanations is discussed in chapter 3.7.2 and in figure 3.14(b) in chapter 3.7.4. In this chapter, it is addressed how move sequences can play a role in an explanation model. Another important aspect of the level of detail in arguments that has been discussed is player level. In chapter 4.1.1 it is discussed that the level of the player using the explanation may impact how much detail is required for an expla-

nation. Referencing books and videos of the level of the player can add to an explanation without having to overcomplicate the model.

3: What defines a good explanation for chess endgames?

The third question is used to determine the requirements of a good explanation. In chapter 5.1 a set of questions is discussed that can be used to verify whether an explanation is good. This is a set of five questions which can be answered with yes/no resulting in a clear overview of the goodness of an explanation. These questions are adapted for chess endgames, and some additional explanation is given for context. The questions are used in the next research question to confirm the quality of the model.

4: Is argumentation using the defined approach sufficient for a good explanation?

The fourth question addresses the quality of generated explanations. The metrics established in question three are used to evaluate a sample of explanations that are generated using the model in chapter 5.4.

Furthermore, one of the explanations has been compared to an explanation generated by DecodeChess in chapter 5.5.

4.1: How does the model generate an explanation?

In chapters 3.4.1 and 3.7.2 25 rules are listed. These rules are used as the cornerstone of the model. In chapter 4.1 the grounded argument game is used in combination with the rules that can be constructed from the premises available in the knowledge base. This chapter gives several examples of a chessboard position being turned into an explanation.

4.2: How does a generated explanation get evaluated by the established metrics?

The examples generated in chapter 4.1 are evaluated using the metrics. The examples score the following three scores: 4/5, 5/5, 5/5. The only lacking area was the level of detail in the explanations.

4.3: How does the model perform on those metrics in comparison to the software Decode Chess?

One of the examples is compared to DecodeChess. DecodeChess scored 2/5. This score may not be completely fair because Decode Chess is built as a product that gives insight into all game stages. This shows that an ASPIC+ based model has the potential to outperform a market leader, albeit in a specialized section. The ASPIC+ based model does, however, give up some scalability and flexibility in comparison.

6.2 Conclusion

The main research question of this thesis, "Can computational argumentation be used for the explanation of chess endgames?", can now be answered. It seems that computational argumentation has its uses for the explanation of chess endgames. The explanations generated using ASPIC+ give insight into the positions, and the quality of this explanation fully depends on how well the underlying rules are formulated. The generated explanations seem

to improve or at least add to current technology. Both in this thesis and related literature, a subset of chess endgames is approached. The wide variety of chess endgames makes it difficult to cover it all. The KPvK endgame is relatively simple, yet 24 rules were required in this thesis to generate explanations. With the increasing complexity of other endgames, manually generating rules may not be feasible.

As long as the rules follow a clear structure, the quality of the explanation follows. As seen in the explanations in chapter 4.1 rules for KPvK endgames translate well to an explanation. The textual description of the rules, however, needs to be detailed while also being generic enough that it fits all situations where the rule applies. Any detail outside the rules will not be mentioned in the explanation. This provides extra integrity for an explanation but may impact completeness. This may lead to cases where the answer is technically correct but omits some of the complications in the position.

In comparison to the only product in a similar niche, DecodeChess, albeit in only one example, the model performs well for KPvK endgames. This was to be expected, as the explanation model is specifically designed for KPvK endgames, where DecodeChess is a generic product that performs well in middlegames and openings.

6.3 Limitations

While the proposed model seems to result in good explanations using the established metric, this is only shown with a small set of rules. It is important to note that only a small subsection of endgames has been discussed. A similar approach may achieve different results in other endgame contexts. Furthermore, the set of rules and the relevance of these concepts in an endgame are one of many ways the endgame may be approached. The concepts are generally used in chess explanations, but there may be other ways to formulate them. This is more relevant in other types of endgames where there could be two (or more) entirely different solutions that lead to the same final result. This comes with the additional problem that computers and algorithms are not well suited for picking the simplest option.

Another limitation is the way the model has been tested. While the positions are checked for exceptions, it may be possible to construct a position where the generated explanation omits information that is of importance.

Another limitation of the model is the way it brushes over tactical lines and only discusses the beginning and end positions. The historical development of XAI in chess is interesting. It used to be a topic of interest until the focus shifted to performance and creating an engine capable of defeating humans. In the late 70s, the program PARADISE (Pattern recognition applied to directing search) was developed by David Wilkins [18]. This program used patterns to navigate tactical sequences. The calculation of a position results in a tree built from those patterns. Using an approach like this, a structure might be given to move sequences in a way a student's understanding is

enhanced, especially when it is an alternative to just brushing over the long lines like they are trivial calculations. PARADISE is not capable of playing games as a program, the program is limited to tactical positions and has no use in endgames or strategic positions. The performance of PARADISE on puzzles, however, shows that it may be interesting to not have a one-size-fits-all solution but to have several types of specialized models.

Finally, another limitation is that the full power of ASPIC+ has not been explored. The ASPIC+ framework is powerful and allows for a lot of freedom when implementing an argumentation framework. The approach taken in this thesis is quite rigid, with rules having much information on the left-hand side and a conclusion about whether a position is winning or drawn on the right-hand side. This results in an argumentation framework where there are mostly conflicts in the conclusion. Rules constructed with intermediate goals as aim instead of draw or win as a conclusion could lead to different results.

6.4 Further Research

Chess endgame explanation is not widely researched. This leaves ample possibility for further research, both interesting for the field of XAI and the game of chess. Another point of interest is the development of neural network engines, this may enable more research in the direction of chess XAI as these engines could be well suited for the field. This thesis has shown that argumentation can play a big role in building reliable XAI for chess endgames

on a small scale. However, some questions remain about the scalability to other endgames and getting the explanations on the level of human chess instructors.

6.4.1 Comparison to chess instructors

Explanations by a strong player are usually satisfying the curiosity of the player. The field of how these explanations help in increasing the chess-playing capabilities of students has not been heavily studied. Therefore, it is difficult to measure the objective effect of human- and generated-explanations on the performance of the player. Another factor coming into play is the period after which an explanation gets incorporated into play. Players may perform worse in the short term after receiving new information and trying to incorporate it into their play. Winning endgame positions are useful in this regard, as there is usually a direct path that the student will understand fully or not at all. A way to test whether the model improves the player is by separating a group of players of a specific level into three groups. One group that does not gain instruction. One that attends a lecture about the endgames. And one that uses a model that generates (interactive) explanations. Afterwards, a test with similar (not the same, as that, would just be a memory exercise) positions will give insight into whether the model improves performance (compared to studying under the lecturer).

6.4.2 Scale-ability of the model

One of the main issues with the explainability of AI models for chess is the deep richness of different concepts in the game. Not only does a good model require knowledge of all relevant concepts, but a model also needs the information to construct rules from these concepts. In the paper "Acquisition of Chess Knowledge in AlphaZero" [14] 116 concepts have been implemented. These concepts vary from tactical concepts such as pins and forks, to positional concepts like several pawn structures. With neural networks, it is possible to harness information about related concepts to construct rules. It is, however, hard to say anything about the complexity of the resulting rules. The 116 concepts mentioned cover a wide range of positions, usable in all types of endgames. It is, however, interesting to note, that the number of concepts related to simple KPvK endgames is limited to far fewer than mentioned here. This may be because these endgames are considered trivial for top level humans, and especially for chess engines. With traditional engines, it is difficult to explain all chess endgames. A different direction to look in is neural network based engines that use concepts understandable by humans, like the network discussed in the aforementioned paper by McGrath et al. [14]. This requires some thought on two topics:

1. The detail of explanation in endgames that seem like simple calculations for an engine.

2. Using the classified concepts in rules and how the resulting rules can be used to construct coherent explanations.

Successful implementation of such an interface between neural network engines and human explanation would be an exciting leap for both chess and XAI.

In summary, argumentation can be a great tool in chess XAI. There are still some questions about the scalability, but these can likely be resolved by combining the approach with recent technology in chess. Argumentation based models will not replace chess teachers in the short term, but they will be able to add value to current products such as DecodeChess in endgames.

Bibliography

[1] Chess Endgame Training — chess-endgame-trainer.firebaseapp.com. https://chess-endgame-trainer.firebaseapp.com/list/1/0. [Accessed 02-May-2023].

[2] DecodeChess - Chess Analysis, Powered by AI. https://decodechess.com/. [Accessed 02-May-2023].

[3] GitHub - syzygy 7man endgame tablebase. https://github.com/syzygy1/tb. [Accessed 01-Jun-2023].

[4] Kpk - chessprogramming wiki. https://www.chessprogramming.org/KPK. [Accessed 01-Jun-2023].

[5] Ivan Bratko, Dayana Hristova, and Matej Guid. Search versus knowledge in human problem solving: A case study in chess. In Model-Based Reasoning in Science and Technology, pages 569–583. Springer International Publishing, 2016.

[6] Devleena Das and Sonia Chernova. Leveraging rationales to improve human task performance. Proceedings of the 25th International Conference on Intelligent User Interfaces, pages 510—-518, 2020.

[7] Rachael Dottle. Netflix's "the queen's gambit" drives major boom in chess sales. Bloomberg, Dec 2020.

[8] Phan Minh Dung. On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games. Artificial Intelligence, 77(2):321–357, September 1995.

[9] Mark Dvoretsky, Karsten Müller, and Vladimir Kramnik. Dvoretsky's Endgame Manual. Russell Enterprises, Inc., 5 edition, 2020.

[10] Dinesh Gadwal, Jim E. Greer, and Gordon I. McCalla. Umrao: A chess endgame tutor. In International Joint Conferences on Artificial Intelligence, pages 1081–1086, 1991.

[11] Matej Guid and Ivan Bratko. Influence of search depth on position evaluation. In Advances in Computer Games, Lecture Notes in Computer Science, pages 115–126, 2017.

[12] Matej Guid, Martin Mozina, Ciril Bohak, Aleksander Sadikov, and Ivan Bratko. Building an Intelligent Tutoring System for Chess Endgames. Proceedings of the 5th International Conference on Computer Supported Education, pages 263–266, 2013.

[13] Robert R Hoffman, Shane T Mueller, Gary Klein, and Jordan Litman. Metrics for explainable ai: Challenges and prospects. arXiv preprint arXiv:1812.04608, 2018.

[14] Thomas McGrath, Andrei Kapishnikov, Nenad Tomaš ev, Adam Pearce, Martin Wattenberg, Demis Hassabis, Been Kim, Ulrich Paquet, and Vladimir Kramnik. Acquisition of chess knowledge in AlphaZero. Proceedings of the National Academy of Sciences, 119(47), nov 2022.

[15] Sanjay Modgil and Henry Prakken. The ASPIC+ framework for structured argumentation: a tutorial. Argument & Computation, 5(1):31–62, January 2014.

[16] Henry Prakken. Course reader Computational Argumentation. https://ics.uu.nl/docs/vakken/mcarg/casyl24.pdf.

[17] Henry Prakken. An abstract framework for argumentation with structured arguments. Argument & Computation, 1(2):93–124, June 2010.

[18] David Wilkins. Using Patterns and Plans in Chess, pages 233–257. Springer New York, New York, NY, 1988.