

Contents

- 1 Introduction** **5**

- 2 A brief history of wargames** **9**
 - 2.1 Early games 9
 - 2.2 Wargames 11
 - 2.2.1 Kriegspiel 11

- 3 Game theory notions** **13**
 - 3.1 Introduction 13
 - 3.2 Concepts 15
 - 3.2.1 Game 15
 - 3.2.2 Pure and mixed strategies 15
 - 3.2.3 Zero sum games 16
 - 3.2.4 Equalizing strategy 16
 - 3.2.5 Normal form 17
 - 3.2.6 Extensive form 18
 - 3.2.7 Imperfect and incomplete information 19
 - 3.2.8 Zermelo's Theorem 19

- 4 Kriegspiel** **23**
 - 4.1 Overview 23
 - 4.2 Rule variants 24
 - 4.3 Kriegspiel: gameplay samples 26
 - 4.3.1 Problem 1 26
 - 4.3.2 Problem 2 27

4.4	A notation for Kriegspiel games	28
5	Kriegspiel on the Internet Chess Club	35
5.1	The Internet Chess Club	36
5.1.1	A Kriegspiel game database	38
5.2	A Statistical Analysis of ICC games	40
5.2.1	Outcome	40
5.2.2	Game duration	41
5.2.3	Pseudomobility	42
5.2.4	Endgame material	43
5.2.5	Opening play	46
5.2.6	Further development	51
6	Kriegspiel Algorithms: State of the art	53
6.1	Algorithms for various Kriegspiel endgames	53
6.1.1	King and Rook vs. King (KRK)	55
6.1.2	KRK on Infinite Chessboard	56
6.1.3	King and Pawn vs. King (KPK)	57
6.2	The Monte Carlo sampling player	61
6.2.1	Strategy fusion	63
6.3	Metapositions	64
6.3.1	KRK with metapositions	69
7	Structure of the Darkboard engine	73
7.1	Representing metapositions	73
7.1.1	The main array	75
7.1.2	The age array	77
7.1.3	Other information	78
7.2	Move generation	79
7.3	Evolution of metapositions	81
7.3.1	Evolution with legal moves	82
7.3.2	Evolution with illegal moves	85
7.3.3	Evolution with opponent moves	86
7.4	The move selection routines	88
7.4.1	Game tree structure	90

<i>CONTENTS</i>	3
7.4.2 Umpire prediction heuristics	93
7.4.3 The basic decision algorithm	95
7.4.4 The enhanced decision algorithm	97
7.5 The evaluation function	99
7.5.1 Material safety	99
7.5.2 Position	102
7.5.3 Information	104
7.6 Other components	105
7.6.1 The opening book	105
7.6.2 Stalemate detection	107
7.7 Tuning the evaluation function	108
7.7.1 Evolutionary learning	109
8 Results and sample games	113
8.1 Some Darkboard games	114
8.1.1 Sample Game 1	114
8.1.2 Sample Game 2	118
8.1.3 Sample Game 3	121
8.1.4 Sample Game 4	124
9 Conclusions and future developments	129
A Kriegspiel Rules at The Gambit	131
B Kriegspiel Rules at RAND	135
C Kriegspiel Rules “Cincinnati style”	139
D Rules for Kriegspiel on the ICC	143

Chapter 1

Introduction

The present work deals with the game of Kriegspiel, an etherodox variant of chess with strongly imperfect information, invented by Henry Temple at the end of the XIX century; it follows the same rules as chess, except that the opponent's pieces are hidden from view, and the player must infer their position through careful reasoning. Kriegspiel means “war game” in German, and it was born as an attempt to turn chess into a more realistic warfare simulation, where the position of the enemy is often unknown.

Unlike chess, for which countless computer players of any strength level already exist, Kriegspiel is a very difficult testbed for Artificial Intelligence, and it has only been recently that the first, very few programs have begun to appear; however, research in this field is still early and none of these programs can, as of now, beat the strongest human players. Furthermore, the near totality of the theoretical results so far only apply to narrow subsets of the Kriegspiel problem, such as some famous endgame archetypes. This should not come as a surprise, for Chess and Kriegspiel, while sharing most of their rules, are in fact completely different games, and in general the established techniques used for the former cannot be tranferred to the latter.

In the development of this thesis, a new artificial player for Kriegspiel, named Darkboard, has been created, extending the results of previous research as well as introducing new experimental decision algorithms. Based on the concept of *metapositions*, introduced to artificially remove the lack of perfect information, it combines elements of traditional, well-known chess theory and *ad hoc* solutions unique to Kriegspiel. The program, written in the

Java programming language, has undergone a series of tests and fine-tuning through self-play, play against a random-moving opponent and finally online on the Internet Chess Club against real-world human opponents of all skill levels.

The road map for the present work is as follows. In Chapter 2, a brief and necessarily incomplete history of wargaming is given, reaching the twentieth century. Well before chess-Kriegspiel was born, other games had been given the name of Kriegspiel, and those were actual, and very serious, war simulations played on a tabletop. Some of the finest officers from the Napoleonic era through the XX century honed their tactical skills with such games.

Chapter 3 contains useful notions of Game Theory that will help in the remainder of this work, with a special emphasis on the fundamental differences between perfect and imperfect information games, and why certain fundamental results that concern perfect information games do not hold when considering games of imperfect information.

Chapter 4 introduces the modern chess variant of Kriegspiel, describing its various rulesets and regional twists. A few sample Kriegspiel problems are commented step-by-step, showing the typical reasoning that takes place in the typical Kriegspiel matches, and discussing the main differences between this variant and orthodox chess when it comes to strategy and tactics.

In Chapter 5, we discuss the role and importance of the Internet Chess Club for online Kriegspiel play. Not only is the ICC the premier place for playing this variant online, but it also provides a wealth of information about how humans play it. By gathering and analyzing what is probably the largest database of real-world Kriegspiel matches to date, numbering some twelve thousand games, it is possible to draw some conclusions as to the dominant behaviors shown by the top players.

Chapter 6 summarizes today's state-of-the-art knowledge for Kriegspiel. Algorithms and other results both theoretical and practical are given, mostly dealing with well-known endgame situations where victory can be attained with certainty or quasi-certainty; the chapter also includes a brief description of the existing artificial players that can play the game. Finally, special attention is devoted to Metaposition Theory, which lies at the foundation of the Darkboard engine.

A much more thorough discussion of the Darkboard artificial player is

given in Chapter 7. The Chapter goes into detail describing how knowledge is represented, manipulated, and inferred, and then illustrates how the move selection algorithms and the evaluation function work. Notable subsystems, such as the stalemate prevention system, are also dealt with. Also, the chapter outlines the processes of training and fine-tuning the engine through a form of self-play and examining possible ways of learning from the existing game database. Problems such as assigning reasonable weights to the various components of the evaluation function are reported here.

Chapter 8 presents and comments on several matches played by Darkboard against human and computer opponents on the Internet Chess Club. Transcripts of important victories and defeats help to evaluate Darkboard's current strengths and weaknesses.

Finally, Chapter 9 contains some final thoughts about likely future developments to improve the program.

Chapter 2

A brief history of wargames

2.1 Early games

Mankind has played games since the dawn of times, but the first games of which we possess documented evidence date back to about 5,000 years ago, and while there are several candidates for the position of most ancient game ever, they all share a few traits; they are board games for two players with a pre-marked territory and a set of tokens to be placed on the board itself. All of them presumably held some symbolic, religious or cultural meaning in addition to their sheer playing value, and, while some of them may have allowed a random element, they relied more on skill than on luck.

The most ancient finding pertaining to a game is the depiction of the Egyptian game of **Senet** in the tomb of Merknera, which makes other appearances throughout the ages, from the Third Dynasty to the New Kingdom, where it probably had some deep religious significance, especially in regards to divine protection. Since the Egyptians did not believe in chance, and the game involved a random factor akin to a roll of dice, the winning player was believed to be under the protection of the most powerful deities. The game was played on a thirty-square board, arranged in three rows of ten squares each, and with a certain number of pawns for each player, wooden examples of which have been found in several tombs. Five squares were marked with special symbols, and probably represented special rules pertaining to the particular square. Unfortunately, the rules of Senet are, as of yet, unknown, though two researchers have proposed different rulesets for the game.

Still, even without a full description, it appears that Senet may have been the ancestor of **Backgammon**, as a complete information game with tokens following a linear path and an element of chance.

Another candidate for the title of oldest game is **Oware**, another complete information game, mainly played in West Africa and the Caribbean. It is part of a broader family of games called *mancala*, which could be roughly translated as Count-and-Capture. These games involve tokens called *seeds*, and usually consist in two phases: sowing and capturing. In the most widely played Oware variant, each player has six pits, each containing four seeds. Players take turns sowing, that is, picking up the seeds from one of their pits and distributing them into the next pits (including the opponent's own pits) in a counter-clockwise fashion. After the sowing, the capturing stage ensues, wherein if the last pit to receive a seed has exactly two or three seeds in it, those are captured by the player who performed the move, and in that case the check is repeated on the second last pit, and so on. As it is readily seen, this game can be solved by building a classical game tree, having a branch factor of 6. In fact, in 2002, two Dutch researchers claimed they had solved Oware, building a tree numbering about one trillion nodes, even though they did not consider the most popular variant of the game.

The third candidate, and probably the one that enjoys the most popularity nowadays, is **Go**, born in China between 2000 BC and 200 BC. It is a strategic, complete information board game whose corresponding tree is so large that it is not believed to be solvable through computer brute-forcing. Legend has it that Go was invented as a teaching tool to educate an Emperor's son to the ways of balance, justice and rational thinking, but it probably found other, more mystical uses, as a means of divination, especially in relation to flood prediction and control. The strategic depth in Go is unmatched by most other board games, even if the basic ruleset is quite simple, with Black and White taking turns placing stones of their color on the intersections of 19 by 19 grid. Connected stones of the same color form *chains*, and the free intersections next to the stones of a chain are called its *liberties*. A chain that loses all of its liberties (thus being surrounded by the opponent) is removed from the game and adds to the opponent's score. Since this can be likened to a fight between two military forces, Go could qualify as the first *wargame* in recorded history.

2.2 Wargames

There are many more ancient games that may have been, at some point, used as an abstraction for conflict, but often, the same game would change rules over the decades and centuries, and the reference to pawns, generals and territory was mostly symbolical. The first board game to sport a consistent military background is arguably the Indian game of **Chaturanga**, even though there is no physical historical evidence about it. Considered to be the ancestor of **chess** and other chess-like games, including **Jangki** (Korean chess), **Makruk** (Thai chess), **Shogi** (Japanese chess), and **Xiangqi** (Chinese chess), it was allegedly played in the seventh century AD, and its pieces were modeled after the actual Indian military, with the general and his advisor, slowly-advancing infantry, knights for flanking enemy lines, fast but difficult to maneuver chariots (rooks), and devastating war elephants (bishops). Its rules were very similar to those of chess, except that, instead of *checks*, the simpler mechanism of capturing the enemy King was used. Aside from a few differences in the rules for moving some pieces, however, it was basically the game of chess. Chess itself did not have a consistent ruleset until much later, when standardized rules became necessary in order to host tournaments and competitions.

2.2.1 Kriegspiel

Interestingly, while chess and other games of its kind were believed to hone one's strategic skills and made a common pastime among generals and men of war, it took many century before someone realized that games could be more than just a metaphor of war, but the war itself. It was the highly advanced Prussian military that first understood the potential of a realistic war simulation in the training of their officers and tacticians, but in order to provide such benefits, the game would have to evolve beyond the simplicity of a chess-like game, most importantly abandoning the realm of **perfect information**. In a situation of conflict, an assumption such as knowing the position of the enemy as well every square inch of the battlefield was unrealistic, and so it was that **Kriegspiel** was born.

A game played on three identical boards representing actual territory, one

for each player and one for the referee, the only one with a perfect knowledge of all units on the field. The units, an evolution of chess pieces, had the same features and capabilities as their real-life counterparts, and the players would communicate their moves to the referees, who would update his board accordingly, decide the outcome of combat, and notify the players about new happenings and new units in their line of sight. It is commonly believed that Kriegspiel granted an important edge to the generals who employed it in several battles throughout the last few centuries. The most detailed Kriegspiel might number literally hundreds of different unit types, each with its own special rules, including camouflaged troops, scouts, logistics, engineers, HQ, and more.

Imperfect information added a whole new dimension to the gaming world, because it expanded the strategy beyond the immediate game mechanics, forcing the players to evaluate risks, devise ways to acquire information and hide their own from the opponent. It is no surprise that most of the newer wargames are based on imperfect information settings, with limited knowledge of the opponent's territory or resources. The game this work deals with, also called Kriegspiel, is actually a chess variant invented by Michael Henry Temple at the end of the XIX century as the answer of chess to modern wargaming. As a variant where the player cannot see their opponent's pieces, it is considerably difficult to play well, and yet luck seems to be much less important than it might appear. Chess-based Kriegspiel will be treated in great depth in the remainder of this work.

Chapter 3

Game theory notions

This chapter summarizes basic game theory notions and concepts that will be needed in the remainder of this thesis. Since the following are all extremely well-known results, exposition will be concise and to the point.

3.1 Introduction

Game theory is a branch of applied mathematics that studies situations where intelligent and rational agents (players) need to choose actions that maximize their returns. The first implicit mention of game theory dates back to 1713, when James Waldegrave wrote a letter in which he solved a version of the card game *Le Her* with a *mixed strategy* based on *minimax*. A century later, in 1838, Antoine Augustin Cournot published *Researches into the Mathematical Principles of the Theory of Wealth*, which dealt with the dynamics of an economic duopoly seen as a game.

It can be safely assumed that the first real formulation of this very young discipline in its current form is due to John von Neumann and Oskar Morgenstern (*Theory of Games and Economic Behavior*) [26]. The theory was developed in the years of the Cold War, and one of the concepts it modelled was that of mutual assured destruction (a model providing a possible explanation for the race to nuclear weapons as a strategy for avoiding the worst case scenario, the end of the world). This fact alone proves that game theory is an extremely powerful science, which aims to model very diverse aspects of reality that include animal, human and social behavior, ethical

choices, evolution processes and more. In fact, games such as the *prisoner's dilemma*, where selfish behavior penalizes everyone, have been the ground for philosophical thought.

Game theory mainly differs from decision theory in that it assumes the presence of more than one player, and the optimal strategy does not only depend on the environment, but also on the opponents' choices. Hence we can distinguish between *cooperative* and *non-cooperative* games, depending on whether players may form coalitions to pursue their mutual interests, or they base their decisions uniquely on themselves. Von Neumann dealt primarily with cooperative games, whereas John Nash studied the non-cooperative case, also arguing that cooperative games can be translated into an equivalent non-cooperative form (*Nash programme*).

Major game theory advances include John Nash's formulation of the cornerstone theory of *Nash equilibria* in 1950 (situations wherein no player can further increase his payoff by unilaterally changing his strategy) [28], which provides an optimal strategy for multiplayer games, and the following year, his *Nash bargaining solution* for coalition-based games. His theories were later extended and refined. For the purpose of this work, it is important to remember H. W. Kuhn's contribution [21] in first researching *imperfect information* games, where players do not know what moves have been played by their opponents so far; many games being the object of research belong to this category, as they provide a better model of real-world situations, where perfect information is the exception rather than the rule (perfect information is not the same thing as *complete information*; in a complete information game, each player knows the strategies and payoffs of every player, but not necessarily their actions; incomplete games are also called *Bayesian games*).

In the following years, the previous theories were refined and adapted to a variety of fields. Lloyd Shapley introduced the *core* (an economy-oriented version of Nash equilibrium based on resource allocation) and *Shapley's value* (a theory for the fair allocation of gains from a coalition to its individual components). The concept of *subgame perfect equilibria* is due to Reihard Selten (1965), and it is an extension of Nash equilibria to subgames of an existing games. He also introduced *trampling hand perfection*, a notion of equilibrium based on perturbed games (games where players cannot have pure strategies, that is, their hands will 'tremble' every once in a while, leading them to a

different choice). Also, the notions of *evolutionary stable strategy* (equilibrium through evolution, designed for biology as Nash equilibrium assumes that players act rationally, whereas evolution replaces reason in a biological context), *correlated equilibria* for Bayesian games, and *common knowledge* (knowing that everyone knows, knowing that everyone knows that everyone knows, etc.) were established.

Game theory-related research was the basis for awarding the Nobel Memorial Prize in Economics to John Harsanyi, John Forbes Nash and Reinhard Selten (1995, equilibria in non-cooperative games) and Robert Aumann and Thomas Schelling (2005, game theory analysis of conflict and cooperation).

For an in-depth textbook on general game theory, we refer to [14] and [30], which is also freely available online. As for historical texts, aside from the aforementioned [26] and [28], we recall [33] for a treatment of game theory from an evolutionary standpoint for application in biology.

3.2 Concepts

3.2.1 Game

A *game* is a mathematical entity consisting of a set of *players*, a set of moves (*strategies*) available to the players, and a set of *payoffs* specified for each combination of strategies. Generally, players are assumed to be intelligent and rational, that is consistently pursuing the goal of maximizing their own payoffs. In the case of repeated games, it is assumed that rational players will want to maximize their *average* payoff; this is often not the case with human players, and indeed, humans often play irrationally according to a game theoretical definition (in several documented cases, for example, humans rarely if ever play Nash equilibria).

3.2.2 Pure and mixed strategies

A player is said to follow a *pure strategy* if he plays the same strategy each and every time. On the other hand, in a *mixed strategy* the player chooses among two or more moves according to some probability distribution ρ .

3.2.3 Zero sum games

In a zero sum game, for any possible outcome the sum of the players' payoffs is equal to zero. In other words, a player can only improve his or her payoff at the expense of some other player.

Many classical games such as chess are zero sum, in that there is a winner and a loser. In a non-zero sum game the actual payoffs are entirely depending on the particular outcome; for example, in the *prisoner's dilemma*, either player (prisoner) can either cooperate or defect (betray). If one player cooperates and the other defects, the cooperator gets the worst payoff and the traitor gets the best payoff; if both defect, both get low payoff; if they both cooperate, both get good payoff.

Non-zero sum games can always be translated into zero sum games by simply adding a dummy player whose payoffs balance the players' net payoffs.

In the case of two-player zero sum games, it is sufficient to give the payoff function for one player; the second player will simply use its negation.

3.2.4 Equalizing strategy

A strategy for a zero sum game that guarantees the same average payoff independent of the opponent's strategy is called an *equalizing strategy*.

The strategy that guarantees the highest average payoff independent of the opponent's strategy in a two-player zero sum game is an *optimal (minimax) strategy*, and its average payoff is called *value* of the game; it represents the average payoff for the player if both players play at their best (one player cannot win more than the value, the other cannot lose more). Every finite game of this type has a value.

John von Neumann proved this result, known as *Minimax theorem* in 1928, noting that an optimal strategy always exists in this kind of games and can be found by solving a set of simultaneous equations to find ρ_1, \dots, ρ_k , distribution coefficients for the optimal mixed strategy that plays moves m_1, \dots, m_k . A minimax strategy exists for both players.

	P2 Cooperate	P2 Defect
P1 Cooperate	3, 3	0, 5
P1 Defect	5, 0	1, 1

Table 3.1: Prisoner’s dilemma, normal form.

3.2.5 Normal form

A game in normal form is a complete listing of the players’ strategy spaces with the related payoff functions, where a strategy space is a set containing every possible sequence of actions for each part of the game. Describing a game in normal form is convenient for spotting dominated strategies (that is, strategies that will never be selected because always worse than others) and Nash equilibria.

Most often, normal form coincides with a *payoff matrix*, especially when there are only two players. Then, the strategy spaces for the first and second player are interpreted as the rows and columns of a matrix whose elements are couple of payoffs for the players if those strategies are chosen. For example, a possible payoff matrix for the aforementioned Prisoner’s dilemma would be as shown in Table 3.1.

Normal form makes it easier to intuitively spot equilibrium situations. For example, in the Prisoner’s dilemma, let us consider Player 2’s options. If Player 1 cooperates, it is in Player 2’ interest to defect so he will receive maximum payoff. And if Player 1 defects, then Player 2 would have to defect as well, in order to protect himself from the worst-case scenario. In either case, cooperating is a dominated strategy for Player 2, and since the game is *symmetric*, the same applies to Player 1. In fact, defect-defect is a Nash equilibrium for this game, even though it is not the choice that maximizes the total payoff for both players (actually, it minimizes that quantity), meaning that in the same game, a Nash equilibrium and a *Pareto optimum* may be opposites (there are many more interesting facts about the Prisoner’s dilemma, including that if the game is iterated an *unknown* number of times, then cooperation becomes crucial).

This representation assumes that players choose their moves *simultaneously*, or at least there is no way for them to know the opponent’s strategy

before they have chosen their own. For this reason, such games have *imperfect information*. Games where the players take turns choosing moves that are observable by the opponent (*sequential* games) can be represented in normal form by expanding the second player's strategy space with fictitious moves that include the opponent's possible previous moves as their premises. However, sequential games are often easier to represent in *extensive form*.

3.2.6 Extensive form

A game in extensive form is represented through a *tree* where every non-leaf node represents a choice by a player, and each leaf contain payoffs if such a sequence of choices occurs. Figure 3.1 (left) shows a simple game in extensive form where nodes marked with 1 or 2 stand for Player 1 or Player 2's moves, and Player 1 moves first. In the example, no matter what Player 1 chooses, Player 2 (if he acts rationally) will play R' as it maximizes his payoff. Knowing this, Player 1 will play L, which maximizes his own payoff.

While the tree form lends itself well to sequential games of perfect information, it can be extended to treat simultaneous or imperfect information games, as well, by connecting nodes with a dotted line (or enclosing them in a circle). In this way (see [21]), it is meant that the nodes belong to the same *information set*, that is, the player who moves next will not know exactly which node in the set represents the current situation. For example, in Figure 3.1 (right), Player 2 does not know whether Player 1 played L or R. Reasoning with perfect or imperfect information will, most of the time, change the equilibria for the same game. It is possible to define a perfect information game as a game in which every information set is a singleton, that is contains only one node.

The concept of *information set* as a set of strings (move sequences) the player cannot distinguish from one another, is especially crucial for the remainder of this work, when dealing with the game of Kriegspiel. Since in the game of chess there is a 1-1 correspondence between a (legal) move sequence and a possible chessboard layout, it follows that in chess with imperfect information (Kriegspiel), information sets are sets of chessboards. These will be abstracted through the notion of *metaposition*.

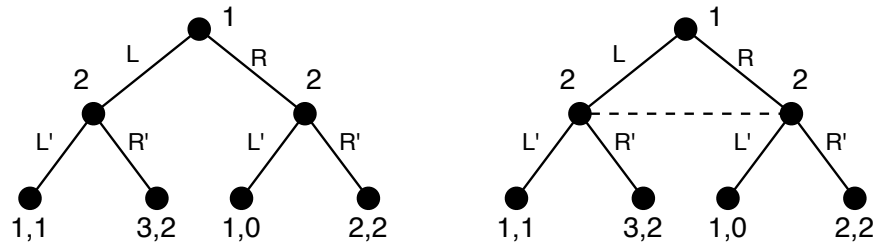


Figure 3.1: Extensive form with perfect or imperfect information.

3.2.7 Imperfect and incomplete information

As stated in the previous subsection, a game is said to have *imperfect information* if at least one of its *information sets* is not a singleton.

On the other hand, in an *incomplete information* game, the other players' strategies might not be necessarily unknown; however, players lack information about the *types* of the agents they are playing against, or about the outcome of their own possible strategies, or about some payoffs. For example, a potential employer looking for skilled employees through a series of interview does not know the other players' types (skilled or unskilled). As a consequence, the employer's payoff from selecting a certain candidate is not clear, either.

As a general rule, it is possible to translate a game of incomplete information into an equivalent one of *complete, but imperfect* information by introducing *Nature* as a player. Aspects of incomplete information are represented as Nature's choices following a given probability distribution. In the above example, Nature chooses whether an applicant is skilled or unskilled, and the payoffs for the employer in either case become uniquely determined. Of course, Nature is not a rational player and stochastic techniques are then used to calculate an optimal strategy (*Bayesian games*).

3.2.8 Zermelo's Theorem

Theorem. Let P1 and P2 be two players participating in a zero sum game with strategy space S . Let f_1 be P1's payoff function, and $f_2 = -f_1$ be P2's payoff function. Then there exist strategies s_1 for P1 and s_2 for P2, and an

outcome $V \in S$ such that

- no matter P2's strategy, if P1 plays s_1 , the outcome O will satisfy $f_1(O) \geq f_1(V)$. There is no better outcome than V for which the above holds;
- no matter P1's strategy, if P2 plays s_2 , the outcome O will satisfy $f_2(O) \geq f_2(V)$. There is no better outcome than V for which the above holds.

The outcome V is the value of the game.

Proof. The Theorem can be proved by induction. First, let us consider a game in extensive form and note that we can *prune* the resulting game tree; if T_1 is the set of nodes belonging to P1 with maximum depth, from T_1 emerge leaves with different outcomes. If P1 is rational, he can certainly choose, for every node, the leaf that maximizes f_1 . We can therefore prune the last level of the tree, replacing the decision node with P1's favorite outcome. The same reasoning applies to P2, allowing us to reduce the depth of the Kuhn tree by one upon every iteration.

This being said, induction follows easily. The theorem obviously holds for one-move deep games, as P1 will simply move to his favorite outcome, which is V . Now let us suppose that the Theorem holds for $(k-1)$ -deep trees (that is, the longest sequence of moves in the game is $k-1$): we must prove that it also holds for k moves. If T' is the pruned tree for a k -move game T , its depth is certainly $k-1$. But by assumption, the game associated with T' has some value X . And because of the way trees are pruned, if P1 can guarantee himself a payoff of X in T' , he can do the same in T (the same applies to P2). Since P1 and P2 have, by assumption, strategies that guarantee at least X in T' , it follows that they also have one for T , and the Theorem is proved.

Corollary. Concerning the game of chess, only one of the following applies.

- White can always win.
- White and Black can always draw.
- Black can always win.

Proof. Follows trivially from Zermelo's Theorem by simply representing White's victory as 1, Black's victory as -1 and draw as 0. If these are the only values the payoff functions can take, then the value of the game will be one of the above. Depending on which one it is, one of the three above statements holds.

This corollary is extremely important and applies not only to chess, but to any two-player zero sum perfect information game, including checkers, go and Othello. Its meaning is that there is a strategy that can assuredly obtain the value of the game independently of the opponent's moves; that is, there exists a *perfect game*. If both players play their perfect game, that is they play their best possible strategies throughout the game, then the outcome is necessarily the value of the game.

This means that games like chess could be solved trivially by simply exploring their Kuhn trees. In fact, chess is no different from tic-tac-toe in this regard. The only reason chess is still interesting nowadays whereas tic-tac-toe is but children's play lies uniquely in the relative sizes of their game trees. Tic-tac-toe can be fully explored and its value computed; it is 0, meaning that neither player can force a win if they are both playing optimally. The Kuhn tree of chess (and checkers, go, etc.) escapes any attempt at representation due to its immense size, far beyond the capacity of any computer system to handle. But there certainly exists a strategy that guarantees at least one of the players to never, ever lose, and possibly to always win.

As an important addendum, it should be noted that the Theorem can only be applied to perfect information games. Any other case does not admit a perfect strategy or a perfect game.

Chapter 4

Kriegspiel

4.1 Overview

Kriegspiel is a chess variant in which the player cannot see their opponent's pieces and moves. The game is played on three chessboards, one for each player and one for the referee (**umpire**), the only one possessing complete information on the state of the game. The players are given a full set of pieces of their opponent's color, and are free to place them anywhere on their chessboards to aid their memory or visualize their guesses on the opponent's deployment, but this has no effect on the game itself.

When a player is requested to move, he or she will announce the move to the umpire (and only the umpire; there should be no direct interaction between the players in Kriegspiel). The umpire will then check on his chessboard whether the attempt is legal.

- If the move is illegal, he will say “illegal” and ask the player to choose another move instead.
- The referee should say “nonsense” if the move was trivially illegal even on the player's board, for instance if he were to try to move a knight like a rook; this to prevent one player to trick the other with a large number of illegal moves in order to mislead the opponent about his actual resources.
- If the move is legal, the umpire will be silent, or say something along the lines of “Black moved” or “White to move”.

In addition, the umpire will notify both players in the following cases.

- If a piece is captured (specifying where, and possibly some information on the captured piece depending on the rule variant, but never will he say anything on the nature of the offending piece).
- If a player's King is in check, he will specify the direction (or directions, if it is a double check) from which the check is being given.
 - Rank check.
 - File check.
 - Long diagonal check (from the king's point of view).
 - Short diagonal check (from the king's point of view).
 - Knight check.

The umpire's messages are therefore laconic, and as a rule, everything he says can be heard by both players, even though they will draw different information out of them. In Kriegspiel, you know what you know, but you do not know what your opponent knows.

Unfortunately, Kriegspiel is hardly a standardized game, which is both a cause and a consequence of its scarce popularity throughout the XX century, at least until more recent years. This variant has, itself, several variants that, while keeping the original spirit of the game intact, differ slightly in the way the umpire communicates his messages, and the amount of information contained therein.

4.2 Rule variants

Chess Kriegspiel was born in England, and the oldest ruleset is referred to as 'English rules'. The rules enforced at The Gambit (a famous chess club in London), an example of English rules, are listed in Appendix A. It can be said that the spirit of the English ruleset is the most akin to that of the old Kriegspiel used to simulate war. It makes for slower, but subtler gameplay in which every action is to be carefully considered, and information is expensive

to acquire. In fact, the rules are designed to force the player to pay a price for each piece of information he gets.

The most notable rule here is called '*Are there any?*', a sentence which has become quite famous (Kriegspiel is known as '*Any?*' in the Netherlands). It is also the name of a collection of Kriegspiel problems by G.F. Anderson; a problem from that book will be examined in the next section. This rule allows the player to ask the umpire, before his move, whether he has any possible *pawn tries*, that is, legal capturing moves with his pawns. If there is none, the umpire will say 'No'; otherwise he will say 'Try'. In the latter case, the player must try at least one capture with his pawns. If the try is unsuccessful, he is not forced to try another pawn capture. In this way, the player *pays* for the information he has been given, possibly losing his freedom to choose. Also, the English rules do not specify whether a captured piece is a pawn or not.

The second important ruleset is due to J.K. Wilkins, an American mathematician (Kriegspiel has always been most popular in anglosaxon countries). He directed the RAND Institute after the Second World War and introduced Kriegspiel into the Institute as a means of training in the analysis of war scenarios (RAND being a large think tank with the goal of providing advice to the government on many topics, including the new cold war). This ruleset is known as RAND rules and is listed as Appendix B. RAND games are usually faster than games played under the English rules.

There is an additional American ruleset that lies halfway between the English and RAND rules, and it is called '*Cincinnati style*', listed in Appendix C. This ruleset forms the basis for variant wild 16 (Kriegspiel) on the Internet Chess Club, whose actual rules are described in Appendix D. In these rulesets, pawn tries are automatically announced before every move, with no try being forced upon the player. Since most Kriegspiel games are now played on the ICC, Cincinnati style rules are the most likely candidate for standardization in the event of official competitions.

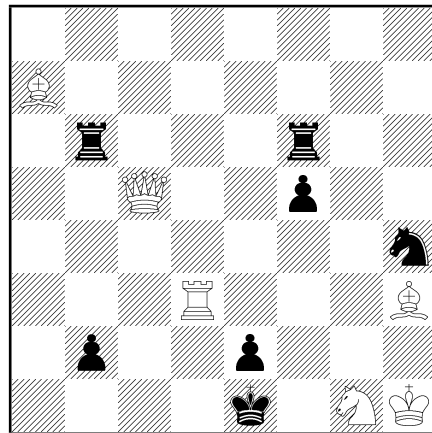
It should be noted that, in many situations and most scientific literature on Kriegspiel (such as optimal endgame strategies and algorithms), the ruleset of choice is irrelevant. Many Kriegspiel problems can also be solved under more than one ruleset.

4.3 Kriegspiel: gameplay samples

In order to better convey the differences between Kriegspiel and normal chess, it may be helpful to show and comment on some actual play examples. Instead of showing snippets from real games, it will prove more immediate to reason on custom Kriegspiel *problems*. Problems exist in Kriegspiel in the same way they do in orthodox chess, with the basic difference that the initial position of the enemy piece or pieces may or may not be revealed to the solver, whose goal is to provide a sequence of steps and move attempts that will guarantee checkmate within the required deadline (most often, one or two moves). Therefore, the solver is permitted to express each move as a sequence of attempts, where attempt a_n will be made if and only if all the attempts a_1, \dots, a_{n-1} turn out to be illegal.

There are a few sources dealing with Kriegspiel problems. The problems we are going to discuss are included in [7].

4.3.1 Problem 1



White mates in two moves (G.F. Anderson 1959 - Are there any?, 3).

This problem is the work of Gerald Frank Anderson (1898-1983), English diplomat and first important author of Kriegspiel problems.

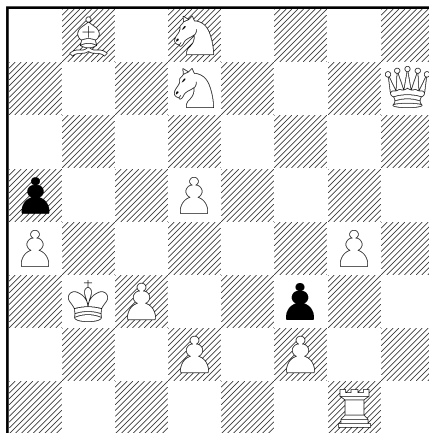
White must be able to checkmate on his second move, no matter how Black replies to his first. We take notice that, with White's rook and bishop keeping the enemy king in place, capturing the pawn in e2 would do the trick, which can indeed be done in two moves. However, what shall White do?

If he goes with Qe3, for instance, he might not be able to mate within the requirements set for the problem (if Black moves either rook to e6, the queen will be captured following Qxe2).

Therefore, it appears that the correct move for White in this situation is Qe7. In this way, upon trying —Qxe2—, if the attempt succeeds, Black is checkmated. If the umpire answers 'No', it means one of the rooks has been moved, and White has to find out which one. White can still mate with Qxh4 or Qb4. However, if the latter move is chosen first, there is a possibility that the queen will be captured if the rook is still on b6. Therefore, White will play Qxh4 first, and win if successful. If the umpire says 'No' again, it means the f6 rook is still in place; White will then conclude that it is safe to play Qb4, which is a guaranteed win. In other words,

1. Qe7
2. Qxe2; Qxh4; Qb4.

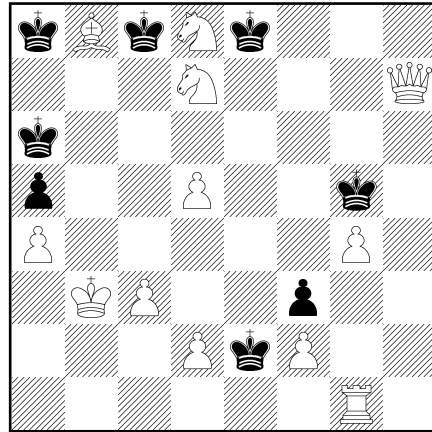
4.3.2 Problem 2



White mates in two (T. R. Dawson, 3190 *Fairy Chess Review*, 1938). Whereabouts of Black king are unknown.

This position is interesting because it might actually occur during a Kriegspiel game. The White player may have previously found the locations of the two pawns by pushing with a5 and f3, and failing; then, counting the amount of captured material, he realized that Black has no other pieces left. The task of locating the king begins, so that White can come up with a

certain mating sequence. First, we mark every square that is not threatened with a Black king, obtaining the following result.



First of all, we notice that a few of Black's tentative positions are very dangerous for White, namely a6, a8 and g5, as a stalemate can occur on the next move unless he takes action. It can be seen that only one move can prevent stalemate on any of those squares, and that move is Nf6. What happens next is totally dependent on the umpire's message. One of three things will happen:

- Knight check, Black moves with silent umpire. Then the king's position must have been e8, and it has been moved to f8. While will play 2. Qf7# and win.
- The king captures a piece on one of the following squares: b8, d8, f6, d2, f2. The checkmating move depends on which piece was captured: Qb7 (for b8), Qd7 (d8), Qf7 (f6), Qc2 (d2), Qh2 (f2).
- The king moves quietly (only possible from a6 to b6). Then it is just Qa7 and it is over.

4.4 A notation for Kriegspiel games

In order to record the transcript of a chess game, there exists a file format specification called PGN (Portable Game Notation). This well-known format

was designed to be both easily read by a human and easily parsed by a computer program, and consists of a series of tag-data pairs, some of which are mandatory, as well as a sequence of moves given in standard Algebraic Chess Notation. Actually, there are two formats to PGN, one being the relatively more lax “import” format, which can correctly parse human-created PGN files, and a strict “export” format, which a computer can generate to best adhere to the standard. A more thorough description is given in [17] and the full format specification can be found in [9].

PGN can be used to record chess variants as it supports the “Variant” tag (not to be confused with the “Variation” tag, which denotes a peculiar variation on a given chess opening). While some variants are harder to encode into PGN due to radical rule changes with respect to orthodox chess (because of different piece types, multiple turns, etc.), Kriegspiel is virtually identical to chess as far as the final product is concerned—that is, the resulting game is a full-fledged game of chess that can be exported as PGN without any modification.

However, it is clear that a Kriegspiel game recorded in this way loses a significant part of its usefulness, only registering events from the umpire’s perspective. In fact, PGN is not adequate for use in Kriegspiel *because it will not record illegal moves*. While what the umpire says after each legal move is easily figured out by replaying the game, all information about illegal moves is forever lost, and the rationale that motivated those illegal moves is gone with it. The legal move recorded on the PGN transcript might be a second, third, or fourth choice, and there would be no way to tell it from what the player really viewed as the “best” move in that situation, which is just as important, if not more, as knowing the best legal move.

Fortunately, a simple workaround to this problem is to insert the missing information into a PGN comment. PGN allows for comment to be included at any point in a file by typing text inside curly braces, so that, for example, a human can comment on a given move halfway through the game. A normal PGN parser will merely ignore them and hence interpret our Kriegspiel game as a plain chess game. There have been a few proposals in this direction; the extended PGN format that will be used in the remainder of this work, which is also the format output by the Darkboard engine for its transcripts, originates from the Berkeley University Kriegspiel project [19]. Minor modifications

have been applied to fit our purpose, because the original specification was not intended for use with the Internet Chess Club ruleset.

The extended PGN format is a well-formed PGN file, consisting of a preamble containing square bracket-delimited tag-value pairs, seven of which are mandatory, in this order (**Event**, name of event or tournament; **Site**, location of the event; **Date**, day when the event took place; **Round**, round number for multi-round events; **White**, name of White player; **Black**, name of Black player; **Result**, outcome of the game, which can be one of the following 1-0, 0-1, 1/2-1/2 or * if the game is unfinished; in all cases but the last the outcome is also appended at the end of the file).

The remainder of the file is called *movetext* and lists the moves in the following way.

```

1.  e4   {:}           f5   {P1:}
2.  exf5 {Xf5:exd5}  h6   {:}
...  ...   ...           ...   ...

```

Each move is preceded by its sequence number, and then listed as usual with the White ply coming first. All the additional information is stored inside the curly braces, according to the format

$$\{ \textit{umpire-info} : \textit{illegal-move-list} \},$$

where both *umpire-info* and *illegal-move-list* are comma-separated lists. The latter is easily explained as the list of moves attempted by the player and found to be illegal by the umpire before the first legal move was tried. Therefore, in the above example, White's second move has a failed attempt in which he tried to play exd5. Finding that move impossible, and knowing that Black could not have moved any piece far enough to reach any other pawn, thus causing a pawn try, White realizes that exf5 will capture something.

The fact that White had a pawn try in the first place is conveyed by the *umpire-info* field of Black's previous move. While the *illegal-move-list* tells of what happens before the legal move, the *umpire-info* logically follows the move, coding any umpire message for the players. This is not technically necessary as umpire messages are deterministic; the feature is meant to help a human reader rather than a computer parser. Allowed codes include:

- **P x** , where x is the number of available pawn tries in the next move (for the opponent to take advantage of). If the Kriegspiel variant does not allow pawn tries, this code will simply never appear, or can be replaced with **A x** to mean that a player asked “Are there any?” and the umpire gave the answer x .
- **C x** , where x is an uppercase letter, means that, following this move, the opponent’s King is in check. Legal character codes are **R** (rank check), **F** (file check), **L** (long diagonal check), **S** (short diagonal check) and **N** (knight check). In the event of a double check, two check entries appear in the list rather than two letters following the **C** code.
- **Xa1**, where **a1** represent any square on the chessboard, means that a capture took place on the given square. This is, in most cases, the same square where the current player moved his piece to, the only exception being *en passant* captures.

While this format may appear very straightforward, there are a few interesting points to consider. Firstly, a Kriegspiel PGN file might not contain all the information described above. Only the umpire knows the totality of it, and depending on who is recording the transcript, some information will be missing. The file format authors call these different perspectives *filtered* or *unfiltered* games.

Unfiltered games are typically created by the umpire, and contain everything, including moves and attempts for either player. Filtered games, on the other hand, represent one player’s perspective and can, as such, replace what is not known with the uncertainty placeholder **??**. Also, the authors replace the illegal move list for the opponent with merely a number representing the number of illegal moves tried by the opponent himself. However, depending on the ruleset, even this datum might not be available (e.g. on the Internet Chess Club). An example of filtered game could be:

```

1. e4      {:}    ??  {:}
2. e5      {:}    ??  {Xe5:}
3. Qh5+    {CS:} ??  {:}
... ..    ...    ...  ...

```

To tell a filtered game from an unfiltered one, a **Filtered** tag may be added in the preamble. The authors consider the following legal values for it: *no* (unfiltered), *white*, *black* (player from whose perspective the game is written). In the development of the present work, however, it became apparent that one more class of PGN transcripts should be considered, namely **semi-filtered** games (tag value *semi-White* or *semi-Black*). The Darkboard engine outputs semi-filtered games when possible.

A semi-filtered game is viewed from one player's perspective, but also contains information about the opponent's moves; only the opponent's illegal moves are not disclosed. On the Internet Chess Club, it is possible to obtain transcripts for a Kriegspiel game (as a regular PGN) after it is finished, and a player can note down his own illegal moves. By combining these data, a program can compute a semi-filtered PGN file.

Another interesting point which needs to be raised is a purely technical one, and concerns notation. Algebraic notation aims to shrink moves down to a more compact format whenever possible; "Rg4" is used instead of "Rh4g4" unless the starting square (or just one coordinate of it) is required to disambiguate the move, for example if two rooks are on the same rank. When one can reason with complete information, disambiguation is a trivial matter; however, *the presence of hidden enemy pieces makes moves that are illegal from the umpire's perspective, potentially legal*. Therefore, a shortened move transcript that is perfectly sound for the umpire, might seem ambiguous to the player who acts on imperfect information. It makes sense to ask oneself which kind of disambiguation rules to apply when recording a Kriegspiel game.

To this end, it is useful to remember that Kriegspiel was born to be played on three chessboards. When deciding whether a move can be shortened, *one should refer to the chessboard that is telling the move*. In the case of unfiltered or semi-filtered games, it is the umpire that recounts the story of the game, and disambiguation should take place according to the situation on the umpire's chessboard. For filtered games, necessarily it is the filtering player's chessboard that should be used as reference (the other two chessboards are obviously unknown). Since the moves that appear to a player as legal are a superset of the actually legal moves, it follows that filtered games have less compact notation than unfiltered or semi-filtered games as more

disambiguation will be necessary.

Illegal moves, however, work differently. Those are always based on the chessboard of the player who tried them – being illegal, they do not even have a valid transcript on the umpire’s board. Therefore, even in unfiltered or semi-filtered games, illegal White moves should refer to the incomplete White chessboard, and the same applies to Black.

Chapter 5

Kriegspiel on the Internet Chess Club

Kriegspiel (actually, all the games that, in history, were given the name of Kriegspiel) is a troublesome game to play in the real world. The setting itself may cause problems, as it takes up to three rooms, three chessboards and someone willing to take up the boring role of the umpire. It is also a slow game to play, as most time is spent communicating with the umpire and listening to his messages. More than its unusual features, it is probably its tedious flow that kept Kriegspiel from becoming a truly popular game. In truth, the same can be said of the original war simulation bearing the same name; and as a matter of fact, warsim-Kriegspiel variants were soon born where the umpire was no longer a mere dice-rolling executive of impossibly large rulebooks, but an entity with true decisional power, replacing the often awkward rules with his own experience. There is no room for interpretation when it comes to chess-Kriegspiel, though; but on the other hand, when a task is exceedingly boring, chances are a computer can handle it effortlessly. The computer makes a splendid Kriegspiel umpire and completely erases the need for a third chessboard.

In fact, automated Kriegspiel referees for use over a computer network actually predate today's large-scale Internet. Some early Kriegspiel referees are described in [6, 34, 35], their publication dates ranging from 1967 to 1975. Kriegspiel was therefore one of the first games to be transposed to an electronic version (two of the authors of the second program, Wetherell and Buckholtz, described in [24] how they wrote it in FORTRAN and included

several advanced features for its age, such as one of the first Internet-based chat systems). There is also another C program that was developed in the '80s with the same purpose.

5.1 The Internet Chess Club

The Internet Chess Club (ICC) [16] is, today, where most Kriegspiel games are played. The ICC is a large chess community comprising thousands of members, where it is possible to play chess as well as chess variants (called “wild” chess). A few dozens of games of Kriegspiel are probably played on the ICC every day, and it is possible to meet expert human players and weaker computer players. The ICC provides a long list of features to its users, including a very strong community whose “social capital” has even been the subject of research [15], several ranking systems, game recording, adjudication, automatic matching based on parameters, and the ability to observe and replay recent games by any (non-guest) player. This last point makes it possible for a demon process to log on periodically to download any new games of a given type, and this is how it was possible to assemble a sizeable database of Kriegspiel games.

It is to be noted that the ICC follows its own ruleset as far as Kriegspiel is concerned. The Kriegspiel variant is known as “wild 16” and is played *Cincinnati-style*. This ruleset, somewhat of a compromise between the strict English rules and the faster American ones, is designed for the fast-paced, highly competitive gameplay that characterizes the whole of the ICC experience – moreover, it was clearly implemented in such a way as to require as few changes as possible to the server’s framework. Notably, there is no interaction between player and umpire (no “Any?” rule, for example). This is replaced with a simple mention of the number of pawn tries each player has, before their turn.

The ICC framework provides a simplified ELO rating system wherein all chess variants contribute to the same score. For example, Kriegspiel, Atomic chess, and Loser’s chess all count towards the same ‘wild’ rating. Of course, this means that one’s rating is, by itself, useless in determining the player’s strength at Kriegspiel as the rating might have been acquired playing other perfect information variants of chess. Thankfully, registered users of the ICC

have access to a special bot, *Robokieb*, which keeps track of individual ratings for each wild variant type. Thanks to this bot it is possible to see that the average ELO rating for Kriegspiel on the ICC is around 1570 points, with the best human players standing at around 2100. Existing computer players and their ratings are: **G2K** (1921), **Krieg** (1748), **fark** (1682) and **phark** (1485).

Kriegspiel on the ICC presents a completely new set of challenges to the player, be it human or artificial. Because all games on the ICC are timed, to out-time one's opponent is often as good a strategy as any, and time for reasoning is a luxury. Moreover, rarely can one be certain that the move he is planning to perform will be legal, and multiple illegal moves imply more and more time lost. Thus a tradeoff must be sought between acquiring information via tries and testing moves and keeping one's clock safe. While, logically, it may make sense to try very advantageous moves (such as a promotion move), but with low probability of success, on each and every turn, this should be considered carefully in a timed environment, where it might just turn into waste (especially if there is lag!). Such a thought should also concern an advanced artificial player trying to play Kriegspiel with time control settings. It is easier in orthodox chess, where it is usually possible to cut the analysis at any moment if time is running short, and still obtain the best move found so far, while being certain that it will be accepted.

According to studies, the way people play chess on the Internet is also different from the traditional on-the-board behavior, where a percentage of players favor more aggressive play and a tendency to keep playing past the point where a traditional player would probably resign; in other words, to use the Net's automated environment and mask of anonymity to bypass the usual etiquette of chess playing. This may apply to Kriegspiel games played on the ICC, as well; but there is an opposite tendency to resign when a fully rational player would keep playing (in fact, a fully rational player would almost never resign in Kriegspiel, seeing as defeat can be seldom proved as inevitable).

5.1.1 A Kriegspiel game database

While Kriegspiel draws the vast majority of its rules from such a vastly explored game as chess, its nature is necessarily different. An uncountable amount of books and papers have been written on chess, developing a number of theories over the decades. Hundreds of openings exist and are known to master-level players and software alike, together with appropriate counters and follow-ups. Therefore, it makes perfect sense to ask oneself how much of this knowledge translates to Kriegspiel; in other words, how big an advantage, if any, would an expert chess player have when playing Kriegspiel? Do humans usually play Kriegspiel the same way they do chess? If so, does it benefit them? After all, it is known that humans often do not play Nash equilibria in imperfect information games.

The answer is far from straightforward, for several reasons. First, Kriegspiel is, to this day, not a particularly popular game, lacking official ratings, competitions and tournaments. The Internet Chess Club is arguably the premier place for playing Kriegspiel, with several games played on a daily basis, but it will be shown that even the users with the highest victory counts play in a sub-optimal way from a formal point of view. The amount of research and competition put in the game of chess simply does not exist in Kriegspiel. This, combined with the fact that the problem space in Kriegspiel is far greater than the one in chess (which is not small to begin with), means that results which are computable in chess, such as a large number of endings, are out of reach in Kriegspiel.

As discussed in the previous chapter, even when transcripts of Kriegspiel games are available, the information they contain is rarely complete; in fact, a Kriegspiel game is more than just the sequences of moves by the two players. The Internet Chess Club does record every game played on its servers, including chess variants, which can be downloaded as a standard PGN file, which is not adequate for representing a Kriegspiel game *because it will not record illegal moves*. While what the umpire says after each legal move is easily figured out by replaying the game, all information about illegal moves is forever lost, and the rationale that motivated those illegal moves is gone with it. The legal move recorded on the PGN transcript might be a second, third, or fourth choice, and there would be no way to tell it from what

the player really viewed as the “best” move in that situation. Hence, even though we have assembled and analyzed a database of Kriegspiel games from the Internet Chess Club, no amount of analysis will ever provide a totally accurate player behavior model.

With the aforementioned caveats in mind, it is still possible to infer substantial information from the available data. Over several years, a sizeable database of Kriegspiel games played on the ICC has been collected, mostly through automated agents or daemons and later filtered and refined to remove unrelated or uninformative material. The final database consists of 11,911 full games and is the largest collection of real-world Kriegspiel play we know of, with material dating as early as 1999 and as late as 2005, though most of it dates to 2002-2005. It is publically available at [18] as a single PGN file listing. This is not, in truth, the only existing Kriegspiel game database, for the authors in [31] have gathered a database of about 1,000 endgame situations, which they use to demonstrate their AND-OR search algorithm. On the other hand, the present database deals exclusively with competitive ICC games, whether played by humans or computers.

The level of play shown throughout the database is highly variable. Many people try out Kriegspiel ‘for the fun of it’, and resign as soon as they lose their queens – a mistake born out of chess mentality, as queens fall as easily as any piece in Kriegspiel, and *promoted* queens tend to be much more decisive than the starting one. There are, however, several players who show uniformly high play levels and consistently high victory ratios.

While data mining this collection of games does yield credible results, and only a small portion of them has been researched in the making of this work, care should be taken treating such a dishomogeneous mass of samples, hardly an ideal population. All of the matches were played online, and most importantly under drastically different time limits. A fair portion of Kriegspiel games on the ICC are played at a furious pace, with time control starting at 60 seconds and no increment – the game is necessarily over in 2 minutes at most. Kriegspiel-playing software, some of which does play on the ICC, tends to favor this style of play as the fast pacing tends to overshadow its lack of strategy and planning, as well as increase the chance of human blunders. Thankfully, the PGN headers include time setting information for each game, so it is possible to discriminate.

Outcome	Number	%
Black checkmated	2330	19.56
Black resigns	1454	12.21
Black forfeits on time	1355	11.38
White checkmated	2310	19.39
White resigns	1406	11.80
White forfeits on time	1428	11.99
White stalemated	390	3.27
Black stalemated	350	2.94
Not enough material	544	4.57
White out of time, Black cannot mate	86	0.72
Black out of time, White cannot mate	81	0.68
Game drawn by mutual agreement	117	0.98
Game drawn by the 50 move rule	38	0.32
Game drawn by repetition	22	0.18

Table 5.1: Outcome results for ICC Kriegspiel games.

5.2 A Statistical Analysis of ICC games

5.2.1 Outcome

In analyzing the corpus of games (with the above remarks in mind), it seems natural to start from the end, that is, find a breakdown of possible outcomes for the typical Kriegspiel game. Chess is a game where the player who gets to move first has a significant advantage (often measured as 60-40 for professional matches, enough to warrant different scores for White or Black victories), just like Risk. In both games the reason is that the first player can expand faster and take the initiative (or capture territories which in turn generate material) sooner. And interestingly, such advantage is nowhere to be seen in the imperfect-information version of either game, as stated in [7], among others.

The database provides a further confirmation of this fact, as can be seen in Table 5.1. White wins 5139 games (43.145% of the total) whereas Black wins 5144 (43.187%), with 1628 draws (13.668%). Actually, counting checkmates and resigned games only, White still appears to hold a tiny advantage, which we postulate to exist because many Kriegspiel players apply chess notions to

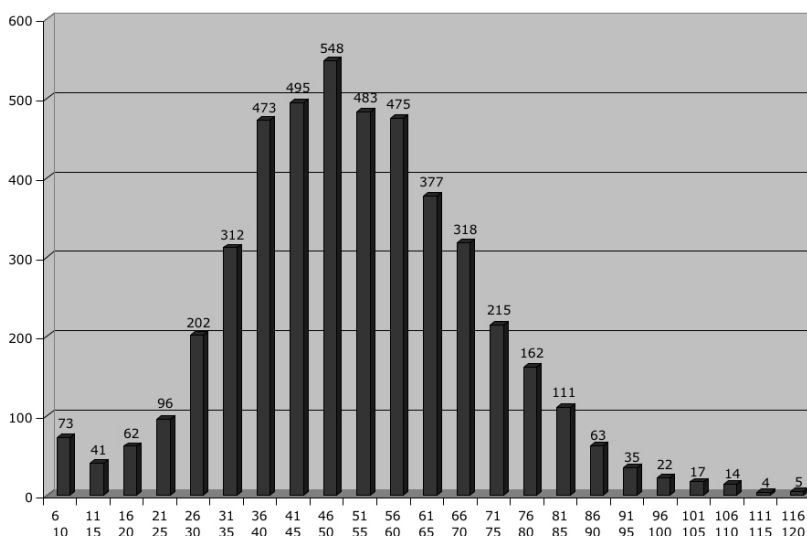


Figure 5.1: Duration table for checkmated games. About 20 games, whose durations ranged from 121 to 352, were left out.

their games, especially in regards to opening play. However, once we take forfeited games due to time control into consideration, the situation is, once again, levelled. Because White moves first, White also tends to run out of time first.

Unfortunately, resigned games are not very significant, because in many cases, players resign when they would still stand a fighting chance for a draw.

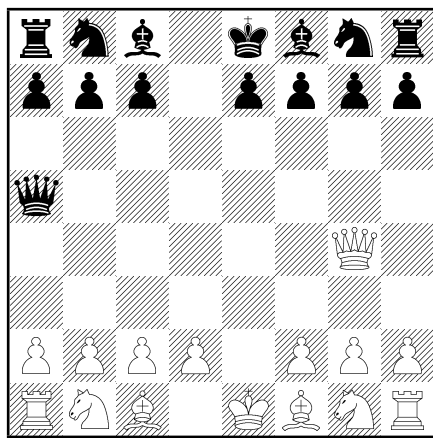
5.2.2 Game duration

A breakdown chart of the duration of Kriegspiel games on the ICC (limited to games that end with a checkmate) is given in Figure 5.1. Average duration is 52 moves (which decreases to 51 including non-checkmate games, due to players resigning early on), with a standard deviation of about 20. Most games last between 30 and 70 moves.

Possibly the most interesting fact about this chart concerns the very first columns of it, with a relatively high amount of checkmates taking place very early into the game (114 in the first 15 moves, out of 4640 games, or 2.46% of the total). There is a small but definite peak in checkmates at the beginning

of a game, as is also shown graphically. The reason for it is to be sought in the way many ICC players approach Kriegspiel with a chess mindset, often playing their openings as in orthodox chess, which may lead to early and unexpected mates.

Examples of the above are given in [7]. For instance, the move sequence 1. e4, d5 2. exd5 Qxd5, 3. Qg4, Qa5 leads to the following diagram.



Here, Black was expecting White to play 3. Nc3, but he moved the queen instead. At this point, White plays 4. Qxc8# and wins. An artificial player would need to avoid this kind of mistakes, as well.

5.2.3 Pseudomobility

By replaying every game, move by move, it is possible to compute the perceived number of pseudolegal moves that each player had at any given time. This number, as can be expected, is greater than the branching factor for the game of chess, which is typically between 30 and 35; however, it is not as large as one would be led to believe (for instance, [7] quotes about 60 pseudolegal moves). It is seen that, on average, players have about 40 possible moves to choose from in the middle game, as shown in Figure 5.2. Hence, in many cases, a player's *pseudomobility* will range between 30 and 50.

The most plausible explanation for this discrepancy is that Kriegspiel players do not set up their pieces in the same way a chess player would. In chess, mobility is an important factor; in fact, it is a typical component of many chess evaluation functions as it is seen that there exists a positive

connection between mobility and victory ratio. The same can be said of Kriegspiel, where such connection is even more developed in the endgame, but *excessive* mobility in the middle game seems to outweigh its own benefits. Many players tend to cluster their pieces together in order to take advantage from long capturing sequences – this is about the only way to acquire material advantage in the game, that is, having more pieces control the same square than the opponent does. On the other hand, scattering one’s pieces all over the chessboard will improve mobility at the expense of protection, which works better in the later stages of the game or when the player is already winning and hunting down the remnants of the enemy forces.

In the endgame, the difference in mobility between the winner and the loser is impressive, as shown in Figure 5.3. The loser is left with little more than the king (which accounts for about 8 mobility points), whereas the winner holds a 20-25 point advantage, which loosely corresponds to a queen. Interestingly, there is a smaller mobility peak around moves 65-85; it most likely represents pawn promotions, since it is only limited to the winner’s curve, though the loser experiences an increase in mobility later on. This can possibly be due to the loser’s own promotions and to the fact that most games have ended by this time, and the remaining ones are tougher battles for the winner.

5.2.4 Endgame material

Material balance throughout the recorded games is shown in Figure 5.4, confirming that the winner is usually decided in the first 50 moves, after which material is unlikely to change drastically. After move 60, the loser lacks the 5 material points required, as a rule of thumb, to win the game.

Investigating endgame positions also brings some unexpected results. All the games in the database have been parsed to the end, to review what final material was available to the players. The results for all won games are given in Table 5.2. It is seen that the KQK endgame is by far the most common, followed by KQQK, KQRK and and KRK. However, the comparison between the left column, listing all games that ended in a victory, and the right column, which is limited to checkmated games only, shows that not only are the ranks different, but the checkmate ratios differ wildly, as well.

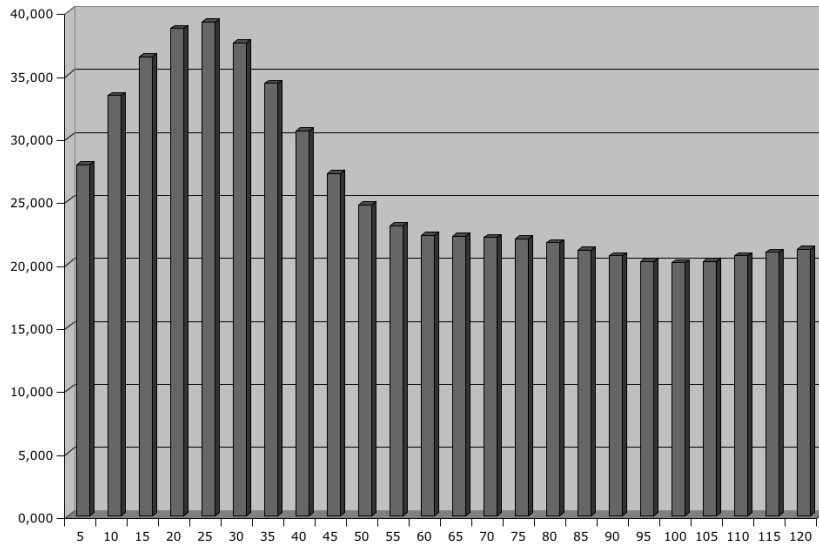


Figure 5.2: Pseudomobility chart (number of pseudolegal moves throughout the game, x axis representing move number).

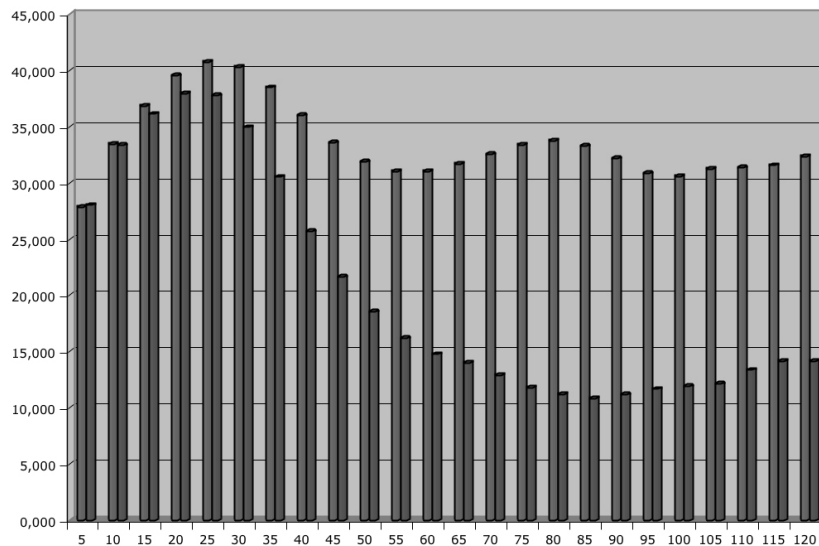


Figure 5.3: Pseudomobility chart for all games but draws; left columns represent the winning player's pseudomobility, right columns represent the losing player.

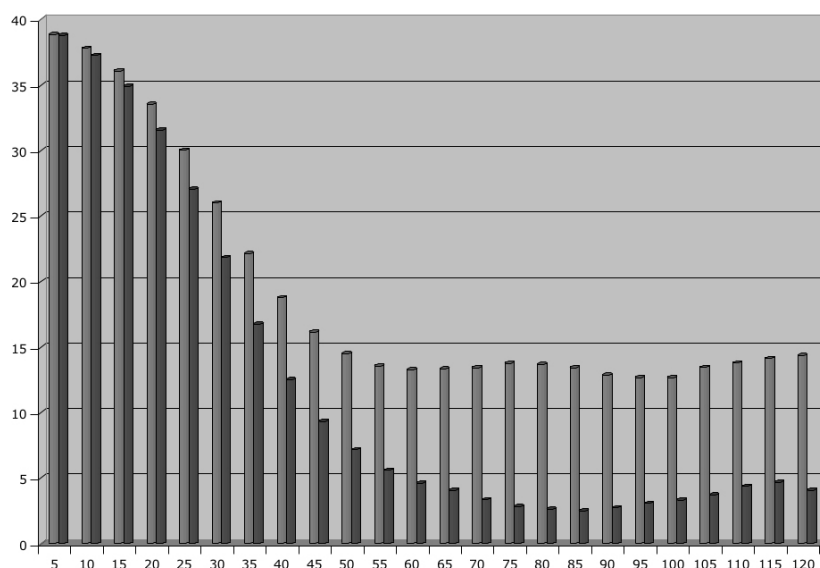


Figure 5.4: Material value chart for all games but draws, using conventional chess piece values; left columns represent the winning player's material, right columns represent the losing player.

In particular, there are 'easy' and 'hard' endgames for the average ICC player. An example of easy endgame is KQQK, which is extremely similar, if not identical, to its chess counterpart, where the king is simply pushed towards an edge by the queens, without the need for the allied king to even move at all. This endgame has a very high checkmate-to-victory ratio, about 80%. This is not always the case; for example, KQK is another matter entirely, with only 184 mates out of 347 victories, or 53%. Certainly, even an optimal KQK strategy takes longer than a KQQK one, so the losing player might lose interest and resign after a while; however, this certainly shows that many ICC players have trouble checkmating with the queen alone.

The data become even more significant when cross-compared with Table 5.3, which shows the most common positions that led to a draw. The vast majority of these games are either stalemated or stopped by the umpire, due to the 50 move rule or lack of mating material (which the ICC applies automatically to the following positions: KK, KBK, KNK, KNKN, KBKB, KBKN). KQK occupies the third place on the list, with 121 draws, a very

high number if compared with the 184 mates. Many players resigned in the KQK position, but among those who did not and played to the end, it turns out two out of five games were drawn. The players do not know the theory of Kriegspiel endgames, and thus play at a sub-optimal level. The KRK endgame is even worse in this respect (as it is a little harder to checkmate in this endgame), with 54 mates and 41 draws. KPK, which is 'almost' always won in theory (see Chapter 6), is drawn 44 times – and these numbers do not even account for the 273 positions with the kings being alone on the chessboard. Necessarily, all of these positions must be former KQK, KRK and KPK, where the winning player made a mistake and lost his piece. If we take these into account, the combined checkmates for the three endgames are actually *outnumbered* by the draws, meaning that Kriegspiel players on the ICC have a very sketchy notion of how these endgames are meant to be played.

As an aside, it should be noted that victory, and particularly checkmating, is extremely rare without a queen. In particular, endgames such as KBBK and KBNK ending in a mate are practically unheard of in the database (even though they can be won in Kriegspiel). Players seem to react to checks notifications in the endgame, and resign much more often when they realize the opponent has a queen. Also, checkmates rarely occur so long as the losing player has at least one piece left on the chessboard.

5.2.5 Opening play

Until the umpire's first non-silent message, a player is practically moving the dark, knowing nothing about the opponent. This period of limbo can last anywhere from one to even about 15 moves, and during that time, players tend to adhere to established opening schemes, some of which borrowed from chess, others unique to Kriegspiel. Among the top players, some follow their master plans more devoutly than others; for example, player Sting-R plays d2-d3 90.1% of the time, mostly as the very first move.

As a test, after selecting five players with the some of the highest game and victory counts in the database, their most frequent opening moves (played as their first 10 moves) have been extracted and used to recreate a hypothetical diagram, representing that player's ideal battle formation. Interestingly, one

Victories		Checkmates	
KQ vs. K	347	KQ vs. K	184
KQQ vs. K	180	KQQ vs. K	144
KQR vs. K	162	KQR vs. K	118
KR vs. K	138	KR vs. K	54
KQRP vs. K	92	KQRP vs. K	54
KQB vs. K	88	KQQB vs. K	47
KQP vs. K	87	KQB vs. K	45
KP vs. K	65	KQRB vs. K	41
KQN vs. K	62	KQQP vs. K	37
KQRB vs. K	60	KQRN vs. K	37
KQQB vs. K	57	KQN vs. K	33
KRP vs. K	56	KQQR vs. K	30
KQQP vs. K	49	KQRBP vs. K	29
KQRN vs. K	49	KQQN vs. K	27
KQRBP vs. K	43	KQQQ vs. K	25
KQRPP vs. K	40	KQRP vs. KP	24
KQRP vs. KP	38	KQRRBBNNPPPPPPPP vs. KQRRBBNNPPPPPP	19
KPP vs. K	38	KQRBN vs. K	19
KBP vs. K	37	KQRPP vs. K	19
KQBP vs. K	35	KQRBP vs. KP	18

Table 5.2: Top 20 endgame positions (for all victories and checkmates, respectively).

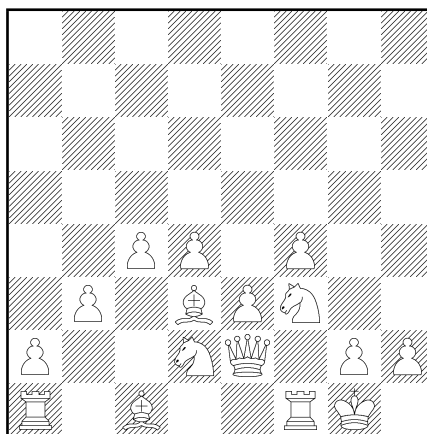
of the players is Krieg, an artificial player also known as Fark which has been playing on the ICC for a long time.

What follows is a listing of the ten most played opening moves for these players, together with the resulting diagrams (integrated with some other frequent moves, as some moves in the tables are mutually exclusive and only one could be played in those cases).

Draws	
K vs. K	273
KB vs. K	121
KQ vs. K	121
KN vs. K	103
KP vs. K	44
KR vs. K	41
KQB vs. K	27
KQN vs. K	21
KQQ vs. K	20
KQBN vs. K	19
KB vs. KB	16
KN vs. KN	16
KQR vs. K	15
KBN vs. K	12
KQBP vs. K	10
KQQN vs. K	10
KBP vs. K	10
KQRP vs. KP	9
KQQB vs. K	9
KQQP vs. K	9

Table 5.3: Top 20 drawn positions.

rjay	
d2-d4	66.89%
Ng1-f3	63.91%
f2-f4	60.76%
e2-e3	51.82%
O-O	43.54%
Bf1-d3	41.23%
e2-e4	39.07%
c2-c4	38.74%
b2-b3	36.75%
Nb1-c3	30.63%

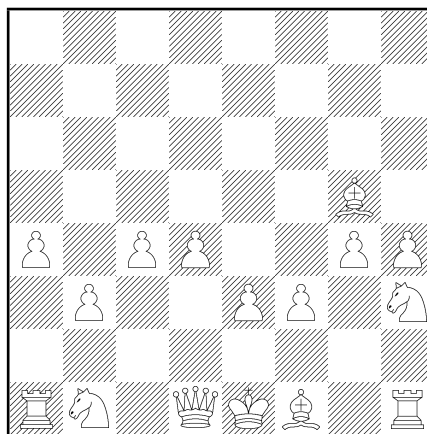


Player rjay's typical plan involves pushing the central pawns early in the game, developing a knight and bishop and with a marked tendency towards

kingside castling (queenside castling occurs about 5% of the time). The 'a' and 'h' pawns are stationary.

CJunk

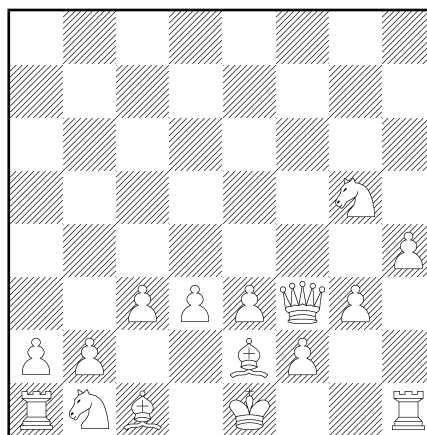
d2-d4	68.35%
h2-h4	52.37%
a2-a4	40.03%
Ng1-h3	38.77%
f2-f3	31.80%
c2-c4	31.65%
f2-f4	28.32%
Bc1-g5	26.27%
e2-e3	24.53%
g2-g4	23.58%



This player is an aggressive pawn-pusher who aims to reduce the opponent's maneuvering space rather than developing his or her own pieces. This style is completely different from the first sample player's.

Sting-R

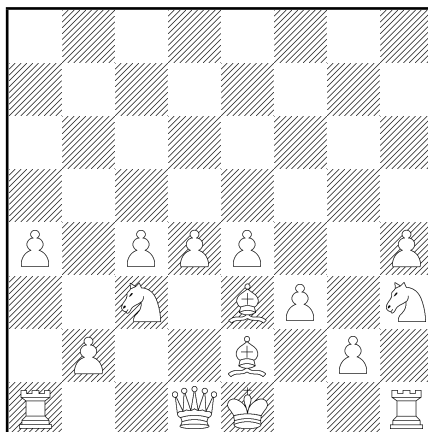
e2-e3	90.18%
Ng1-h3	55.44%
d2-d3	45.61%
Bf1-d3	44.56%
Qd1-f3	35.79%
g2-g3	35.09%
h2-h4	35.09%
f2-f3	31.58%
c2-c3	31.23%
Ng1-f3	31.23%



This player exhibits a slower pawn advance, usually starting from the middle files and moving right, which seems his or her favorite direction to attack from. Sting-R never castles in the first 10 moves of a game, and has a tendency to move the queen earlier than the previous players.

Budgie

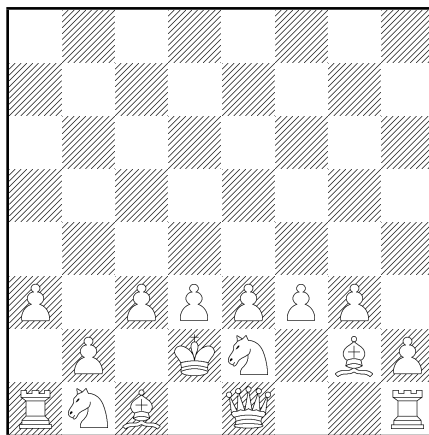
e2-e4	63.10%
d2-d4	62.26%
Nb1-c3	41.30%
c2-c4	33.33%
f2-f3	29.98%
c2-c3	27.88%
d2-d3	27.04%
h2-h4	26.83%
Ng1-f3	26.00%
a2-a4	24.74%



This player is another pawn-pusher, though his or her action is invariably the strongest on the middle files, whereas two files, 'b' and 'g' follow much later. Also, Budgie takes fewer risks with bishops than CJunk does.

Krieg

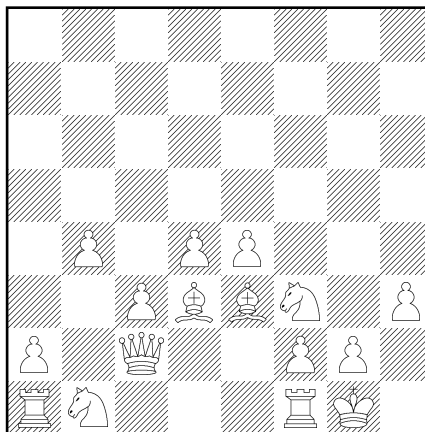
d2-d3	88.09%
f2-f3	83.79%
e2-e3	79.30%
g2-g3	66.99%
c2-c3	57.03%
Ke1-d2	47.07%
Ke1-f2	33.98%
Qd1-e1	29.88%
Ng1-e2	24.02%
Qd1-b3	18.36%



Krieg is an artificial player, and its opening play seems rather repetitive, as shown by the high percentages next to its most frequent moves. In fact, when left undisturbed, Krieg generally moves its pawns by one, starting from the queen pawn and spreading out, and then clusters several pieces in the newly created niche. Clearly, the goal is to maximize mutual protection, but this appears to be very sensitive to opponent modeling techniques. Interestingly, Krieg is the only player, of the five considered in this section, to move its king (other than castling, which Krieg never does) with high frequency in the first 10 moves (something that Darkboard tends to do, as well; it seems

that humans are more reluctant than computers to move their kings at this stage).

paoloC	
e2-e4	98.58%
d2-d4	82.46%
Ng1-f3	72.51%
h2-h3	67.77%
c2-c3	60.19%
Bc1-e3	55.92%
Bf1-d3	42.18%
Qd1-c2	37.91%
Nb1-c3	37.44%
b2-b4	26.06%



Paoloc almost invariably begins a game by pushing the 'e' and 'd' pawns by two, in that order, often making use of them as shields to develop his bishops. Kingside castling is relatively frequent (23% of time in the first 10 moves), whereas queenside castling is almost unheard of.

5.2.6 Further development

A complete analysis of the database is beyond the scope of this work. The results shown are meant to exemplify the differences between chess and Kriegspiel chess, and how a purely chess-based approach to Kriegspiel seems to lead to poor results. A more detailed effort on the database is a priority for the future and includes, among others, the following points.

- For each top player, to analyze their overall attack schemes. Good players orchestrate large-scale attacks on a single square to gain material advantage. By studying the squares they target as well the planning that precedes the attack, not only could an artificial player perform effective opponent modelling, but also reproduce such advanced behavior.
- Conversely, the study of frequent weak points in a player's piece setup may prove a good opponent modeling tool.

- It would be desirable to find out the perceived 'value' of a piece for a given player. While there are established values for chess pieces, ranging from 1 for pawns to 9 for queens, the matter is unclear in Kriegspiel. A possible way to accomplish something like this (albeit with approximation) with the Darkboard engine would be to replay a player's games under different piece values and see when the program's choices are a best fit for the player's moves.

Chapter 6

Kriegspiel Algorithms: State of the art

Several aspects of Kriegspiel have been the object of research in the last decades. This chapter lists some of the most important contributions to date.

6.1 Algorithms for various Kriegspiel endgames

Herbert Simon, winner of the Nobel Memorial Prize in Economics in 1978 for his research on decision processes within economic organizations, was among the first scholars to realize the importance of the game of chess as a model of difficult decision processes (see [7]). Being interested in the mechanics of rational decisions, he first introduced the definitions of *substantial rationality* and *procedural rationality*. A behavior is said to be substantially rational when it looks for the *best* possible action to take. A behavior is said to be procedurally rational when it looks for an *acceptable*, reasonable action, keeping in mind objective constraints and available resources.

As it has been shown, Kriegspiel's complex nature makes a substantially rational approach to the whole game pointless – there is no perfect game in Kriegspiel because Zermelo's theorem and consequently the Minimax theorem do not hold with imperfect information. By reducing the situation to a small subset, however, it is indeed possible to prove interesting results. The Kriegspiel endgame is the main playground in this respect, with some

classical endgame archetypes having been considered in literature.

Endgame theory is extremely important in the game of chess; after all, checkmating the opponent is the only way to win. Typical endgames are among the first notions taught to beginner chess players, though they are rarely seen in professional games because an expert player can foresee his defeat and resign well before that point. In Kriegspiel, on the other hand, it appears that the losing player will more often keep playing even with the King alone, in an attempt to luckily capture his opponent's pieces or force a stalemate. Indeed, stalemate is usually the winning player's real opponent during the endgame, and the ratio of stalemated games shown in the Kriegspiel database is abnormally high. Interestingly, major victories with multiple queens and other pieces on the chessboard prove to be even more dangerous, as it is easy to unwillingly trap the King and draw by stalemate.

For the reasons outlined above, a thorough knowledge of endgames may be even more crucial for the Kriegspiel player than the chess one. On the other hand, Kriegspiel endgames do not translate exactly to Chess endgames; the same position may lead to a victory for White in chess, and not guarantee victory in Kriegspiel, and vice-versa yield a draw in chess but "almost" certain victory in Kriegspiel. For chess, in which the solution is obviously deterministic, commercial databases exist that can be plugged into the more popular chess-playing programs and contain solutions to all endgames with five or fewer pieces (which may become six in the near future). This means that a program with such a database will feature perfect play as soon as only five pieces are left on the chessboard.

Kriegspiel is not as straightforward. In [8] the authors explain that some endgames can be won *algorithmically*, whereas others only *statistically*. A position is won algorithmically if an algorithm can be written and proved that will solve it in a finite amount of moves. For example, the Rook vs. King endgame is algorithmically solved. However, this is not always the case, and in some endings (most notably, Pawn vs. King) White can never attain certain victory, though he can raise his probability of victory as close to 1 as he wishes (unless there is limit to the duration of a game, such as the 50 move rule). This is usually proved via classical Game Theory results. The following sections outline some general results, even though the exposition is

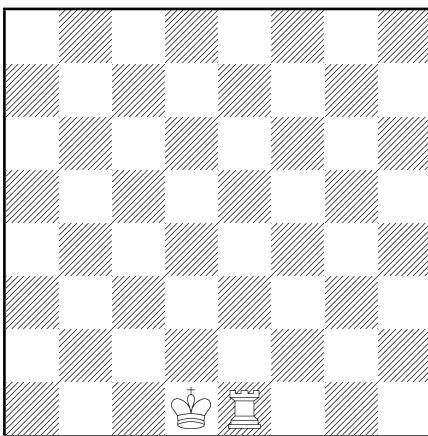
necessarily incomplete; see, for example, Ferguson's treatment of the bishop and knight (KBNK) [10] and the two bishops (KBBP) [11] endgames.

6.1.1 King and Rook vs. King (KRK)

A well-known endgame in orthodox chess, studied since the XIX century. [2] is largely devoted to this endgame, describing how Torres y Quevede built the first mechanical player for this endgame in the last decade of 1800. The accomplishment was impressive even though the machine's algorithm was inefficient and could require more than 50 moves to checkmate the black king. This endgame is computationally easy to treat in chess, as it is possible to merge many different positions where the pieces are 'reflected' along the x or y axis, or the diagonals. With this optimization, solving the resulting 23424 positions is not particularly hard.

Even though this endgame can always be won, the problem space in Kriegspiel is, on the other hand, intractable via brute-forcing even in this simpler scenario, numbering roughly 2^{52} information sets. Magari [22] provides the following KRK algorithm for Kriegspiel.

- First, White positions his pieces as in the following diagram.



- Then, White begins to move his pieces upwards, with Kd2, Re2, Kd3, Re3, ..., Kd8, Re8. If at any time the umpire announces a check, White knows whether the king is on the left or right side of the chessboard. If checks are never announced, the king must be on the left side.

- The White king is used to systematically reduce the possible locations of the opposing king, until it is forced against an edge. Then, the rook mates.

A more detailed algorithm is given in [5], as a set of simple directives. The algorithm is straightforward but rather lengthy; it is based on securing both kings into a single quadrant while keeping the rook safe. It will not be reproduced in this work.

6.1.2 KRK on Infinite Chessboard

An interesting variation upon the KRK theme is on an *infinite chessboard*. An algorithm for this unusual case has been given by L. Shapley and A. Matros during the ninth Game Theory convention held in Stony Brook, NY, in 1998. The algorithm manages to transform the problem into the usual case on a finite chessboard. The authors assume the following starting conditions.

- The chessboard is a quadrant of the two-dimensional plane.
- The White pieces start out near the origin, as shown in Figure 6.1.
- White does not know Black's starting position.
- White must win 100% of the time (obviously, there is no 50 move rule here).

The algorithm proceeds as follows.

- The first step is to check the enemy king once. White plays Ra3, Rb3, Rc3, Rc1, Rc2, Rc3. If no check took place, we know that the king is not anywhere in the first three rows or columns.

Then, White plays Rc1. After Black's move, the king is at least in the third rank and to the right of the 'c' file. White chooses randomly one of the ten files next to 'c', each with probability 0.1, moving the rook there (it is safe as the second rank is clear). If the move does not end in a check, White call back the rook with Rc1, clears the first two ranks once again with Rc2, Rc3, Rc1, and then selects another file, this time

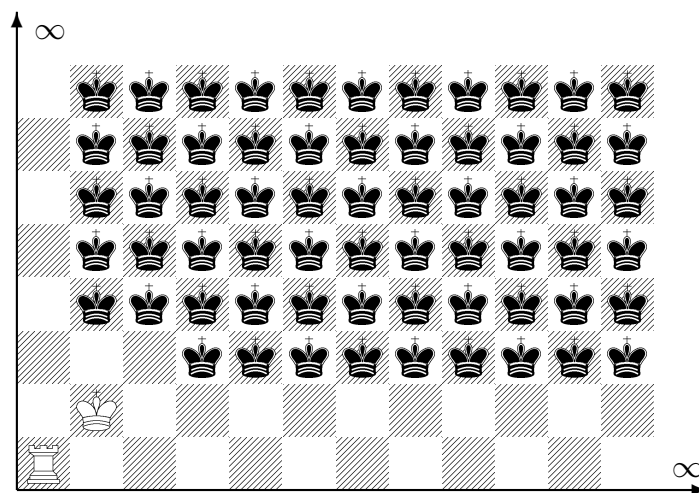


Figure 6.1: KRK on infinite chessboard (Shapley).

from the first 20 with probability 0.05. Each time, he will select one of the first 20, 30, 40, etc. files. Eventually, White is bound to hear a check message.

- White moves the rook 100 files to the right of the file where he checked the king. Then, he repeats the first step on the other axis. Only, since he does not have the king to protect the rook there, if he has not checked the king again after 97 moves, he will move the rook further 100 moves to the right. Again, White is bound to hear a check sooner or later.
- White moves his rook 100 ranks above the point where he checked the king, and then brings his own king to protect it. It may be necessary to further move the rook to the right.
- The Black king is trapped within a finite quadrant of the chessboard. It is now possible to checkmate as in the finite case.

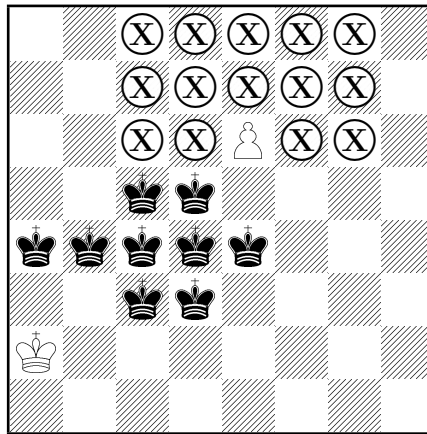
6.1.3 King and Pawn vs. King (KPK)

This endgame archetype, discussed in [8], is extremely interesting. White can always win this endgame in orthodox chess, assuming he can secure his

pawn in the beginning; however, it can be shown that, even acting under substantial rationality as the authors do, this endgame is only statistically won in Kriegspiel, meaning that White will win with probability $1 - \epsilon$, with $\epsilon > 0$ small and depending on the length of the game.

What we call *metaposition* in this thesis, the authors in [8] call *wave*, as the possible locations of the Black king can be likened to a quantic wave. They describe a Prolog program that solves this endgame by comparing the current position with five patterns, and taking appropriate actions. The patterns are checked in the following order.

- **Wave out of quadrant.** There is an easily proven rule of thumb in chess, stating that if the opposing king is outside an imaginary half-square determined by the pawn's distance from the last rank, then the pawn can freely advance without fear of retaliation. The quadrant is shown as X's in the next diagram.

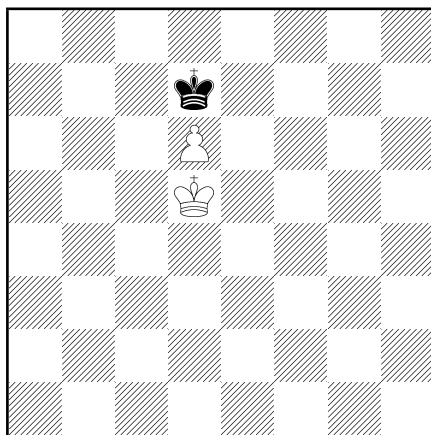


Since the king's wave does not intersect the quadrant, White will merely push his pawn forward in this position.

- **Blotto.** Named after a famous problem in Game Theory, Blotto happens when the pawn is in the sixth rank and the White king is behind and next to the pawn. Since the Blotto problem guarantees $1 - \epsilon$ payoff with optimal play, the payoff extends to this situation, which is equivalent to it. It is the most complex case, better described separately.

- **King next.** If the first two patterns do not match and the king is next to the pawn, this pattern is chosen. The agent now has three courses of action, with decreasing priority: *push* the pawn forward, *move* the king forward, or *maintain position* (move the king randomly so it is still guarding the pawn). The third action is always possible.
- **Distance WK-WP < BK-WP.** If the White king is closer to the pawn than the closest Black king in the wave, White moves the king closer.
- **Distance WK-WP > BK-WP.** If anything else fails, there is no guarantee White can win this position. Since it is not clear whether the enemy king is behind the pawn (in which case, he should push the pawn) or ahead of it (and then he should move his own king closer), White employs a mixed strategy of the two.

Of the five patterns, Blotto deserves the most attention. A possible situation matching the pattern is depicted in the following diagram.



Let us assume that Black is now to move. He will move to one of c8, d8 or e8. Moreover, he will spend the remainder of the game in those three squares, because any other strategy would lead to White trivially promoting his pawn and mating as described in the previous sections. Then White has four strategies, α_1 through α_4 .

- α_1 : 1. Kc6 ... 2. try Kc7, if legal then White wins, else Kd5.

- α_2 : 1. Ke6 ... 2. try Ke7, if legal then White wins, else Kd5.
- α_3 : 1. Kc6 ... 2. try Kc7, if legal then White wins, else c7.
- α_4 : 1. Ke6 ... 2. try Ke7, if legal then White wins, else c7.

The first two strategies involve no risk on White's part, as he merely tests some ground with the king. If his king moves are legal, then White can promote his pawn without resistance. However, if White refuses to take risks, limiting himself to α_1 and α_2 , and Black realizes this, Black has a counter-strategy that will put his king in d8 every other move, thus preventing White from ever succeeding. Therefore, White must eventually take a risk and play α_3 or α_4 . The risk here is that, if the Black king is aligned with the White one, Black can play d8 and put himself on the pawn's way. Then, White would have to choose between stalemate, if he kept protecting the pawn, or a draw if he moved his king away from it for Black to capture. However, if Black was already in d8 (as he would have to, in order to protect himself from the expected risk-free strategies α_1 and α_2), he would be forced to leave the eighth rank and surrender.

This situation is equivalent to *Blotto's problem*. Colonel Blotto had three divisions under his command, and was charged with taking an enemy camp, defended by two divisions. However, Blotto must make sure that the enemy did not take his own camp in the process. In order to conquer a camp, an armed force had to outnumber the defenders, otherwise they would just retreat to their camp, ready to resume fighting the next day.

Let $\beta_0 \dots \beta_3$ be Blotto's strategies associated with sending 0, 1, 2 and 3 divisions, respectively. Also, let $\gamma_0 \dots \gamma_2$ be the corresponding strategies for the enemy commander. Then we have a *recursive game* which can be expressed in normal form as in Table 6.1, where 1 represents Blotto's victory, -1 represents defeat, and Γ means the game is repeated. It is seen that β_0 never leads to victory and will never be chosen. Here, the risk-free strategy is to always send one division, but if the enemy knows this, they can prevent Blotto from ever winning.

Thus, Blotto, just like White in the KPK endgame, has to take a risk and play β_2 or β_3 every once in a while. In fact, the optimal strategy is a mixed one with probabilities $0, 1 - \delta_1 - \delta_2, \delta_1, \delta_2$. Moreover, the smaller

	β_0	β_1	β_2	β_3
γ_0	Γ	Γ	Γ	1
γ_1	Γ	Γ	1	-1
γ_2	Γ	1	-1	-1

Table 6.1: Blotto’s problem, normal form.

δ_1 and δ_2 , the longer the game will last, and the higher Blotto’s chances. It appears that patience is Blotto’s virtue, and the same applies to White in the equivalent Kriegspiel endgame. The only difference is that, while Blotto can have his chance of victory approach 1 by waiting indefinitely, White has a time limit due to the 50 move rule, and having to launch the final strike before that time, he cannot arbitrarily improve his chances.

6.2 The Monte Carlo sampling player

Recently, A. Parker, D. Nau and V.S. Subrahmanian [29] have proposed what is considered the first serious attempt at an artificial Kriegspiel agent capable of playing an entire game instead of merely one specific endgame situation; this does not account for the AI’s available on the ICC (mainly Fark/Krieg and G2K, the latter being admittedly based on heuristics aimed at protecting the player’s pieces and only knowing a few checkmating algorithms).

The authors call the information set associated with a given situation a *belief state*, the set containing all the possible game states compatible with the information the player has gathered so far. They apply a statistical sampling technique, which has proven successful in several imperfect information games such as *bridge* and *poker*, and adapt it to Kriegspiel. The technique consists of generating a set of sample states (i.e. chessboards, a subset of the information set/belief state), compatible with the umpire’s messages, analyze them with well-known perfect information algorithms and evaluation functions, such as the popular and open source GNUChess engine, choosing the move that obtains the highest average score in each sample (Monte Carlo sampling).

Obviously, in the case of Kriegspiel, generating good samples is far harder than anything in bridge or poker. Not only is the problem space immensely

larger, but also the duration of the game is longer, with many more choices to be taken and branches to be explored. For the same reasons, evaluating a chess move is computationally more expensive than a position in bridge, and a full minimax has to be performed on each sample; as a consequence, fewer samples can be analyzed even though the problem's size would command many more.

The authors describe four sampling algorithms, three of which they have implemented (the fourth, AOS, generating samples compatible with all observations, would equate to generating the whole information set, and is therefore intractable).

- **LOS** (Last Observation Sampling). Generates up to a certain quantity of samples compatible with the last observation only (it has no memory of what happened before the last move).
- **AOSP** (All Observation Sampling with Pool). The algorithm updates and maintains a pool of samples (chessboards), numbering about a few tens of thousands, all of which are guaranteed to be compatible with all the observations so far.
- **HS** (Hybrid Sampling). This works much like AOSP, except that it may also introduce last-observation samples under certain conditions.

The authors have conducted experiments with timed versions of the three algorithms, basically generating samples and evaluating them until a timer runs out, for instance after 30 seconds. They conclude that LOS behaves better than random play, AOSP is better than LOS, and HS is better than AOSP.

It may surprise that HS, introducing a component of the less refined LOS, behaves better than pure AOSP, but it is in fact to be expected. The size of the AOSP pool is minuscule compared with the information set for the largest part of the game. No matter how smart the generation algorithm may be or how much it strives to maintain diversity, it is impossible to convey the full possibilities of a midgame information set. so the individual samples will begin to acquire too much weight, and the algorithm will begin to evaluate a component of *noise*. The situation worsens as the pool, which is already

biased, is used to evolve the pool itself. Invariably, many possible states will be forgotten. In this context, LOS actually helps because it introduces fresh states, some of which may not in fact be possible, but prevents the pool from stagnating.

It is to be noted that the Darkboard player described in the next chapter follows the opposite philosophy of storing a *larger* information set than the (already huge) belief state associated with a Kriegspiel game, thanks to metapositions.

6.2.1 Strategy fusion

Any attempt at solving an imperfect information game through sampling must be wary of *strategy fusion*. The term is due to Frank and Basin [13]. The authors, who deal with bridge as their main research interest, show that techniques such as Monte Carlo sampling with repeated minimax suffer from an inherent flaw that leads them to sub-optimal play. As previously stated, a standard chess engine like GNUChess may be used to compute minimax values for many possible states, MAX being the player who tries to maximize his payoff, and MIN being the opponent who does what he can to minimize it. In the evaluation of a game state via traditional minimax, both players are assumed to make the best choices available to them, both of which are incorrect assumptions.

The term strategy fusion refers to the fact that MAX incorrectly believes he can distinguish among the various states he might be in, when he cannot, and thus he concludes that he has a profitable strategy for each possible world (which can be true, but since he does not know in which world he is, this fact is of no use to him). The authors make use of the example tree in Figure 6.2. Here, if one were to use statistical sampling with normal minimax (sampling five worlds w_1, \dots, w_5), Player 1 would conclude that, in the left subtree, he always has a strategy yielding payoff 1, just like the right branch. Thus, any rational strategy would evaluate Player 1's root choices as indifferent, when it is obvious that the right branch is better. Strategy fusion means that Player 1 knows the best strategy for every node and thinks he has the luxury of choosing which one to apply on a case-by-case basis instead of choosing *one* strategy across all the possible nodes, since those nodes will all

look the same to him. The mistake leads to overestimate the value of each node. The authors argue that statistical sampling may yield good results in practice because of the underlying structure of the game world, where 'similar' strategies often bring 'similar' payoffs.

There is also a problem with modelling the opponent's moves. Is it correct to assume that MIN will always play his best move? Any evaluation using GNUChess or a similar engine will act under the assumption that MIN knows the location of MAX's pieces. This is an open problem in and of itself. Expecting the worst-case scenario from the opponent works with perfect information because of the assumption of rationality as common knowledge: I know you are rational, and I know you know I am rational, and so on. But in the case of imperfect information, such an assumption leads to sub-optimal (over-conservative) play, as also shown in [20] – unless, of course, the opponent *really* acts with perfect information; there are chess variants, halfway between chess and Kriegspiel, where knowledge is asymmetrical, one player moving in the dark and the other with perfect information but playing with the immense handicap of no major pieces.

Experimental results show that the loss of accuracy that arises from assuming MIN's optimal strategy depends on the size of the problem's space. As the size of the information set decreases and the game's uncertainty fades away, MIN's strategy will tend to approach the optimal minimax strategy with perfect information.

This fact leads to speculate that, in the case of Kriegspiel, where the information set's size varies wildly and decreases greatly towards the endgame (especially for the winning player), different algorithms may be necessary to handle different stages of the game.

6.3 Metapositions

The concept of *metaposition* is first given in Sakuta [32]. The primary goal of representing an extensive form game through metapositions is to transform an imperfect information game into one of perfect information, which offers several important advantages and simplifications, including the applicability of the Minimax theorem. A metaposition, as described in the quoted work, *merges different, but equally likely moves, into one state* (but it can be

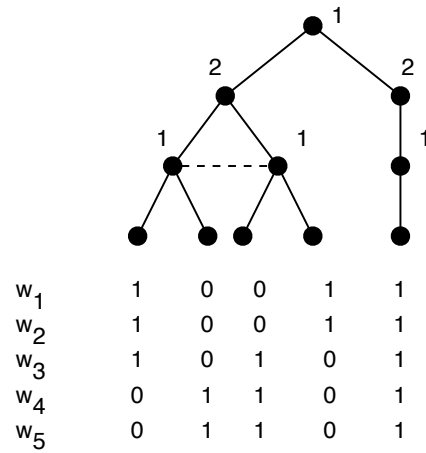
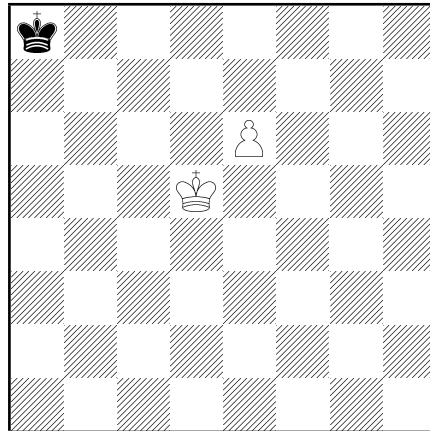


Figure 6.2: Strategy fusion (from Frank and Basin, 1998).

extended to treat moves with different priorities).

Let us introduce the concept through a Kriegspiel example based on the following diagram.



Let us also suppose that Black is now to move. He has three possible choices: a7, b7 and b8. White's possible moves on the next ply depend on which one Black chooses now; in particular, if Black plays Ka7 or Kb8, White has the same 7 king moves plus a pawn move. However, if Black selects b7, White will not be able to play Kc6, and will only have 6 king and 1 pawn move to choose from. In other words, a7 and b8, while different moves, do

not differ in the strategy space available to White on his next move. They are excellent candidates for merging into a single *metaposition*.

The result of the merging is shown in Figure 6.3. Uncertainty has disappeared, at least officially; White knows where he is from his current strategy space, as no two child nodes can share the same move set (or they would be merged). Also, since the game is now of perfect information, it makes sense to generate an *evaluation function* and start assigning each node a minimax value. The value of a metaposition node could be the minimum value across all the positions that make up the metaposition.

It should be said that Bolognesi [2] finds this basic use of metapositions both impractical and insufficient to implement a player that solves the KRK endgame. Impractical because, in order to compute a minimax value for a metaposition node, it is still necessary to invoke the evaluation function on each and every member position. Insufficient because the amount of nodes is still very high, depending on the situation; in fact, it becomes *even more* insufficient when one tries to extend it to the whole game, as is the aim of the present work. [2] deals with scenarios with a branching factor of about 20 for White, with Black moves being able to stop only a few of them at a time (the turning point here is that Black only has a king, which cannot block rook moves but only the White king's moves). Therefore, in the KRK endgame it can be argued that many Black moves will lead to the same information set for White, and a decent merging ratio. In the whole game, on the other hand, with a branching factor of 60-70 *for both players*, many White moves will have a Black move that makes them impossible (or conversely, make new moves possible). In the typical Kriegspiel middle game, relatively few positions could be merged together.

A more effective solution for Kriegspiel games consists of changing the notion itself of uncertainty. The definition of metaposition changes so that the merging will not only include the moves that lead to the same information sets for White, but *any move that is possible from the current metaposition, even if that move is not possible in every state making up the metaposition*. This is equivalent to saying that *all of Black's moves for a given metaposition are merged together*; Black does not select a strategy anymore – he only has *one* choice – and effectively stops being a player, at least according to White's new vision of the world. The player vs. player situation shifts to player vs.

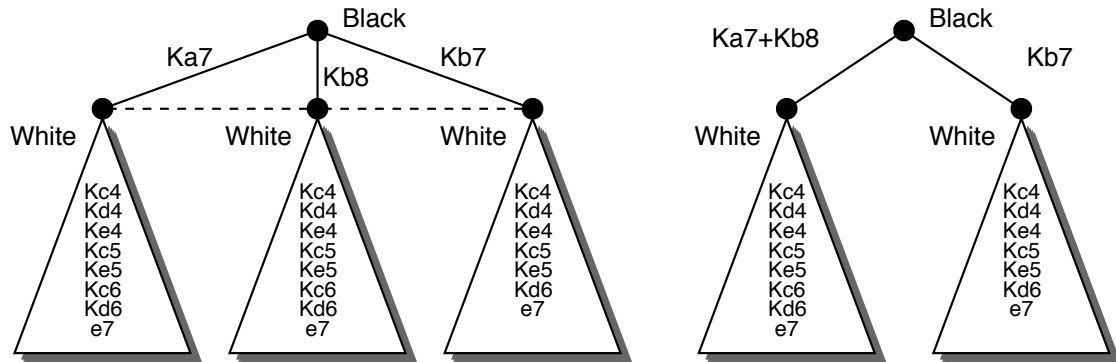


Figure 6.3: Partial Kuhn tree (left), with state merging and basic metapositions (right).

nature, where all the opposing strategies are considered simultaneously. We can no longer speak of moves for Black, then; instead, we define this aggregate entity as a **metamove**. The partial Kuhn tree of Figure 6.3 can be reformed to make use of metamoves as in Figure 6.4.

It should be noted that it is not necessary for every Black move to be grouped into a single metamove; for example, in the case of Kriegspiel, White has some limited information on Black's choices coming from the umpire, and as such there may be more metamoves, grouping moves that cause different umpire messages (silent, capture, etc). One advantage of making metamoves so generic is that we can choose what we want to put into them, depending on our preferences, need for accuracy, and size of the problem space. The Darkboard engine, which works with an approximation of metapositions at its core, generates a single metamove per node (also, it will not generate *every* possible opponent move) due to its own scope. For smaller problems, it may make sense to split the metamoves for more accuracy, so long as an appropriate algorithm can be found to evaluate the relative priorities of each.

As a consequence, branching factor becomes asymmetrical. The player's color chooses his moves as usual, whereas the opponent's are grouped together; performance is exponentially better as far as raw node numbers are concerned. Of course, a price must be paid for this apparent 'miracle', that is, White no longer knows his own possible moves since his information sets have been juxtaposed. But this precisely reflects the nature of Kriegspiel,

where it is very rare for a player to know that his moves will be legal before he plays them. All a player knows is that there are moves that *may* be legal. In the example, Kc6 may or may not be accepted by the referee, but is included as an option because it is a legal move in at least one possible state. We refer to these options as **pseudolegal moves** and the act of playing a pseudolegal move is a **pseudomove**. This, too, is a concept reminiscent of quantum mechanics; the idea that we do not know if a move is legal until we actually play it, and we get to find out more about the game world while changing it at the same time.

Here, it becomes apparent that Kriegspiel breaks the established Minimax scheme. If a move fails, the player is required to try again. Hence, one or more MAX moves may appear without any MIN move in between. An algorithm could explore the possibility of illegal moves as branches of MAX's moves. For example, Bolognesi considers branching a pseudomove into three: silent umpire, check notification, and illegal move. Full exploration is, however, impossible; with n pseudolegal moves and upon the first one being rejected, MAX still has, in general, $n - 1$ pseudolegal moves, which can in turn be illegal. An intractable complexity of $n!$ follows from even a single metaposition.

The author in [2] solves the problem by invoking the evaluation function on the three child nodes, and expanding only the one with the lowest value. Darkboard moves from the same theoretical reasoning but cannot afford even this greatly reduced computational effort. The program uses a set of acceptable, hard-coded directives in order to generate a single child node for a given future pseudomove. In particular, *Darkboard assumes its moves to always be legal*. Indeed, most of the time an illegal move is 'better' than a legal one because it reveals information for free; the player knows more while still retaining the right to move. Darkboard assumes the worst, which generally entails the move will be accepted (note that this does *not* apply to Invisible Chess, a chess variant where only a few pieces are hidden; there, an illegal move causes the player to lose his turn and the above reasoning would not hold). If the chosen move then proves to be illegal, Darkboard will re-run the algorithm with the new information and generate a new strategy.

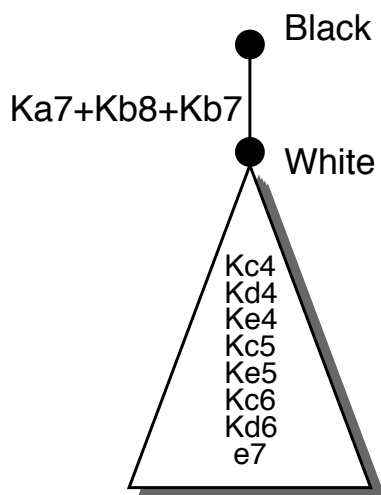


Figure 6.4: The same tree with extended metapositions: Black’s metamove.

6.3.1 KRK with metapositions

The best way to prove that metapositions actually work is to see them at work, as show in [3] and [4], where the KRK endgame, which has been treated with conventional algorithms earlier in this chapter, is tackled with this tool. This technique was the main source of inspiration in creating Darkboard, even though there are numerous and important differences between the two: the authors here use *exact* metapositions (luxury of dealing with the king only) whereas Darkboard analyzes *approximated* metapositions and follows far more complex rules to generate reasonable pseudomoves and metamoves (in fact, one could argue that Darkboard does not really play Kriegspiel, but an equivalent perfect information game whose optimal strategy is hopefully a reasonably good Kriegspiel move).

To summarize, from a practical point of view, metaposition-based programs never perceive individual positions; instead, they have a ‘quantic’ notion of opposing pieces, wherein they fight against multiple copies of the same pieces on different squares. In the KRK endgame, there are as many enemy kings as there are squares on which the real king may be at any given time. Each of these kings inherits the full capabilities of a real king; they can move to neighboring squares, capture unprotected pieces, and so on. In the case

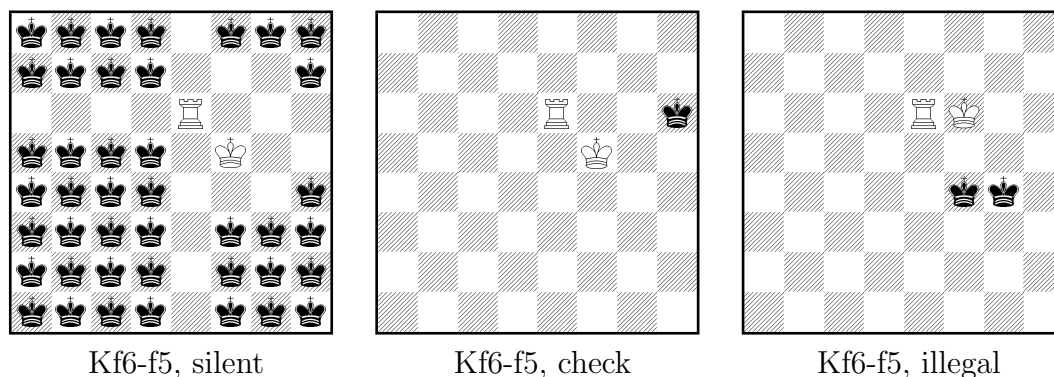


Figure 6.5: The three possible evolutions following White's pseudomove Kf6-f5. The program will only explore the worst case (the leftmost example, in this case).

of KRK, it is easy to generate the appropriate 'evolutions' of a metaposition given a pseudomove or a metamove (see Figure 6.5). It is much harder to do so when up to 16 enemy pieces are on the chessboard, but it can be done with the appropriate simplifications (see Chapter 7).

With the caveat of expanding only the worst possible outcome of a pseudomove, the KRK endgame, along with the similar KQK and KBBK (king and queen, king and two bishops, respectively), becomes tractable. The rook always has at most 14 pseudolegal moves to choose from, and the king has 8. The branching factor for the opponent's moves is 1, as shown. The crucial point, now, is to devise a good *evaluation function* that will gauge a metaposition and help track the player's progress towards checkmate. Clearly, such a function will not have much in common with an evaluation function seen in an orthodox chess program. Concepts such as mobility, pawn structure, king safety and so on become rather pointless in this context unless properly adapted for Kriegspiel.

Thankfully, the evaluation function for the KRK endgame is rather simple. The rationale behind the function is the same as the one motivating every algorithmic solution to this endgame: protecting the rook is the highest priority, avoiding stalemate comes a close second, then trapping the king inside as few quadrants as possible and finally delivering the killing blow.

The function is of the type

$$\text{Eval}(S) = c_1 f_1(S) + c_2 f_2(S) + \dots + c_5 f_5(S),$$

meaning that there are five components, each with an appropriate weight. The components are the following.

- **Rook safety.** This bonus is added if the rook is safe in the current metaposition. Since its weight is 840 (more than everything else combined), in practice the program will not willingly move to an unsafe position unless it has no other choice.
- **King distance.** This function is the distance between the White king and the farthest Black king on the chessboard. Since the algorithm aims to reduce this distance, it is given a weight of -1.
- **Quadrant area.** This function attempts to minimize the total area occupied by the enemy king. It is not only the sheer number of squares that counts, but also the number of quadrants (we have seen that the previous KRK algorithms can only checkmate when the king is trapped inside one quadrant). Therefore, the function is the product of the occupied area by the number of quadrants, and has a negative weight.
- **King-rook distance.** Moves that keep the rook next to the king are preferred, everything else being equal. This is a less crucial component, having a weight of 1.
- **Edge push.** The algorithm will try to push the king towards the edge of the chessboard. This is done by means of a matrix with different weights for different cases, edges and squares being highly preferred.

While the modified Minimax using the above evaluation function does not win every time because of loops, it does win most positions (about 96% in its first version, even more with stalemate detection added to it). In fact, Darkboard makes use of a few functions similar to these in the endgame. Moreover, opportunely crafted evaluation functions can solve different endgames such as KQK and KBBK. Using metapositions requires much more custom code than a Monte Carlo sampling solution; on the other hand, it

does not suffer from the 'amnesia' flaws of a sampling pool, and since it does not consider individual game states, it circumvents the problem of strategy fusion. Also, the burden of writing *ad hoc* evaluation functions that resemble nothing seen in chess is compensated for by the fact that metapositions portray the uncertainty in a situation, and the evaluation functions may evaluate the uncertainty itself. A metaposition-based engine will then be able to *willingly gather information*, whereas a sampling solution based on chess analysis of a set of possible states cannot do this.

Chapter 7

Structure of the Darkboard engine

Darkboard is a game engine for playing Kriegspiel under the ICC ruleset (Cincinnati style). It is written in the Java programming language and will run on any computer with the Java Runtime Environment version 1.3.1 or later.

Darkboard is a versatile program, made of an inner engine able to exchange information with any class inheriting from the `Umpire` superclass, as is seen in Figure 7.1. Currently, two subclasses of `Umpire` are available, `LocalUmpire` which allows for local play against humans or other artificial players, and `RemoteUmpire` which is used when the other player is not managed by the program itself, its only subclass being `ICCUmpire` for play on the Internet Chess Club, where the program is registered with the handle “darkboard” (anyone, including unregistered users, can view its profile and examine its latest 100 games). At the time of this writing, Darkboard only plays unrated games as it is still undergoing experimental testing. Hopefully, it will be able to engage in rated games in the near future.

7.1 Representing metapositions

The core of the program’s intellect resides in the `SimpleChessboard` class, which represents a single metaposition, and the `BestAIPlayer` class, which is responsible for selecting moves. Many operations on metapositions are

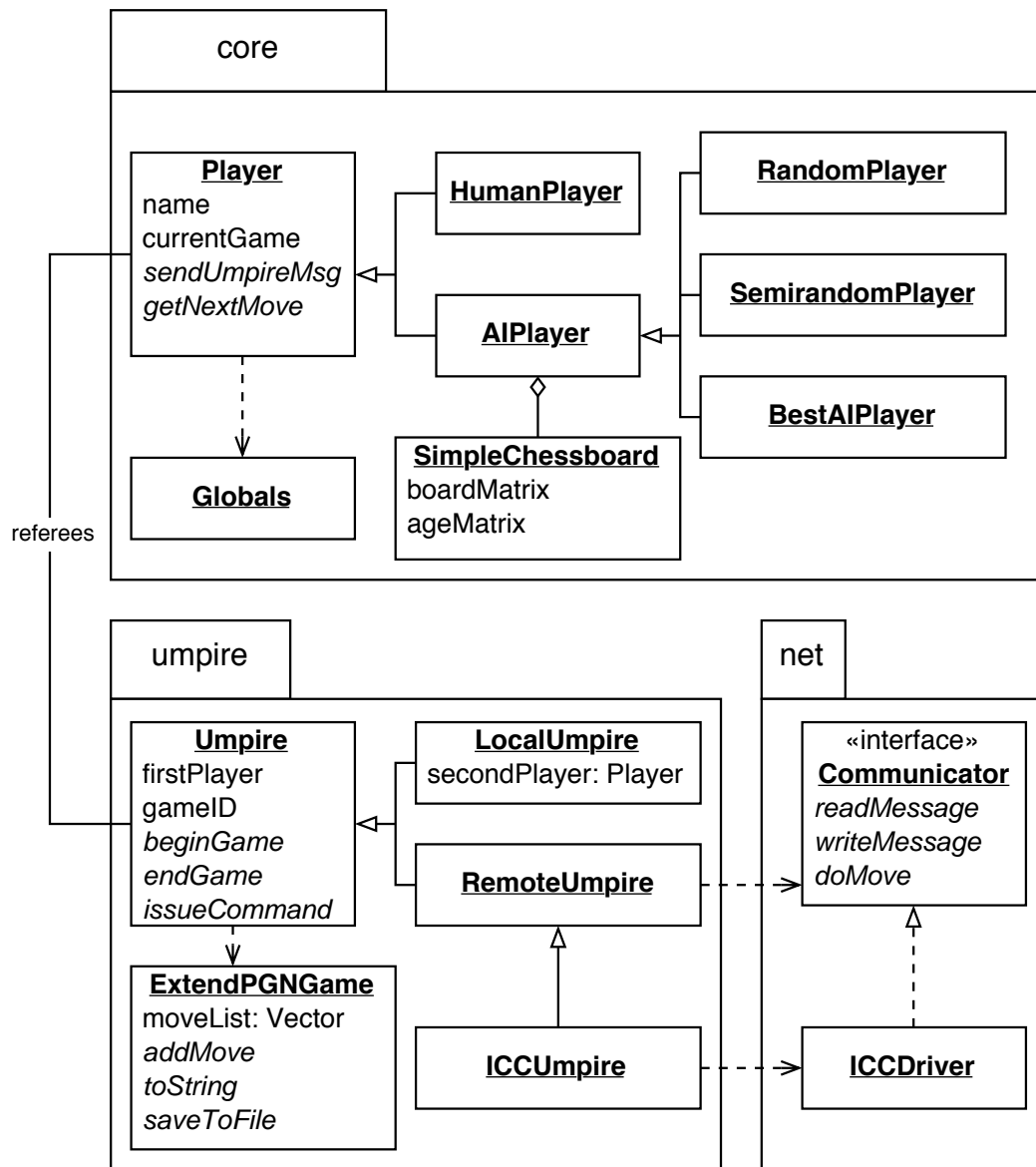


Figure 7.1: Simplified structure of the Darkboard engine.

possible, and these are all performed by the aforementioned classes, including:

- Editing the metaposition (i.e. amending the information it contains, thus extending or narrowing the information set).
- Evolving the metaposition with a successful player move.
- Evolving the metaposition with an unsuccessful player move (illegal move).
- Evolving the metaposition with the opponent's metamove and its associated messages.
- Generating the possible pseudomoves for the player to choose from.
- Calculating useful facts about the metaposition, including a protection matrix, and various estimates about the safety of each piece.
- Evaluating a metaposition.

It is to be noted that the metapositions used in Darkboard do not exactly match the definition given in the previous chapter. For practical reasons, a trade-off had to be chosen between the expressive power of a metaposition and the need to operate on it in a computationally feasible fashion; in other words, Darkboard's metapositions are necessarily less informative than strict metapositions because they contain more positions, but at the same time they can be easily represented using some 200 bytes of data, as opposed to the immense quantity of information that would be needed, if we were to list each position individually (which can easily grow to billions in a few moves; and that would still be *one* metaposition).

7.1.1 The main array

In Darkboard, a metaposition is, at its core, a 64-element one-dimensional byte array, where each byte represents one square of the chessboard. It could have been a two-dimensional 8x8 matrix, but performance dictated the use of a single array, especially because evolving a metaposition into another involves duplicating its data structure, and this happens very frequently.

Each byte in the array is actually a bitmask containing information about a single piece. Throughout Darkboard, each piece has a code number associated to it, as seen in Table 7.1. It may appear strange to consider 'empty' as a piece, but metapositions in Darkboard are merely concerned with what is impossible or possible on a given square at a given point. Our "simple" metapositions do not know exactly which positions they are comprised of, but they do know whether at least one of them has an enemy queen in d4, for example. In fact, the lower 7 bits in each byte of the main array match the piece codes. Bit 0 is set to 1 if there is a possibility for an enemy pawn to be in that square, and so on.

It should be noted that the above sentence reads *if*, not *if and only if*. Just like metapositions themselves, the process of evolving a metaposition is an approximation of the real thing, with trade-offs to allow the program to compute something useful within acceptable time limits. Darkboard's computations maintain the following invariant (barring any bugs): *if the piece bit is set to 0, then that piece is guaranteed not to be there*. The reverse is not true, and Darkboard will sometimes mark enemy pieces as possible in places where, strictly speaking, they could not be; it is a conservative approach which further enlarges the information set, though not significantly.

The bitmask's eighth and uppermost bit is used to signal the presence of an allied piece on the square, as seen in Table 7.2. When that bit is set, the other bits no longer represent a possible enemy piece; instead, by performing a simple bitwise AND operation to remove the first bit, we quickly obtain the piece code for the friendly piece; this is simpler than marking the corresponding piece bit and then using a lookup table.






Note: In the remainder of this work, it will be necessary to visually represent Darkboard's metapositions. The following conventions will be used to integrate a normal chessboard layout:

Piece	Code
Pawn	0
Knight	1
Bishop	2
Rook	3
Queen	4
King	5
Empty	6

Table 7.1: Darkboard piece codes.

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
isFriend	Empty	King	Queen	Rook	Bishop	Knight	Pawn

Table 7.2: Meaning of the metaposition bitmask. When isFriend is set to 1, the other bits change meaning, representing the piece code for the friendly piece on the square.

Symbol	Meaning
	Generic enemy piece (but can be empty)
	Possible piece (not all piece types)
	Confirmed enemy piece (cannot be empty)
	Possible king
	Possible pawn

7.1.2 The age array

The age array is another 64-element array, though its elements are of type *char*. Its main function is to keep information about the metaposition's history, as we have seen that, due to strategy fusion, the best move does not only depend on the current position, but also on what came before it. In practice, Darkboard needs more subtle information than the binary nature of the main array provides. Knowing that a piece may or may not be in a given square is obviously important, but not so much in the middle game as

in the endgame. Middlegame metapositions contain so many positions that pretty much anything is possible, anywhere.

Every square has an associated age value, which generally represents *the number of moves since Darkboard collected information about that square*. This has a broader meaning than just “since the player last visited the square”, because physically reaching a square or traveling over it is not always necessary to infer what it contains. For example, the absence of pawn tries (and checks), if the pawn and king are placed in such a way that the former cannot be protecting the latter, may indicate empty squares just as effectively, and their ages would be cleared back to zero. This is just one of several optimizations Darkboard can carry out.

The age array is involved in several calculations, two of which are especially important to the program: first, squares with high age values are seen as undesirable by the evaluation function, thus encouraging the player to visit them, and secondly, high age is associated with danger when estimating the safety level of a friendly piece. For this reason, when Darkboard finds that a path is obstructed, determining that a piece must be somewhere, it may artificially raise a square’s age level to represent increased danger.

7.1.3 Other information

A metaposition also carries several more fields, some of which are typical of a normal chess game (such as castling information), whereas others are unique to Kriegspiel (like the amount of captured material). These data are stored in arrays for faster copying, and include the following:

- Color information. (are we White or Black?)
- Castling information (kingside, queenside, both, neither).
- Captured pieces and pawns.
- Minimum and maximum pawn number on each file, inferred through captures and pawn tries. Aside from providing a positive bonus when a file is pawn-free, this is especially important when considering *pawn control* (see next section).

- Last pawn try count for both players.
- Total age count. This is the sum of the age values for each square on the chessboard, stored in a convenient field for performance reasons as it is often needed.
- Depth information. When a metaposition is evolved with a player's pseudomove, the depth is copied over; when it is evolved with the opponent's metamove, depth is increased by one. When building a pseudo-game tree, this field allows the evaluation function to know how deep the search is.

7.2 Move generation

Generally speaking, the move generation function is of the type

$$(\text{Metaposition} \times \text{boolean}) \rightarrow \text{Vector},$$

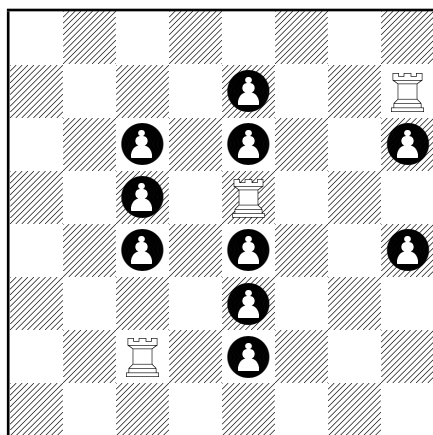
as it accepts a metaposition and a boolean and returns a Vector of Move objects containing the possible pseudolegal moves for the artificial player. The boolean parameter specifies whether the move is *top-level* or not; that is, whether the input metaposition represents the current state of the chessboard or a possible future evolution of it. When the top-level parameter is false, any matching move is included within the output Vector; but if the move is top-level, *banned moves* (pseudomoves tried and found to be illegal in the current turn) will not be included. Darkboard is actually a little smarter than that, and whenever a move fails, will not only mark the latest move as banned, but also any moves that are trivially illegal, as well (i.e. if Ra1-a5 fails, there is no point in trying Ra1-a8).

The move generation algorithm reasons like a traditional chess algorithm with the same purpose. Each piece travels as far as it can, stopping only when it meets a border, a friendly piece, or a square whose bitfield has the Empty bit not set. If there are any pawn tries, the algorithm will generate the corresponding moves except for those where the target square does not have any piece bit set.

However, there is one further check that Darkboard does, both here and in several other places throughout its code. This operation, called *pawn control*,

was born out of necessity, as Darkboard's metapositions are square-centered, not piece-centered. In other words, Darkboard knows whether there may be something, somewhere, but its metapositions cannot easily convey the fact that a certain piece is associated with a set of squares it *has* to be in. While the approximation works well enough with most pieces most of the time, it is insufficient to deal with pawns. Being the least mobile pieces on the chessboard but at the same time the most numerous, it became necessary to introduce pawn control to limit vertical moves that are clearly impossible.

Let us illustrate pawn control through a sample diagram.



Squares with the Pawn bit set are marked with the appropriate token. Files *c*, *e* and *h* all have an associated minimum pawn number of 1 or greater. In fact, this is a requirement for pawn control to take place; if a pawn is possible but not certain on a file, all the related pseudomoves will be generated.

The pawn control algorithm proceeds as follows. First, for each file it takes note of the highest and lowest ranks where the Pawn bit is set. The resulting intervals are called *pawn envelopes*. If the piece we are generating pseudomoves for lies inside the envelope, no restrictions are placed. Thus, the e5 rook will be able to (pseudo-)move to both e8 and e1. Clearly, both moves cannot be legal at the same time, but since we are unable to determine which one is, we are forced to accept both.

On the other hand, if the piece lies outside the pawn envelope, we can certainly block any attempt to reach the other side of the envelope. Hence, Rc2-c7 will not be generated; neither will Rh7-h3. However, if the target

square lies within the envelope, we should accept the move; Rc2-c5 is legal, and so is Rh7-h5.

It is to be noted that the algorithm does not consider the further restrictions that emerge with 2 or more confirmed pawns on the same file. Since pawn control takes place very often, and that situation seldom occurs (as it is comparatively much rarer to be positive that a piece of ours was captured by a pawn), such analysis is discarded.

7.3 Evolution of metapositions

As can be recalled from the last chapter, metaposition theory does not deal with moves, but with pseudomoves and metamoves. As a metaposition represents a grouping of a very large number of positions which cannot be told apart from one another, it is clear that updating such a data structure is no trivial task; in truth, despite being a simplification of the real thing, this process does account for the better part of Darkboard's computation time. Evolving a real metaposition, an information set, with a pseudomove would involve finding all the positions compatible with the outcome of that move (legal, not legal, check, etc.), discarding anything else, and applying the move to the compatible chessboards. Evolving a metamove would prove an even more daunting task, as we would have to consider each possible move for each possible chessboard in the set. Again, this is a problem that can only be overcome through a suitable approximation (or by limiting the number of chessboards down to a manageable pool, as in [29]).

Darkboard is provided with three different evolution algorithms, one for updating a metaposition with a legal move, one for illegal moves, and one for the opponent's metamoves. All of the above accept a metaposition and the appropriate umpire messages as their inputs, and return a new, updated metaposition. It may appear strange that the heart of the program's reasoning does not lie in the evaluation function but in these algorithms: after all, their equivalent in a chess-playing software would trivially update a position by clearing a bit and setting another. However, the evaluation function's task is to evaluate the current knowledge. It is the evolution algorithms that compute the knowledge itself, and given Kriegspiel's strongly imperfect information, it is imperative to infer as much information as possible in the

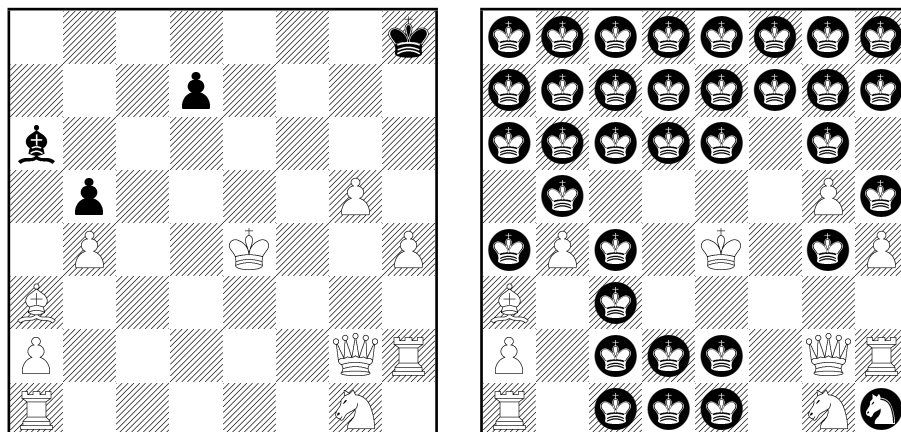
process.

7.3.1 Evolution with legal moves

This algorithm accepts as its input a starting metaposition, a move which is assumed to be legal, and the following information: *capture type* (which can take one of the following values: *noCapture*, *capturePawn*, *capturePiece*), *check1/check2* (as there can be up to two simultaneous checks; accepted values are *noCheck*, *knightCheck*, *rankCheck*, *fileCheck*, *shortDiagonalCheck*, *longDiagonalCheck*), *pawn tries* (for the opponent, after this move; the player's pawn tries are handled as part of the opponent's move evolution).

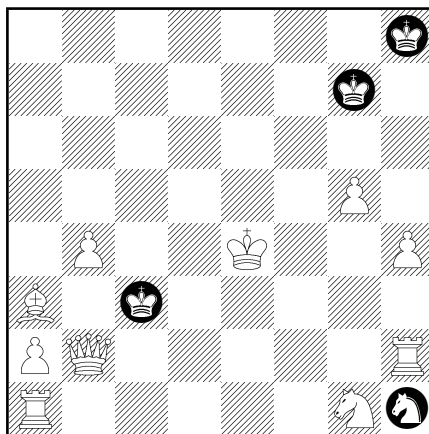
The function performs a few simple operations first, such as setting the visited squares to empty status and clearing their age values. Then, if the player captured something, it updates the count of captured material. If a pawn is captured, and the maximum pawn count for its file was 1, all pawns are removed from that file. In particular, if the amount of captured pawns reaches 8, pawns are removed from the chessboard altogether. Unfortunately, the same cannot be done with pieces when their capture count reaches 7, as pawns may have since attained promotion. However, if the player manages to capture 15 times, everything is cleared off the main array but king bits. In this case, obviously, the metaposition's accuracy increases drastically.

This being said, dealing with checks is the only non-trivial task here. Let us illustrate it with an example taken from an actual Darkboard game. The opponent is a semi-random player, which always uses pawn tries and recaptures when possible, and otherwise plays random moves. It makes a much stronger opponent than a fully random player (both have been used in the development of Darkboard). The diagram on the left shows the umpire's view, whereas the one on the right is Darkboard's knowledge (rather, a part of it; it is difficult to convey exactly which pieces are possible on this type of diagram, so we will narrow the scope of the diagram to the king; squares rendered as empty are not necessarily empty according to Darkboard, but they cannot contain a king.)



The knight token in h1 was placed on purpose, as Darkboard knows that the only piece which could be there is a knight. The enemy king's possible locations are easily determined by firing 'sweeping beams' from allied pieces. A beam will stop as soon as it encounters any square that is not certainly empty (or king-only, of course); meaning that so long as enemy pieces other than pawns remain on the chessboard, beams can rarely proceed further than one square away. This fact is backed up by the tendency of human players to finish off any remaining enemy pieces before focusing on checkmating the opponent.

If now White plays Qb2 and the umpire accepts it with the message "White moved, long diagonal check", Darkboard will evolve the metaposition as follows.



That is, the previous king's locations are scanned one by one, and only those compatible with the latest move and check type are allowed to remain.

Actually, reality is a little more complicated than this, and the process does not always prove to be straightforward, due to captures and *discovery checks*, wherein the piece that moves is not the one threatening the king. Therefore, the algorithm proceeds in this way:

- In the event of a double check, there is no ambiguity whatsoever; the piece responsible for the discovery check is also uniquely determined. The intersection of the two sets of squares for the two checks provides the king's exact location.
- If the check type is compatible with the piece being moved, *and that piece is not a pawn*, it cannot be a discovery check. Simply remove the king's current locations that do not match the check type. The king can never be found in the opposite direction of the piece's movement (or it would have been in check even before the move); and it can only be found "at 12 o' clock" if a capture also took place (i.e. the piece that protected it was just captured). This is especially important with diagonal checks, as 'long' or 'short' diagonal refers to the king's perspective, not the attacking piece's.
- If the check type is not compatible with the piece being moved, such as a file check when a knight was moved, look for discovery check candidates. Fire 'beams' from the piece's starting square along every direction that is compatible with the check type. If the beam reaches an allied piece that is compatible with the check type, we have found a candidate. At least one candidate is guaranteed to be found, but in rare cases, depending on the placement of the opponent's pieces, there could be two or more candidates.

For each candidate, there exists a set of target squares for the king to be in. The set is a segment that extends from the moved piece's starting square in the opposite direction than the candidate. We therefore rule out any current location that does not belong to any of the candidates' sets.

- The worst case scenario happens when the player has moved a pawn and the umpire announces a diagonal check. Because pawns do not

capture the same way in which they move, it could be either a genuine pawn check, or a discovery check from a bishop or queen behind the pawn. As a consequence, the algorithm will try both schemes and rule out any square that matches neither.

- In order to narrow the choices down even more, pawn control could and should be taken into account with file checks.

7.3.2 Evolution with illegal moves

Extracting information from illegal moves is extremely important because, unlike most other umpire messages, such information is asymmetric; there is no way for the opponent to know what move was rejected (and on the ICC, there is no way to know that an opponent's move was rejected to begin with). If we were dealing with a theoretical metaposition, an information set, we would simply drop any position that considered the move as legal. Unfortunately, such a subset consists of highly diverse positions, which are impossible to fully describe with Darkboard's data structures. The following actions can, however, be taken with relative ease.

- If the king is the only enemy piece left, a failed king move will narrow its possible locations to five squares at most. A failed pawn push can pinpoint the king's location.
- If the tentatively moved piece is not protecting its king (that is, the king cannot be found along any of the eight compass directions from its starting square, except the very direction it was trying to take) and the path was two squares long (or one for pawn moves), then the middle square obviously contains something. Its Empty bit is cleared and its age increased.
- If the move spanned across more squares, Darkboard will then create and register a *power move*. A power move is a new pseudomove, whose piece and starting square is the same as the last failed move, but has a shorter scope, usually one square shorter than the original move unless the new destination square is certainly empty, in which case it is further

shortened until a possible target square is found. For example, if Ra1-a8 fails and enemy pieces are possible in a7, the power move Ra1-a7 is generated.

Power moves play an important role in the artificial player. When Darkboard builds a pseudo-game tree, interpreting metapositions as positions of a perfect information game, it will try to evolve the current metapositions by predicting the future umpire's messages. *When evolving a metaposition through a move that is a power move, Darkboard will assume it is a capturing move.* This reflects the common tactic for a player to try long moves, and upon hearing the umpire reject them, shorten them one square at a time until they end up capturing something.

7.3.3 Evolution with opponent moves

To evolve an information set along an opponent's unknown move means to generate every possible evolution (compatible with the umpire's next message) for every position in the set; the union of the resulting sets, barring duplicates, represents the new information set. Again, Darkboard employs an approximation of the real thing that makes the opponent more mobile than it really is. However, it guarantees that every position in the information set is still part of Darkboard's representation.

For each square, we treat each possible piece as a real, existing piece (*pseudo-piece*) and move it according to its rules, just like we generate pseudolegal moves for Darkboard. To this end, a support chessboard is employed, which starts out without any enemy pieces on it. For each possible move, the corresponding piece bit is set on the destination square of the support chessboard. When this phase is over, a bitwise OR operation between the source metaposition and the support chessboard returns the intended evolution.

This is arguably Darkboard's most processor-intensive task, as the number of potential opponent-controlled squares normally outnumbered the number of friendly pieces. In fact, it can be easily seen that on a chessboard of size k and k^2 squares, and assuming that the number of potential opponent squares is, most of the time, $O(k^2)$, with pieces able to move $O(k)$ squares in one or more directions, the resulting complexity is $O(k^3)$.

The algorithm performs a few additional refining steps in the process, among which are the following.

- If the opponent captured a piece, every bit for that square is cleared, including the Empty one, before doing anything else. Also, the pseudo-pieces are only permitted to move to the targetted square, meaning that after the algorithm has run, the square will contain exclusively the piece types that could be responsible for the capture. This can prove useful if the attacking piece is immediately captured back (though, currently, Darkboard does not try to guess which pieces it captures).
- As a corollary of the above point, if a capture takes place but the umpire had mentioned no pawn tries for the opponent, pawns are not considered.
- If a pawn is a potential capturing piece, the minimum pawn count for the adjacent files is decreased by one and the maximum pawn count for the target file is increased by one.
- If a square has the Empty bit not set, meaning that it certainly contains an enemy piece, but at least one move (for any of the possible pseudo-pieces on that square) can move it away from there, the Empty bit is set, otherwise it stays unchanged.
- A pseudo-king will never move to squares that are certainly threatened.
- If the move causes a knight check, only knights are moved.
- If the move does not cause a check, the squares around the friendly king are cleared off the appropriate piece bits (queens and bishops on its diagonals, etc.)
- Pawn control applies normally to pseudo-pieces to at least limit their tendency to behave like “ghosts” that other pseudo-pieces can move through.
- After the algorithm has run, each square is checked. If it is certainly empty, its age is set to 0; else, if it is assuredly non-empty, its age is

set to a high, hard-coded constant value; else, its age is increased by one. Also, the total age field is recomputed and updated.

As mentioned, enemy pseudo-pieces are only blocked by friendly pieces, squares with the Empty bit not set and pawn control. This approximation leads to a weak interpretation of the first two or three moves, wherein pieces are assumed to be able to develop faster than they actually can. However, by the time the first umpire message arrives, the situation will have sufficiently stabilized, and no special treatment seems to be necessary for the very first few moves.

7.4 The move selection routines

Darkboard's core is the move selection algorithm. The main purpose of information sets, and by extension metapositions as well, is to make the construction of a game tree possible even in the context of imperfect information games. In the previous sections several functions have been described that model the possible transitions between metapositions and their evolutions. Such functions can be used to generate child nodes from root nodes, representing both the player and the opponent's moves. The selection algorithm will therefore construct a (pseudo-)game tree and use it to determine its next move.

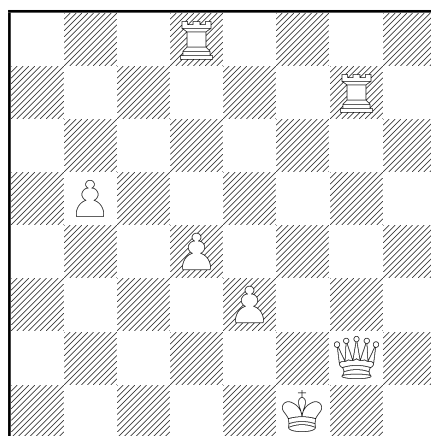
When dealing with chess, the function is most often a minimax. The reasons why minimax does not apply to Kriegspiel have been given in the past chapters, and need not be repeated in full. What matters here is how to proceed in evaluating a metaposition tree to obtain a move that is not necessarily the *best* (which is an empty word in this game as a whole), but a *reasonable* one.

The first fact to consider is that game trees assume the existence of an evaluation function. Thus, it appears there will be some function

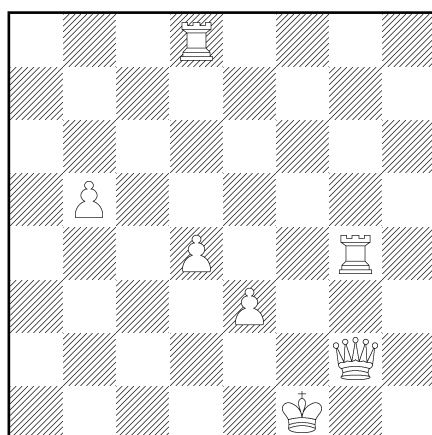
$$f : (\mathbf{Mt} \times \mathbf{B}) \rightarrow \mathbf{R}$$

that evaluates metapositions, also accepting a boolean representing whose turn it is to move. But, on closer inspection, the above definition, taken straight from chess, is not adequate for the task at hand. Chess evaluation

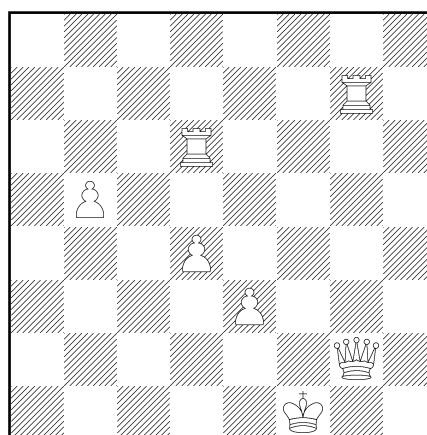
functions are built to judge on a given position, which is a snapshot of the game in progress; past events are meaningless in that context. On the other hand, it has been shown that the optimal strategy for an imperfect information game does not only depend on the current situation, but also on the events that led to it, that is the full history of the game. A Kriegspiel example is not hard to come up with. Let us consider the following metaposition, showing the White pieces only. A few Black pieces are left on the chessboard, and the umpire had nothing to say about White's latest move. How "good" was that move?



The correct answer would be, "Impossible to tell". This metaposition could be the result of many different White moves. The following diagrams shows two of them.



Rg4-g7



Rd6-d8

Depending on which move was taken, the evaluation is bound to change. As a rule of thumb, the piece that moves is also the most at risk of being captured, statistically speaking. It can be assumed that, in the first example, Rg7 is the more unsafe rook, whereas it is Rd8 in the second. However, Rg7 is protected (or rather *pseudo-protected*; protected barring unseen pieces in the middle) and Rd8 is not. Should the enemy hear a Rank check notification from the umpire following Rd6-d8, and if he plays rationally, he will try to capture the offending piece first. White's rook could then be lost with no room for retaliation. Clearly, a metaposition's backstory is important. Since it is impractical to carry the whole history of the game on every metaposition, Darkboard will just record the last move. Its evaluation function is of the type

$$f : (\mathbf{Mt} \times \mathbf{Mv} \times \mathbf{Mt}) \rightarrow \mathbf{R}.$$

In other words, it takes *two metapositions* and a move as its input. The first metaposition represents the state of the game *before* the move, whereas the second represents its evolution *after* the move. The latter could be computed by the evaluation function itself from the source metaposition and the pseudomove, but the function accepts two metapositions for two main reasons:

- Because it is not the evaluation function's job to decide how the starting metaposition should be evolved. In other words, predicting future umpire messages is a task best left to the selection algorithm, not the evaluation function.
- For performance reasons. As it will be shown, the evaluation function may be called up to twice on any metaposition, and it is desirable to compute its evolution only once for a major speedup.

7.4.1 Game tree structure

Since a metaposition's evolution depends exclusively on the umpire's messages, clearly it becomes necessary to simulate the umpire's next messages if a game tree is to be constructed. Ideally, the game tree would have to include every possible umpire message for every available pseudomove. Unfortunately, a quick estimate of the number of nodes involved rules out such an option. It is readily seen that:

- All pseudomoves may be legal (or they would not have been generated by the previous algorithms).
- All pseudomoves that move to non-empty squares can capture (except for pawn moves), and under ICC rules, we would need to distinguish between pawn and piece captures.
- Most pseudomoves may lead to checks.
- Some pieces may lead to multiple check types.
- The enemy may or may not have pawn tries following this move.

A simple multiplication of these factors may yield several dozens potential umpire messages for any single move. But worst of all, such an estimate does not even take into account the possibility of *illegal moves*. An illegal move forces the player to try another move, which can, in turn, yield more umpire messages and illegal moves, so that the number of cases rises exponentially. Furthermore, the opponent's metamoves pose the same problem as they can lead to a large number of different messages.

- On the opponent's turn, most pieces can be captured (all but those marked with a safety rating of 1).
- The king may typically end up threatened from all directions through all of the 5 possible check types.
- Again, pawn tries may or may not occur, and can be one or more.

For these reasons, any metaposition will be only evolved in exactly one way, and according to one among many umpire messages. This applies to both the player's pseudomoves and the opponent's hidden metamoves, so that the tree can be summarized as in Figure 7.2.

As a consequence, the tree's branching factor for the player's turns is equal to the number of potential moves, but it is equal to 1 for the opponent's own moves. This is equivalent to saying that Darkboard does not really see an opponent, but acts like *an agent in a hostile environment*. It also means that the opponent's metamove can be merged with the move that generated it, so

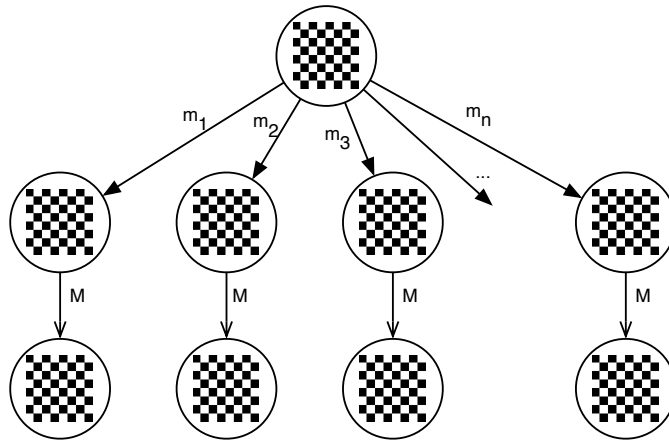


Figure 7.2: Two-ply game tree, m_1, \dots, m_n are pseudomoves, M 's represent metamoves; also denoted with different arrow heads.

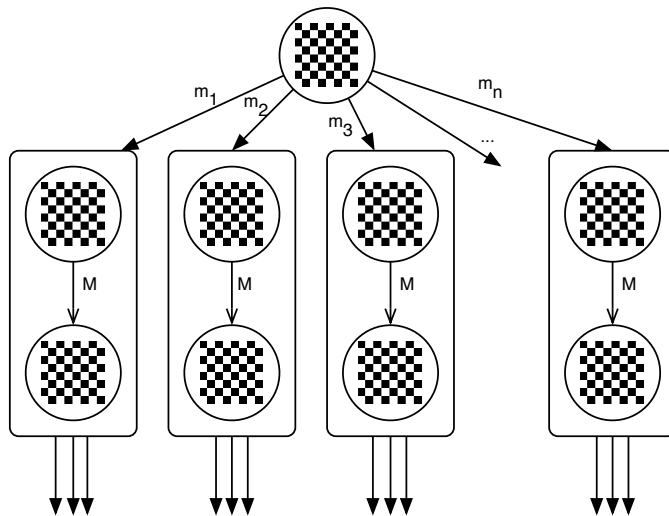


Figure 7.3: Compact form for the game tree in Figure 7.2; each node but the root contains two metapositions.

that each level in the game tree no longer represents a ply, but a full move (see Figure 7.3).

Interestingly, the branching factor for this Kriegspiel model is significantly smaller than the average branching factor for the typical chess game, seeing as in chess either player has a set of about 30 potential moves at any given time, and Kriegspiel is estimated to stand at approximately twice that value. Therefore, a two-ply game tree of chess will feature about $30^2 = 900$ leaves, whereas Darkboard's tree will only have 60. However, the computational overhead associated with calculating 60 metapositions is far greater than that for simply generating 900 chessboards, and as such some kind of pruning algorithm will be needed.

7.4.2 Umpire prediction heuristics

Bolognesi [2], in tackling the KRK endgame, where the artificial player's moves have only three possible outcomes (silent, check, illegal) and having to choose one to expand upon, relies upon the evaluation function to pick the most unfavorable option. However, even such a modest luxury seems beyond reach in the present work due to both the number of options and their different probabilities. The only remaining way is for us to propose a set of hard-coded heuristics that work well most of the time, and make sure that they will work reasonably even when they are proved wrong. Darkboard generates the umpire messages that follow its own moves in the following way.

- Every move is always assumed to be legal. Most of the time, an illegal move just provides information for free, so a legal move is usually the less desirable alternative.
- The player's moves do not generally capture anything, with the following exceptions:
 - Pawn tries. These are always capturing moves by their own nature.
 - Non-pawn moves where the destination square's Empty bit is not set, since the place is necessarily non-empty.

- Power moves obtained from previous illegal moves (see 7.3.2). This applies to the root metaposition only, as hypothetical illegal moves cannot be generated.
- If any of the above apply, the captured entity is always assumed to be a pawn, unless pawns should be impossible on that square, in which case it is a piece.
- Pawn tries for the opponent are generated if the piece that just moved is the potential target of a pawn capture.

On the other hand, the following rules determine the umpire messages that follow a metamove.

- The opponent never captures any pieces, either. The constant risk that allied pieces run is represented by danger ratings instead, which affect the evaluation function by changing the value of a piece.
- The opponent never threatens the allied king. Danger ratings encourage the king's protection.
- Pawn tries for the player are never generated.

The above assumptions are overall 'reasonable', in that they try to avoid sudden or unjustified peaks in the evaluation function. Captures are only considered when they are certain, and no move receives unfair advantages over the others. There is no concept of a 'lucky' move that reveals the opponent's king by pure coincidence, though if that happens, Darkboard will update its knowledge accordingly.

Even so, the accuracy of the prediction drops rather quickly. In the average middle game, the umpire answers with a non-silent message about 20-30% of the time. Clearly, the reliability of this method degrades quickly as the tree gets deeper, and the exploration itself becomes pointless past a certain limit. At the very least, this shows that any selection algorithm based on this method will have to weigh evaluations differently depending on where they are in the tree; with shallow nodes weighing more than deeper ones.

7.4.3 The basic decision algorithm

Now that the primitives have been discussed in detail, it is possible to describe the selection algorithm for the Darkboard player. We shall first discuss the generic version, and then introduce the pruning algorithm that makes the player efficient enough to handle fast online play on the ICC. Such separation is not only for the sake of clarity; in fact, both algorithms have their place in Darkboard, and either one is used depending on the situation. The generic algorithm makes for shallow, but exhaustive searches in the game tree, whereas the pruning-enhanced one allows deeper, but approximated exploration.

The whole stratagem of metapositions was aimed at making traditional minimax techniques work with Kriegspiel. Actually, since MIN's moves do not really exist (MIN always has only one choice) if we use the compact form for the tree, as described in the last section, the algorithm becomes a *weighed maximax*. Maximax is a well-known criterion for decision-making under uncertainty. This variant is weighed, meaning that it accepts an additional parameter $\alpha \in]0, 1[$, called the *risk coefficient*. The algorithm also specifies a maximum depth level k for the search. Furthermore, we define two special values, $\pm\infty$, as possible output to the evaluation function *eval*. They represent situations so desirable or undesirable that they often coincide with victory or defeat, and should not be expanded further.

The selection algorithm makes use of the following functions:

- *pseudo*: $(\mathbf{Mt} \times \mathbf{Mv}) \rightarrow \mathbf{Mt}$, which generates a new metaposition from an existing one and a tentative move, simulating the umpire's responses as described in the last section.
- *meta*: $\mathbf{Mt} \rightarrow \mathbf{Mt}$, which generates a new metaposition simulating the opponent's move and, again, virtual umpire messages.
- *generate*: $\mathbf{Mt} \rightarrow \mathbf{Vector}$, the move generation function.
- *eval*: $(\mathbf{Mt} \times \mathbf{Mv} \times \mathbf{Mt}) \rightarrow \mathbf{R}$, the evaluation function, accepting a source metaposition, an evolved metaposition (obtained by means of *pseudo*), and the move in between.

```

function value (metaposition met, move mov, int depth) : real
begin
  metaposition met2 := pseudo(met, mov);
  real staticvalue := eval(met, mov, met2);
  if (depth ≤ 0) or (staticvalue = ± ∞)
    return staticvalue
  else
    begin
      //simulate opponent, recursively find MAX.
      metaposition met3 := meta(met2);
      vector movevec := generate(met3);
      real bestchildvalue := maxx∈movevec value(met3, x, depth-1);
      //weighed average with parent's static value.
      return (staticvalue*α)+bestchildvalue*(1 - α)
    end
  end.

```

Figure 7.4: Pseudocode listing for value function.

The algorithm defines a *value* function for a metaposition and a move, whose pseudocode is listed in Figure 7.4. The actual implementation is somewhat more complex due to optimizations that minimize the calls to *pseudo*.

It is easily seen that such a function satisfies the property that a node's weight decrease exponentially with its depth. Given the best maximax sequence of depth d from root to leaf m_1, \dots, m_d , where each node is provided with static value s_1, \dots, s_d , the actual value of m_1 will depend on the static values of each node m_k with relative weight α^k . Thus, as the accuracy of Darkboard's foresight decreases, so do the weights associated with it, and the engine will tend to favor good positions in the short run.

Parameter α is meant to be variable, as it can be used to adjust the algorithm's willingness to take risks. Higher values of α lead to more conservative play, whereas lower values will tend to accept more risk in exchange for possibly higher returns. Generally, the player who is having the upper hand will favor open play whereas the losing player tends to play conservatively to reduce the chance of further increasing the material gap. Material balance and other factors can therefore be used to dynamically adjust the value of α

during the game, though this feature is largely untested in Darkboard as of yet.

7.4.4 The enhanced decision algorithm

The previous algorithm suffers from serious performance issues if forced to push its search 3 or more levels down the game tree. For this reason, it is used with a default depth level of 2, and is called upon when any of the following apply:

- The umpire announced captures, checks or pawn tries. Statistics show that all of the above tend to happen in clusters, so that the likelihood of a capture following another capture is much higher than normal. Since our usual assumptions about future umpire messages may not prove reasonable anymore under such circumstances, a deep analysis appears fruitless here, and a shallow, but complete and fast search seems more convenient.
- Under tight time control. Darkboard has a built-in time control manager, and will try to avoid running out of time any way it can. There are several precautions the engine takes under different stress levels, such as reducing the number of metapositions it searches through, but as a last resort, when time is running very short, Darkboard will switch to the shallow but faster search and use it until time climbs back to a safe level.

The enhanced algorithm must necessarily discard some branches of the tree and concentrate on the most promising ones in order to delve deeper into the tree. As a consequence, simple depth-limited recursion does not suffice here, and instead *the number of evaluated metapositions* is used to estimate how far to push the search.

The concept of *killer moves* is well-known in the literature of artificial chess players [1]. A move that has been found to be advantageous somewhere in the game tree is likely to be a strong move even in a different context. Chess programs combine killer heuristics with *alpha-beta pruning* to largely reduce the number of positions that need evaluating. Unfortunately, pure alpha-beta pruning is not applicable to a maximax Kriegspiel

tree; however, something resembling killer moves seems to be more feasible. The fact itself that Kriegspiel's branching factor is quite large also means that most metapositions belonging to the same tree will share many common moves. The algorithm should evaluate each move the first time it occurs, and remember good moves when they occur again.

The pruning-enhanced algorithm makes use of an external table for the purpose of remembering previous moves. Such a table can be queried for a move and return its expected benefit. It can be structured as a hash table, though this is not technically necessary, as the number of possible chess moves is not particularly large. The table's maximum size would be 64 (number of possible source squares for the move) times 64 (number of possible destination squares) times 6 (number of possible piece types), an easily manageable number. The table is hence represented as a directly-accessible array, with the wasted space outweighed by faster performance. The table supports the operations *hasEntry* to find out if a move is in the table, *getEntry* to find its value and *putEntry* to insert one, as well as the procedure *clear*, called to clean up after a move has been chosen.

Other than the familiar α , we introduce two further integer coefficients: *newMoves* and *oldMoves*. These coefficients represent the number of branches that will be expanded. At most *newMoves* branches will be explored whose associated moves do not yet appear in the table; and at most *oldMoves* will be explored among those that already do. The algorithm also accepts a *maxPositions* argument that specifies how many metapositions should be, at most, evaluated, though this is just an estimate and the program's execution will not stop once that number is met.

A simplified listing is given in Figure 7.5; the real version is both longer and more complicated, accepting more parameters to allow major performance gains due to repeated *eval* and *pseudo* calls.

This function can typically reach between four and seven levels deep into the game tree with *maxPositions* set to 5000, *newMoves* set to 5 and *oldMoves* set to 3, bringing good results in practice; however, it is currently the object of further research, mainly on the following points:

- It is not clear if and how table entry values should be amended over time for greater accuracy, perhaps adjusting the entries with the outcomes

of subsequent uses of the same move.

- Empirical tests have shown that the best values for *newMoves* and *oldMoves* change during the game. Also, this function does not behave optimally when the player is close to checkmating the opponent.

7.5 The evaluation function

Generally speaking, the evaluation functions for most chess programs have three main components: material count, mobility and position. However, since captures are an ever foggy matter in Kriegspiel, Darkboard's *eval* needs to introduce equivalent, but different concepts. Darkboard's evaluation function has three main components that it will try to maximize throughout the game: *material safety*, *position*, and *information*.

7.5.1 Material safety

Material safety is a function of type $(\mathbf{Mt} \times \mathbf{Sq} \times \mathbf{Bool}) \rightarrow [0, 1]$. It accepts a metaposition, a square and a boolean and returns a safety coefficient for the friendly piece on the given square. The boolean parameter tells whether the piece has just been moved (as it is clear that a value of *true* decreases the piece's safety). A value of 1 means it is impossible for the piece to be captured on the next move, whereas a value of 0 indicates a very high-risk situation with an unprotected piece.

It should be noted, however, that material safety does not represent a probability of the piece being captured, or even an estimate of it; its result simply provides a reasonable measure of the urgency with which the piece should be protected or moved away from danger.

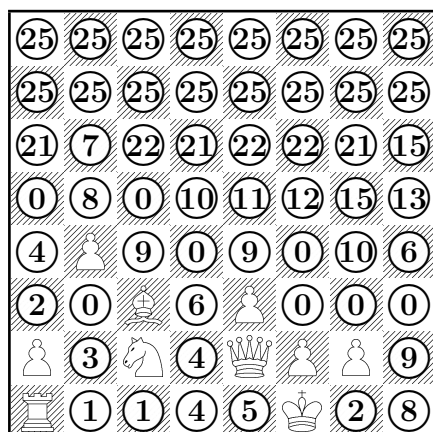
Material safety is obtained by means of a support function, *material danger*. It is a function with the same contract as material safety, but with inverted meaning, wherein 0 means no danger and 1 indicates the highest danger level. Material danger is rather easy to calculate; let us see a practical example.

```

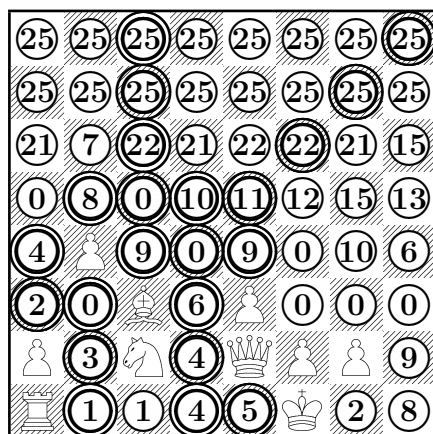
function value2 (metaposition met, move mov, int maxPositions) : real
begin
  metaposition met2 := pseudo(met, mov);
  real staticvalue := eval(met, mov, met2);
  if (maxPositions ≤ 1) or (staticvalue = ± ∞)
    return staticvalue
  else
    begin
      //simulate opponent, recursively find MAX.
      metaposition met3 := meta(met2);
      vector movevec := generate(met3);
      vector old, new, selected;
      //separate old and new moves.
      foreach x ∈ movevec do
        if hasEntry(x) then add(x, old) else add(x, new);
      //add entries to the table for the new moves.
      //their scores are the difference with their parent's eval.
      foreach x ∈ new do
        putEntry(x,eval(met3, x, pseudo(met3, x)) - staticvalue);
      //sort the two move vectors with their values in the table.
      sortx∈new with getEntry(x);
      sortx∈old with getEntry(x);
      maxPositions -= vectorSize(new); //update position count.
      //put the best from either vector into selected, and expand.
      putIntoVector(new, selected, newMoves); //up to newMoves elements.
      putIntoVector(old, selected, oldMoves); //up to oldMoves elements.
      maxPositions /= vectorSize(selected); //split maxPositions equally.
      //now proceed just like the simpler algorithm.
      real bestvalue := maxx∈selected value(met3, x, maxPositions);
      //weighed average with parent's static value.
      return (staticvalue*α)+bestvalue*(1 - α)
    end
  end.

```

Figure 7.5: Pseudocode listing for the pruning-enhanced function.



The diagram shows the age matrix values for the current metaposition. The sum of all age values, stored with the metaposition itself, is 703. In order to find the danger rating for a piece, it suffices to sum the age values along the eight directions (plus knight movement pattern). For example, if we were to calculate the danger rating for the c3 bishop, we would consider the following squares.



The sum of the values for the highlighted squares is 220, and the ratio $220/703$, which amounts to approximately 0.313, is the basic danger level for that piece. There are further modifiers; for example, if the piece were to be the possible target of a pawn try (which is not possible in the example, as d4 is being watched by a pawn), the danger level is increased by the inverse of the pawn envelope's size (that is, if there are 4 possible squares for that

pawn to be in, danger level is increased by 0.25). Also, danger is magnified if the piece has just moved.

In order to compute safety from danger, it is necessary to construct an external data structure, the *protection matrix*. The elements of the matrix are the number of pieces that can attack or defend a given square; moreover, the structure can remember the least important piece defending the square. This is important because, just like in chess, it is more desirable for a piece to be protected by a less important piece than itself – even more so in Kriegspiel, and especially in the first half of a game. Since it is impossible to know how many opposing pieces threaten a square, recapturing moves are exposed to a high risk of further captures, and clearly not every piece is equally good at protecting because of its own high value.

For this reason, each piece has a different *protection coefficient* $\rho \in [0, 1]$. When a piece is protected by some other piece, its danger rating is multiplied by the protecting piece’s protection coefficient. This is highest for pawns and kings and lowest for kings. The coefficients are, however, variable over time, and they all tend to 0 as the number of opposing pieces left decreases. At the end of the game, when only the enemy king is left, they are all equal to 0 (with only the king left, any kind of protection nullifies any danger).

After the process described in the last few paragraphs, material safety is easily computed as 1 minus danger rating. At this point, each piece’s value is multiplied with its safety and the resulting sum constitutes the metaposition’s material rating. In other words,

$$\text{Mat} = v(p_1) * f_{\text{safety}}(p_1) + \dots + v(p_k) * f_{\text{safety}}(p_k).$$

7.5.2 Position

Darkboard includes the following factors into its evaluation function:

- A pawn advancement bonus. In addition, there is a further bonus for the presence of multiple queens on the chessboard.
- A bonus for files without pawns, and friendly pawns on such files.
- A bonus for the number of controlled squares, as computed with the protection matrix. This factor is akin to mobility in traditional chess-playing software, but its usage in Darkboard is still rather unrefined; in

particular, setting this weight too high will cause the pieces to scatter excessively all over the chessboard, weakening the defensive structure. Practical results show that this factor should vary over time and depending on who is winning.

In addition, the current position also affects material rating, as certain situations may change the values of the player's pieces. For example, the value of pawns is increased if the player lacks sufficient mating material.

An additional component is evaluated when Darkboard is considering checkmating the opponent. A special function represents perceived progress towards winning the game, partly borrowed from [2]. A king matrix with the following values serves to identify the squares where the enemy king is easiest to mate, with lower values being better.

1.0	1.2	1.4	1.6	1.6	1.4	1.2	1.0
1.2	1.7	2.0	2.4	2.4	2.0	1.7	1.2
1.4	2.0	2.7	3.5	3.5	2.7	2.0	1.4
1.6	2.4	3.5	4.2	4.2	3.5	2.4	1.6
1.6	2.4	3.5	4.2	4.2	3.5	2.4	1.6
1.4	2.0	2.7	3.5	3.5	2.7	2.0	1.4
1.2	1.7	2.0	2.4	2.4	2.0	1.7	1.2
1.0	1.2	1.4	1.6	1.6	1.4	1.2	1.0

A bonus is then computed from the sum of the matrix values for the enemy king's possible positions, which grows as the sum itself tends to shrink. It is then multiplied by a *checkmate coefficient*, which represents how seriously Darkboard is trying to mate. Such a coefficient is worth 1.0 when only the opposing king is left, but drops very quickly as the number of enemy pieces grows. The coefficient is also artificially set to zero when:

- The artificial player has a negative material balance, that is, when it is losing.
- The player currently lacks sufficient mating material.

The explanation is simple: with a checkmate coefficient of 1, Darkboard puts all of itself into reducing the enemy king's territory, practically forgetting about anything else. This is rather pointless if mating looks unlikely or utterly impossible at the moment.

7.5.3 Information

Darkboard will attempt to gather information about the state of the chessboard, as the evaluation function is designed to make information desirable (precisely, it is designed to make the lack of information undesirable). Darkboard's notion of information gathering coincides with reducing a computable function, which the program calls *chessboard entropy* (E). This definition is not directly related to those used in physics or Information Theory, but its behavior resembles that of an entropy function in that:

- The function's value increases after every metamove from the opponent, that is $(m_2 = \text{meta}(m_1)) \Rightarrow E(m_2) \geq E(m_1)$.
- The function's value decreases after each pseudomove from the player, that is $(m_2 = \text{pseudo}(m_1, x \in \mathbf{Mv})) \Rightarrow E(m_2) \leq E(m_1)$.

Therefore, chessboard entropy is constantly affected by two opposing forces, acting on alternate plies. We can define $\Delta E(m, x)$, $m \in \mathbf{Mt}$, $x \in \mathbf{Mv}$ as $E(\text{pseudo}(\text{meta}(m, x))) - E(m)$, the net result from two plies. Darkboard will attempt to minimize ΔE in the evaluation function. In the beginning, entropy increases steeply no matter what is done; however, in the endgame, the winner is usually the player whose chessboard has less entropy.

Entropy is computed as follows, using a set of constant values as well as the age matrix. For each piece and each square, a negative constant is given representing how undesirable would be to have that piece on that square. These values are generally small, with the exception of enemy pawns on the player's second or third ranks, close to promoting (a highly undesirable situation). In this way, to each square is associated an *undesirability value*, defined as the sum of the negative constants for any enemy piece whose existence is possible on that square. Actually, Darkboard speeds up the process by precalculating those sums for each and every combination of enemy pieces to be found on a square.

It is then given a function $f_E : \mathbf{N} \rightarrow [0, 1]$, monotone and non-decreasing, with the constraint $f_E(0) = 0$. The parameter in f_E is the age matrix's value for a given square, and the function itself models the increase in uncertainty over time. The entropy for a metaposition is then computed as the sum, for

each square, of that square's undesirability value multiplied by $f_E(x)$, where x is the square's age value. It is easily seen that any function matching the mentioned constraints satisfies the two properties given in the beginning. As *pseudo* increases age values, entropy will increase; and as *meta* clears squares, entropy decreases.

The choice of function to be used as f_E deserves separate research. Possibly, it could even be used as a form of *opponent modelling*, in conjunction with variable piece undesirability to try and foresee where the opponent could be moving next. The f_E that Darkboard currently uses is rather primitive, being a simple capped linear function. In other words, it is given a constant $c > 0$ such that

$$E_f(x) = \begin{cases} \frac{x}{c}, & \text{if } x \leq c, \\ 1, & \text{if } x > c. \end{cases}$$

After some testing, the acceptable value of 25 was found for c . This part of the engine is obviously in need of further refinement, as other function types may indeed prove more successful.

7.6 Other components

7.6.1 The opening book

From the statistical data gathered so far, it appears that it generally takes anywhere from 2 to 15 or more moves from the start of the game before the umpire stops being silent to actually offer useful information. Clearly, any previous strategy by either player is based on no information at all. Still, not every strategy seems equally favorable even in total darkness, which rules out the option of purely random play. A different choice could be to simply let the evaluation function decide on the strategy from the very start; however, since *eval* is deterministic, this would lead to the same sequence of moves repeated every single time. In the event of multiple games against the same opponent, or with the option for future opponents to examine Darkboard's past games (which is possible on the ICC), such a repetitive pattern would be easily exploited to create vulnerabilities. It becomes necessary to use an opening book.

Some human players on the ICC tend to employ generic chess openings

when playing Kriegspiel, probably because they are most familiar with those openings; others play moves that would not make sense in chess. In particular, trying to move all of one's pawns by two squares is a very common strategy. It is, in general, difficult to prove how advantageous a certain opening is, since it also depends on the opponent's moves, and unlike chess, there are no immediate counter-moves to be played to defend from a certain opening move. In this context, variability and unpredictability seems to be the truly crucial point.

As a consequence, the natural choice was to compile an opening book from the game database, using move sequences actually played by humans. The openings were extracted with the following criteria.

- First, the program selected the top 10 players in the database. Ranking the players posed the problem of how to evaluate their performance. As mentioned, the Internet Chess Club does not have a specific ELO rating for Kriegspiel, but just a generic rating shared among all the supported chess variants. Most of these are perfect information games, too different from Kriegspiel for the score to be any reliable. Hence, a custom function was used that multiplies the number of games played by the player's victory ratio. In this way, a player will need to have both a high game count and a high victory count to be selected.
- The database is queried for the games played by the top 10. Only victories through checkmates are selected, which does not guarantee at all that the opening will be good, but makes, once again, a reasonable assumption.
- The games are parsed, and the opening lasts exactly until the umpire's first non-silent message. All openings shorter than 7 moves are discarded. The remaining are split between White and Black openings, and saved to file for later use. A random opening will then be chosen at the start of a game, taken from the corresponding color's pool.

This procedure, too, needs further refining, for two main reasons. The first one is a well-known issue in chess software, and happens when the program's evaluation function does not like the opening book's decisions. Occasionally, the evaluation function may try to undo what the opening book did,

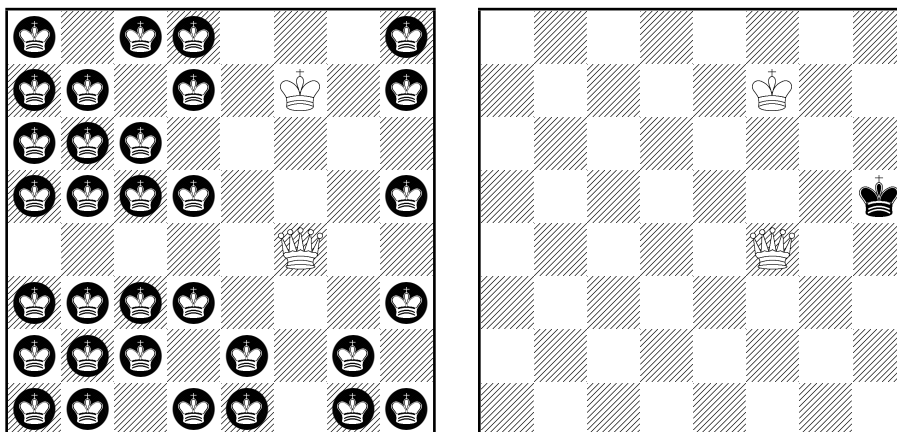
as soon as it takes over control. The problem is more subtle than in chess, as neither the opening book nor the evaluation function need be 'right' or 'wrong', but simply playing different strategies. A possible solution here would be to test the openings against the evaluation function, discarding those which are openly contrasting.

The second problem is that the opening book leads Darkboard to basically defensive play. The reason here lies in the opening themselves, or rather in our length threshold. By selecting only the openings that are longer than x moves, we are considering games where both players were playing defensively, so that captures and pawn tries were delayed. If Darkboard attempts to imitate these games, it will also copy their overall defensive attitude, as opposed to expansion and early conquest of the center. On the other hand, short openings may not be meaningful, and leave the artificial player very puzzled as soon as they end abruptly. Therefore, the problem is still open.

7.6.2 Stalemate detection

Stalemate is an additional challenge in Kriegspiel, unlike regular chess where it can be predicted with ease. As it is impossible to generate the opponent's move, it is also difficult to estimate when the opponent has run out of moves; it is even more unfortunate that stalemate occurrences are directly proportional to the amount of friendly material on the board, meaning that it is easy to turn a major victory into a draw (possibly, also a reason why it could be more convenient, at times, to promote pawns to something other than queens in the endgame). Even human players face this problem, even though the statistics do not show it fully because most humans tend to resign when they are left with the king alone. If a Kriegspiel world championship existed, we would probably see much more stubborn defense.

Bolognesi [2] deals with the stalemate issue when using metapositions to solve the KQK endgame (king and queen versus king), as stalemate may occur frequently in this endgame. For example, the following metaposition is a potential stalemate.



In order to eliminate, or at least reduce the risks, the engine must penalize heavily this kind of metaposition. While the algorithm for doing so is rather easy (it merely looks for 'singleton' kings without neighbors), its computational cost is not negligible, since it needs to be repeated for every single metaposition. For this reason, Darkboard only performs the test when two or fewer opposing pieces are left, other than the king. It would not make much sense, either, to check for stalemate when the opponent clearly has plenty of movement options left.

Of course, it goes without saying that the algorithm in [2] does not deal with multiple opposing pieces like Darkboard's. Mistakes still happen when the king is not alone, though they are not always easy to predict, even for humans.

7.7 Tuning the evaluation function

Darkboard's evaluation function makes use of a set of variable values to decide the relative weights of each component taken into consideration. These include tentative values for each piece type as well as positional values (such as protection bonuses, pawn advancement modifiers and pseudomobility). These particular functions were chosen because they appeared to be reasonable and justified by statistical analysis of the database or analogous results in the game of chess, where safely applicable. For example, higher pseudomobility is usually related to higher chances of victory, and the evaluation function takes this into account by favoring positions with more controlled

squares. However, an excessive weight given to this component would lead to the player's pieces being scattered and its defense jeopardized, which is disadvantageous in the early stages of Kriegspiel, where multiple protection is essential to gain a lead in material.

The matter of assigning values to pieces is, in itself, delicate. In many instances throughout a game, piece *count* appears to be more important than their actual *type*. The issue is discussed by Magari [22, 23]. The author concludes that in basic Kriegspiel, the values for the various pieces would not be overly different from the established ones for chess (pawn=1, knight=3, bishop=3, rook=5, queen=9), except that bishops and rooks would be almost equalized, the former being worth more than 3 and the latter just above 4 (except in the endgame, wherein the rook can mate and the bishop cannot). However, the author himself warns that this solution is unsatisfactory, and piece values appear to be a much more fluid matter than in chess.

The situation is further complicated by the ruleset of choice. In particular, the worth of pawns is higher in the Cincinnati/ICC ruleset than it is, for example, according to the English rules, because of automatic pawn tries as opposed to asking for it and having to try one is successful. The former makes pawns a more effective defensive tool, as well as a better probe during attacks. It can be argued that, no matter the ruleset, pawns are worth more in Kriegspiel chess than they are in chess.

7.7.1 Evolutionary learning

It is possible to apply *evolutionary computation* algorithms to obtain values for the various weights used by Darkboard. In fact, the weights used for online play were computed in this way, though they were later changed and tweaked for more experimentation.

The idea behind evolution as a means to optimize a numeric result in the absence of a clear, algorithmic way to do so, is quite straightforward in principle and represents the computational equivalent to Darwin's evolution theory. Instead of a single weight set, we maintain an entire *population* of weight sets. Each individual is fighting against the *environment*, typically represented by a *fitness function*. Just like in Darwinian selection, the fit will survive and spread their genes, whereas the unfit will succumb. In the

computer science version, this means that the individuals which maximize the fitness function will get to generate the population for the next generation. Children will inherit their values from their parents, though mutations take place to ensure the gene pool will not stagnate. Over time, the population will then evolve to match the fitness function.

This approach has been applied before to the game of chess. We refer to the work of Fogel, Hays, Hahn and Quon [12]. Darkboard's self-learning bears resemblance to that described in the quoted paper, except that the authors perform evolution on weights for a trio of neural networks able to evaluate chess positions, whereas Darkboard does not make use of such networks. The most interesting fact about this result is that, while learning through self-play does not usually lead to significant improvement as far as play quality is concerned, it does in the case of the quoted paper (as it does in the case of [27], with application to Othello).

Darkboard's fitness function is of typically Darwinian type, with the individuals fighting one another for survival in a series of *tournaments*, the population's size being a power of 2 (this structure was chosen to avoid having to pit every individual against everyone but itself; evolutionary computation is very computationally demanding, especially due to the fitness function requiring to play whole games of Kriegspiel for each confrontation). A set number of games is played between two conflicting individuals, the winner moving on the next round, until only one is left. The winner will be the father to the greater part of the next generation, though runner-ups will also play a secondary role in this.

A child's weights are inherited from its father, though a certain percentage of its weights undergoes a mutation (multiplication by a random number). The extent of the mutation depends on factors such as the rank of the father (the children of a individual other than the winner will tend to mutate more, in order to fill the gap with the winner) and the population's size.

Running the algorithm for a few hundreds generations brought the approximated results shown in Table 7.3. It should be noted that Darkboard interprets the value of a piece as the desire to have that piece protected and safe when possible, so a piece is not awarded its value just for existing, as in chess.

An interesting avenue for the future consists of adjusting the weights

Pawn	1.6
Knight	2.8
Bishop	3.0
Rook	4.0
Queen	6.0

Table 7.3: Darkboard's piece value assignments.

according to a database of *expert play*, such as games from our database played by select users of proved skill, as opposed to the program against itself. It is a well-known fact that some chess programs can be trained by observing large numbers of grandmaster games; it is natural to ask oneself whether some of this might carry over to Kriegspiel despite the imperfect information scenario. In other words, is it possible to capture elements of a certain player's style and reproduce them to some extent? Ideally, we would want the computer to play 'like' an arbitrary ICC user whose games appear in the database. At the moment, Darkboard can only mimick a certain player's opening style.

Chapter 8

Results and sample games

The first stage in testing Darkboard's performance involved pitting the program against a random player. The current version of the engine can checkmate the random player about 86% of the time, the rest being draws by stalemate. These typically happen when the random player has one piece left and the program assumes the enemy king enjoys more freedom of movement than it actually does. Some of these situations are indeed complex even for a human to evaluate. It should be noted that, without the stalemate prevention subsystem, victory ratio would fall down to 65%, which is on par with the performance of the best Monte Carlo sampling algorithm described in [29]. This seems to suggest that against a random player, middle game strategies seem to be largely irrelevant so long as the opponent maintains some protection scheme for his or her pieces.

A more interesting testbed for the program is a *semi-random* player that moves like the random player but will always capture back when possible and exploit its pawn tries as soon as they occur. Against such a player, Darkboard currently wins 47% of the time and draws the remaining games. These games are, however, mostly drawn by the 50 move rule rather than stalemated, for the program plays the endgame with fewer pieces and hence a lower chance of accidentally trapping the enemy king. Instead, the program suffers from loops that prevent it from checkmating, even though it would have enough material to do so. Darkboard's own selection algorithm is especially weak at the KQK endgame, and if those were always counted as victories, its victory ratio would be substantially higher. In the future, custom algorithms will be

used to play well-known endgames optimally.

Unfortunately, it has not been possible to test Darkboard against other existing Kriegspiel players other than Krieg on the ICC (See Sample game 3 for a Darkboard vs. Krieg report). Hopefully, the future will bring more recognition to computer Kriegspiel and official tournaments will be held to gauge the strengths and weaknesses of the existing programs. However, Darkboard has played numerous games against human opponents on the Internet Chess Club, a few of which are listed in the following section.

8.1 Some Darkboard games

This section lists a few commented games of Darkboard played on the Internet Chess Club. Since the program saves its own moves and more tries and asks the server for the move lists after each game, the listings are given in semi-filtered extended PGN format, always from Darkboard's point of view. In general, Darkboard wins many more games than it loses, though it currently plays unrated games only, and as such the opponents may not be as motivated, and give up more easily when they could still at least manage a draw. The following games have been selected because they reflect the program's playing style; it should be noted, however, that they were played at different times, and thus the player may show different behaviors among games.

8.1.1 Sample Game 1

White	Darkboard
Black	Jrom
Date	January 10, 2006
Time	13:11:37
Outcome	1-0

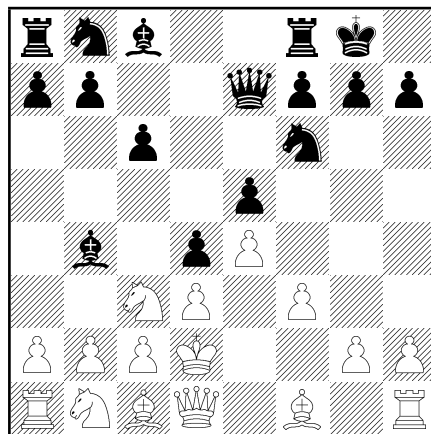
Black is not a very strong Kriegspiel player. This game does not prove to be a particularly difficult win for Darkboard, but is interesting nonetheless.

1.	e3	{:}	e5	{:}
2.	f3	{:}	d5	{:}
3.	d3	{:}	Nf6	{:}
4.	Kd2	{:}	c6	{:}
5.	Ne2	{:}	Bd6	{:}
6.	Qe1	{:}	O-O	{:}
7.	Qd1	{:}	Qe7	{:}

The White queen steps back and forth, but interestingly enough, this is not a flaw in Darkboard's engine itself, as the program is still playing from the opening book. The book itself definitely needs some further edits and openings like this one should be discarded.

8.	Nec3	{:}	Bb4	{:}
9.	e4	{P1:}	d4	{:}

Black completely ignores the first pawn try in his favor. The next diagram illustrates the umpire's view of the chessboard. From now on, Darkboard abandons the opening book and starts using its own algorithms.

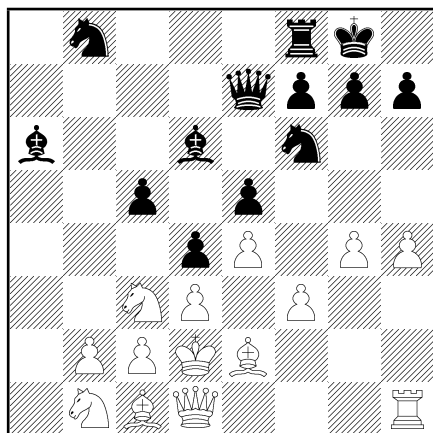


10.	Be2	{P1:}	c5	{:}
11.	g4	{P1:}	Ba5	{:}
12.	h4	{P1:}	Bc7	{:}
13.	a4	{P1:}	Bd6	{:}
14.	Ra2	{P1:}	b5	{:}
15.	axb5	{Xb5,P1: cxb3,gxf5}	a6	{:}

While Black systematically ignores his pawn tries, probably due to lack of experience, Darkboard exploits its own immediately. This version of Darkboard was equipped with an experimental constraint that forced it to capture whenever it could.

16. bxa6 {Xa6,P1:} Rxa6 {Xa6:}
 17. Rxa6 {Xa6,P1:} Bxa6 {Xa6:}

Such constraint shows some limitations here, with Darkboard sending out a lone rook into enemy territory to avenge a pawn.

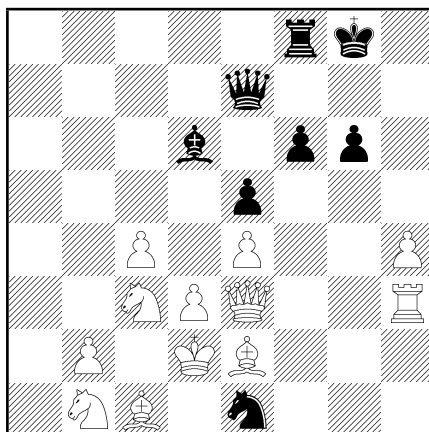


18. g5 {P1:Ke3} Bc4 {P2:}
 19. dxc4 {Xc4,P1:exd5} d3 {P2:}
 20. cxd3 {Xd3:cxd5} Nxe4+ {Xe4,CN,P2:}
 21. fxe4 {Xe4:} Nc6 {:}

Black makes a few mistakes and now White holds some advantage.

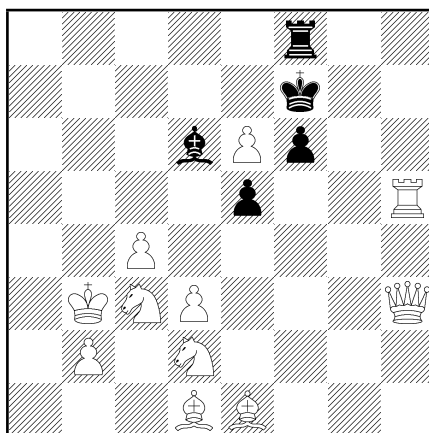
22. Qf1 {:} Nd4 {:}
 23. Qf5 {:} Nc2 {:}
 24. Rh3 {:} Na1 {:}
 25. Qg4 {:c5} Nc2 {:}
 26. g6 {P2:} Ne1 {P2:}
 27. gxh7+ {Xh7,CS:exf5,cxd5} Kxh7 {Xh7:}
 28. Qg1 {:e5} Kg8 {:}
 29. Qxc5 {Xc5:} f6 {:}
 30. Qe3 {:} g6 {:}

Following Black's new mistakes, the situation is now firmly in White's favor.



31.	h5	{P1:e5}	Kf7	{P1:}
32.	hxg6+	{Xg6,CS:cxd5,exd5,cxb5,bxa3,exf5}	Kxg6	{Xg6:}
33.	Rh5	{:e5}	Kf7	{:}
34.	Qh3	{:Rb5,Rd5}	Nf3+	{CN:}
35.	Kc2	{:Qf5,Rb5,Qc8,Qg4,Bg4}	Nh2	{:}
36.	Bd1	{:}	Ng4	{:}
37.	Nd2	{:Qf5}	Ne3+	{CN:}
38.	Kb3	{:e5,Ra5}	Nf5	{P1:}
39.	exf5	{Xf5:exd5,cxd5,cxb5,bxa3}	Qe6	{P1:}
40.	fxe6+	{Xe6,CL:dxe4,bxa3,cxb5,cxd5}	{}	{}

Black loses his queen and resigns (many players tend to do so, even though the loss of a queen is not nearly as serious in Kriegspiel as it is in chess).

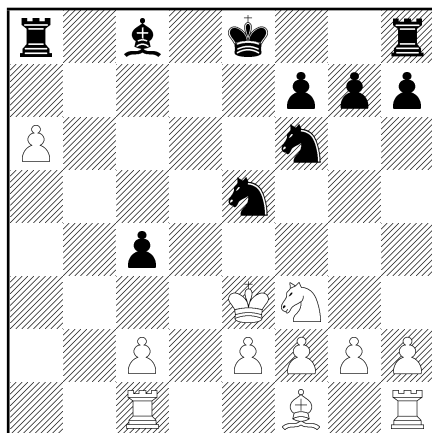


The next game is much more revealing of Darkboard's potential, with a clearly stronger opponent.

8.1.2 Sample Game 2

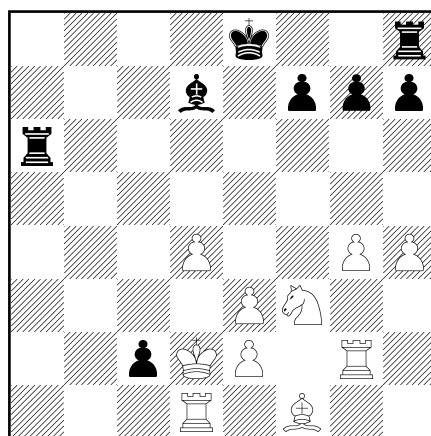
White	Darkboard
Black	NavyBlue
Date	January 4, 2006
Time	11:57:21
Outcome	1-0

1.	d4	{:}	d5	{:}
2.	a4	{:}	e5	{:}
3.	dxe5	{Xe5:bxh3}	Nc6	{:}
4.	Nc3	{:}	Nxe5	{Xe5:}
5.	Nf3	{:}	d4	{:}
6.	Kd2	{:}	dxh3+	{Xh3,CFS:}
7.	Ke3	{P1:}	cxh3	{Xh3:}
8.	Bxh3	{Xh3:}	Nf6	{:}
9.	Qd4	{Kd4:}	Bc5	{:}
10.	Rc1	{:}	Bxh3+	{Xh3,CL:}
11.	Bxh3	{Xh3:}	Qxh3+	{Xh3,CL:}
12.	Kxh3	{Xh3:}	c5+	{CS:}
13.	Ke3	{c3:}	b5	{P1:}
14.	axb5	{Xb5:}	a5	{P1:}
15.	bxh3	{Xa5:fxg3}	c4	{:}



Material-wise, White has two more pawns but one less knight. It can be seen that Darkboard takes more risks with its king than the average human. The White king has been checked repeatedly throughout the game.

16.	h4	{:c4}	c3	{:}
17.	Rh2	{:}	Nd3	{P2:}
18.	cxd3	{Xd3:gxh3,axb7,fxg3}	c2	{:}
19.	g3	{:}	Nd5+	{CN:}
20.	Kd2	{:}	Ne3	{P1:}
21.	fxe3	{Xe3:axb7,dxe4}	Be6	{:}
22.	d4	{:}	Bb3	{:}
23.	Rg2	{:Rc5,Rc3}	Ba4	{:}
24.	Rd1	{P1:Rc5}	Rxa6	{Xa6:}
25.	g4	{P1:}	Bd7	{:}

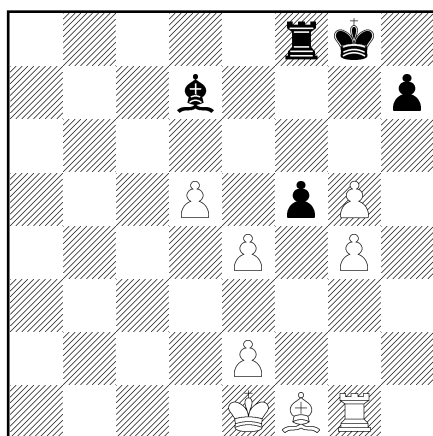


Black is ignoring the rook pawn try that would promote the c pawn for a reason. Ideally, Darkboard would realize that repeated pawn tries for the opponent that do not generate pawn tries on its own turns mean an enemy pawn on either e4 or c2, but unfortunately, the engine is not yet able to draw this kind of inference. Black is setting up a protection plan for its promoted pawn. Will this be sufficient to compensate for Black's material disadvantage?

26.	Rg1	{P1:}	Ra1	{:}
27.	Ke1	{P1:}	c1=Q	{:}

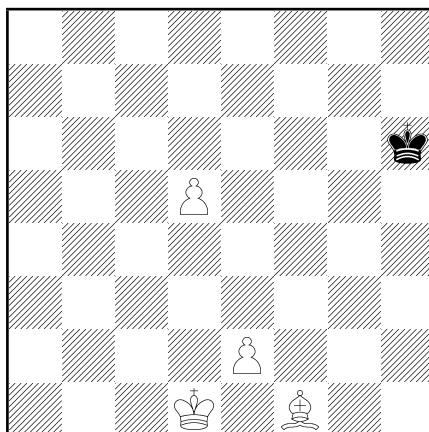
Black promotes without capturing in order to improve his or her chances of keeping the queen.

28.	e4	{:Rd2,Rd3}	Qxd1+	{Xd1,CR:}
29.	Kf2	{:Kd1,Kd2}	Qe1+	{CS:}
30.	Nxe1	{Xe1:}	Rxe1	{Xe1:}
31.	Kxe1	{Xe1:}	O-O	{:}
32.	d5	{:}	g5	{P1:}
33.	hxg5	{Xg5:exd3,exf3,dxe6,exf5}	f5	{P3:}



Black rushed the plan, and succeeded only partially. Black's last attack clashes against White's mostly intact pawn structure. Right now, quantity counts more than quality.

34.	gxf5	{Xf5:exd3,exf3,dxe6,gxh5,gxh6}	Bxf5	{Xf5,P1:}
35.	exf5	{Xf5:exd3}	Rxf5	{Xf5:}
36.	Kd2	{:}	h6	{P1:}
37.	gxh6+	{Xh6,CF:}	Kh7	{:}
38.	Rh1	{:}	Kg6	{:}
39.	Kc2	{:}	Rf7	{:}
40.	Rh3	{:}	Rh7	{:}
41.	Kd1	{:}	Rxh6	{Xh6:}
42.	Rxh6+	{:}	Kxh6	{Xh6:}



Black resigns.

8.1.3 Sample Game 3

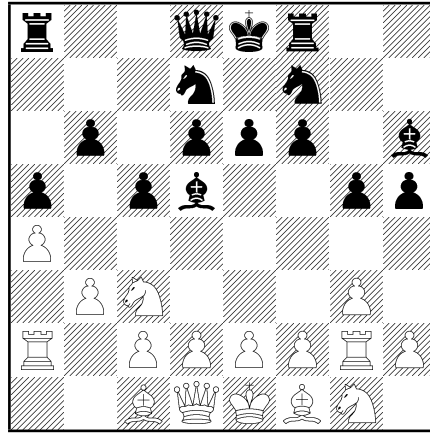
White	Krieg
Black	Darkboard
Date	January 11, 2006
Time	02:33:13
Outcome	0-1

Krieg has been previously mentioned as a resident artificial player on the ICC; this program has been playing Kriegspiel for quite a few years under various aliases, mainly Fark and Phark. Krieg is the newer, rewritten version, which Darkboard happened to challenge a few times while running on a different configuration, heavily favoring high pseudomobility at the expense of protection. As a result, Darkboard ended up at a material disadvantage in most games, yet never losing to its opponent and instead managing a couple of convincing checkmates, one of which is hereby reported. As matches between two computer players tend to drag on for many moves, a full transcript with umpire messages is omitted in favor of regular chess notation and diagrams.

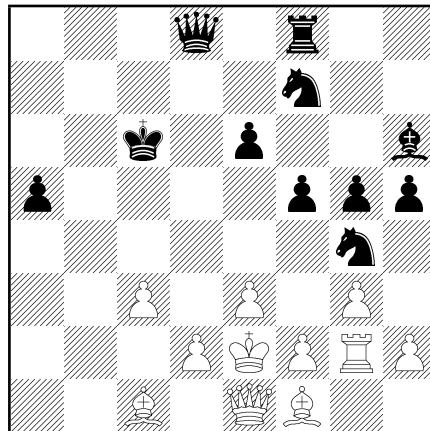
1. Nc3 h5 2. Rb1 b6 3. Nf3 Bb7 4. Rg1 Nh6 5. g3 f6 6. Rg2 e6 7. a4 g6
 8. b3 Nf7 9. Rb2 Bh6 10. Ra2 Rg8 11. Rb2 g5 12. Ra2 d6 13. Ra3 c5 14.
 Ng1 a5 15. Nf3 Bd5 16. Ng1 Nd7 17. Ra2 Rf8 18. Rb2 b5 19. axb5 c4 20.
 bxc4 Bxc4 21. Nf3 Rb8 22. Rb3 Rxb5 23. Nxb5 Bxb5 24. Rxb5 f5 25. Re5
 dxe5 26. Nxe5 Ndx5 27. c3 Ng4 28. e3 Kd7 29. Ke2 Kc7 30. Qe1 Kc6 31.

d4 Qd5 32. Bd2 Nd8 33. Kd3 h4 34. gxh4 gxh4 35. Qe2 h3 36. f3 hxg2 37.
 fxg4 gxf1=Q 38. gxf5 Rxf5 39. Be1 Bg5 40. Bd2 Qb5+ 41. c4 Qb7 42. c5
 Qh7 43. Bc3 Qh4 44. Bd2 Qff4 45. exf4 Rxf4 46. Bxf4 Qxf4 47. Qe5 Kd7
 48. Kc4 Nc6 49. Qd6+ Qxd6 50. cxd6 Kxd6 51. Kd3 Bh6 52. Kc4 Bg7 53.
 Kd3 Bh8 54. h4 Bxd4 55. Kc4 Bc5 56. Kc3 e5 57. Kb2 Bb4 58. Ka1 Bc5
 59. Ka2 Bb4 60. Kb3 Bc5 61. Ka2 Bb4 62. Kb3 Bc5 63. Ka4 Bb4 64. Kb5
 Bc5 65. Ka4 Bb4 66. Kb5 Bc5 67. Ka6 Bb4 68. Kb7 Bc5 69. Kc8 Bb4 70.
 Kb7 Bc5 71. Kc8 Bb4 72. h5 Bc5 73. Kb7 Bb4 74. Kb6 Bc5+ 75. Ka6 Bb4
 76. Kb7 Bc5 77. Kc8 Bb4 78. h6 Bc5 79. Kb7 Bb4 80. Kb6 Bc5+ 81. Ka6
 Bb4 82. Kb7 Bc5 83. Kc8 Bb4 84. h7 Bc5 85. h8=Q Bb4 86. Qd8+ Kc5
 87. Qd7 Kb5 88. Qc7 a4 89. Qb7+ Kc5 90. Qc7 a3 91. Qb7 Kd6 92. Qb8+
 Kd5 93. Kc7 e4 94. Qb7 Bd6+ 95. Kb6 Bc5+ 96. Ka6 Bd6 97. Qb5+ Bc5
 98. Qb7 Ba7 99. Qb5+ Bc5 100. Qb6 Bb4 101. Kb7 Bd6 102. Ka6 Ne5 103.
 Kb7 Nc4 104. Qc6+ Kd4 105. Qc7 e3 106. Kb8 Bc5 107. Qa7 Bb6 108. Qc7
 Bc5 109. Qa7 Bb6 110. Qb7 Bc5 111. Ka8 Bd6 112. Ka7 Be5 113. Qa6 Kc3
 114. Qb7 Kd4 115. Qa6 Bd6 116. Kb7 Be5 117. Qa7+ Kd3 118. Ka6 Bb2
 119. Kb7 Be5 120. Ka6 Bd6 121. Qb8 Be5 122. Kb7 Bb2 123. Qc7 Be5 124.
 Qc6 Bd6 125. Qa6 Be5 126. Qa8 Bb2 127. Qc8 Be5 128. Qa8 Bd6 129. Qc8
 Be5 130. Qb8 Bb2 131. Kc7 Bd4 132. Qc8 Bb6+ 133. Kb8 e2 134. Qb7 Be3
 135. Ka8 Bb6 136. Qa7 Be3 137. Qb8 Bb6 138. Qa7 Be3 139. Qb8 Bb6
 140. Qc8 Be3 141. Kb8 Bb6 142. Qc5 Ba5 143. Qe3+ Kxe3 144. Kb7 Bb4
 145. Kc6 e1=Q 146. Kd5 Nd6 147. Ke5 Kf2+ 148. Kd5 Qe4#

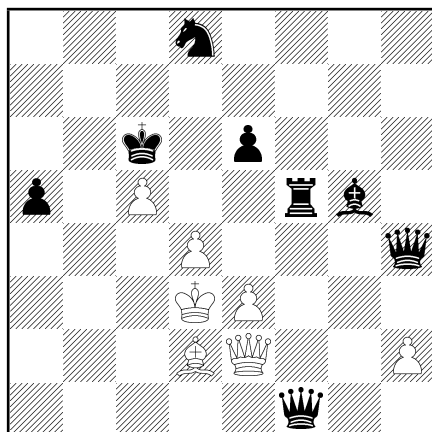
Both programs suffer from loops and lack of progress at some points, but overall Darkboard seems to possess a 'spark' that its opponent does not. Krieg plays an unusual opening compared with its common opening style as recorded in the database. After reaching a stable position, it merely waits for the opponent to attack.



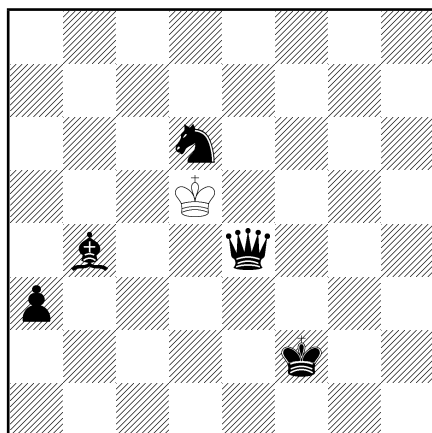
The game after 17 moves. Krieg's crescent formation is extremely defensive.



The game after 30 moves. Krieg's mainstay tactics is to never take the initiative, and simply cluster its pieces together. Darkboard's play is far more open (note the king in c6, defending the breach in Black's pawn structure from the advance of White pawns).



The game after 44 moves. Darkboard has broken through Krieg's bar-ricade. What follows is a long ballet to capture White's remaining pieces, where Darkboard will lose both queens. It will take 100 more moves to strip White of its pieces, but once Black manages the task, only five moves to checkmate the opponent.



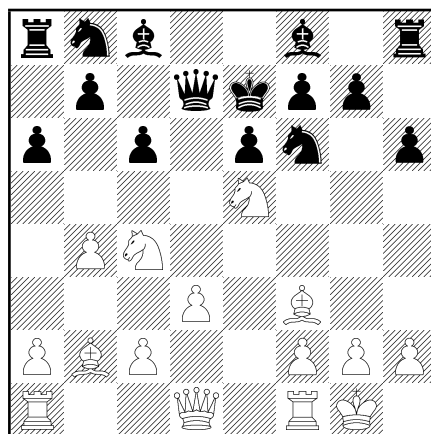
Endgame, White checkmated.

8.1.4 Sample Game 4

White	Decker
Black	Darkboard
Date	December 12, 2005
Time	21:41:46
Outcome	1-0

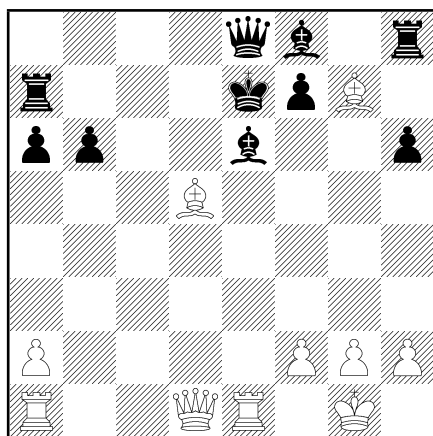
This brief selection of games ends with the record of a honorable defeat by the hand of a very skilled chess player. At the time of this writing, White is an International Master (IM) with an ELO rating of 2513. An early version of Darkboard holds its ground till the latest stages of the game, when it makes a few critical mistakes and is defeated.

1.	e4	{:}	d5	{P1:}
2.	exd5	{Xd5:}	Qxd5	{Xd5:}
3.	b4	{:}	Qe6+	{CF:}
4.	Be2	{:}	Kd7	{:}
5.	Nf3	{:}	a6	{:}
6.	d3	{:}	h6	{:}
7.	O-O	{:}	Kd6	{:}
8.	Bb2	{:}	Qd7	{:Ke5}
9.	Ne5	{:}	c6	{:}
10.	Bf3	{:}	Nf6	{:}
11.	Nd2	{:}	e6	{:}
12.	Ndc4+	{CN:}	Ke7	{:Rg8,Ng8}



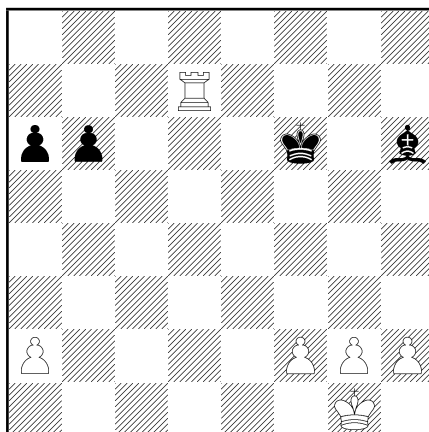
White's chess background is apparent from the way he plays this opening. Darkboard, on the other hand, seems to be confused throughout the first moves, bringing its king forwards and wasting time better spent on developing its pieces. The main culprit here is the early pawn capture on move 2, which truncated the opening Darkboard had selected from its book, forcing the main function to work from a very early stage, for which it is not well-suited.

13.	b5	{P2:}	cxb5	{Xb5:fxg6,cxd5,exd5}
14.	Ne3	{:}	Qe8	{:}
15.	c3	{:}	Nc6	{:}
16.	c4	{P1:}	bxc4	{Xc4,P1:fxg6,exd5,exf5}
17.	dxc4	{Xc4:}	Nxe5	{Xe5:}
18.	Bxe5	{Xe5:}	Bd7	{:e5}
19.	Nd5+	{CN,P1:}	exd5	{Xd5,P1:bxc6,axb5,fxg6}
20.	cxd5	{Xd5:}	Nxd5	{Xd5:}
21.	Bxd5	{Xd5:}	b6	{:}
22.	Re1	{:}	Ra7	{:}
23.	Bxg7+	{Xg7,CF:}	Be6	{:}



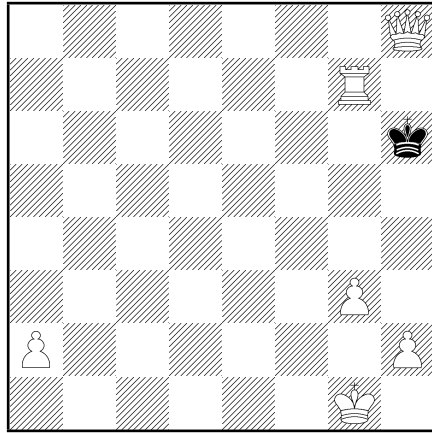
For some reason, Darkboard does not capture the bishop despite the bonus in the evaluation function. Incorrect choices such as this motivated the later change that forces it to always capture when possible. Black will now lose its rook as a result, possibly compromising the whole game. Without this kind of naive mistakes, a draw would not have been out of question.

24.	Bxh8	{Xh8:}	Kd8	{:}
25.	Bxe6+	{Xe6,CF:}	Rd7	{:}
26.	Bxf7	{Xf7:}	Qxf7	{Xf7:Rf7}
27.	Qh5	{:}	Re7	{:Ke8}
28.	Qxh6	{Xh6:}	Bxh6	{Xh6:}
29.	Rad1+	{CF:}	Ke8	{:}
30.	Bf6+	{:}	Qh7	{:}
31.	Rxe7+	{Xe7,CF:}	Qxe7	{Xe7:}
32.	Bxe7+	{Xe7:}	Kexe7	{Xe7:}
33.	Rd7+	{CR:}	Kf6	{:}



Another mistake, though of a less immediate nature. White's last move was probably his only dubious one in the whole game, sending forth his only piece without any protection. When a king is threatened, it should always first try to move in the direction it is being checked from. Black would have taken the rook by doing so, but its evaluation function does not yet handle this kind of (simple) reasoning.

34.	Rh7	{:}	Kg5	{:Kg7}
35.	Rha7	{:}	Kh5	{:}
36.	Rxa6	{Xa6:}	Bf8	{:}
37.	Rxb6	{Xb6:}	Be7	{:Kg6}
38.	Rh6+	{CF:}	Kg4	{:}
39.	Rh5	{:}	Bb4	{:Kg5}
40.	Ra5	{:}	Ba3	{:Kf5,Kh3,Kg3}
41.	Ra4+	{CR:}	Bb4	{:}
42.	Rxb4+	{Xb4,CR:}	Kh5	{:}
43.	Rb3	{:}	Kh4	{:}
44.	g3+	{CS:}	Kh5	{:}
45.	f4+	{:}	Kg6	{:}
46.	f5+	{CL:}	Kg7	{:}
47.	f6+	{CL:}	Kh7	{:}
48.	f7	{:}	Kg6	{:}
49.	f8=Q	{:}	Kh7	{:Kf7,Kf5,Kf6,Kh6}
50.	Rbb7+	{CR:}	Kg6	{:Kg7,Kg8}
51.	Rg7+	{CF:}	Kh6	{:Kf7,Kg7,Kf6}
52.	Qh8#	{:}		{:}



Checkmate. Darkboard can only regret its early mistakes, without which it could possibly have managed a draw.

Chapter 9

Conclusions and future developments

Kriegspiel is a difficult game to deal with for an artificial player. It is impossible to replace strategy with tactics in this context, and computers are traditionally weak in this respect. At the time of this writing, no software seems to exist – neither Darkboard nor any other – that can consistently challenge a real human expert. In fact, the concept of *progress* and *plan* are difficult to model in Kriegspiel. A sampling-based player's notion of progress is simply an increase in the average value of the possible game states. For Darkboard, it is its own evaluation function. None of these programs reason the way a human does; just like chess-playing software, they can tell what move they think is the best, but they cannot tell *why*, aside from stating the raw utility numbers associated with each move.

This being said, Darkboard is an important stepping stone in the right direction. First, because it proves that metapositions, when opportunely approximated, can indeed handle a full game of Kriegspiel very effectively, as opposed to a very limited endgame situations. A Darkboard metaposition is an extremely convenient data structure due to its light weight and high expressive power. Moreover, searching through a very high number of metapositions does not seem to be necessary (or even useful) in order to produce a reasonable move in most (midgame) situations. The program can play in a fast-paced environment such as the Internet Chess Club, most of its moves taking less than 3 seconds to select despite having been written in Java and running on an average personal computer.

There is still much room for improvement and much left to do. Perhaps the weakest link in the chain is the evaluation function itself. Right now what it does is try to maximize piece protection, proportionally to perceived risks, and secondly explore dark areas of the chessboard. Only when the player is close to checkmating does a notion of progress appear. The problem is that such an approach is inherently *reactive*; it tries to be prepared as best it can in the event of an attack, but does little to coordinate such an attack itself, which is what really separates an average player from a good one. Since the game tree allows the function to see and select any strategy available to the player, it *is* possible for Darkboard to improve. However, in order to do so, the evaluation function must lose its static, memoryless character.

This does not only mean that we should use different evaluation functions to deal with different endgame types, but it also means that the functions should be based on something more than just a metaposition; there should be a persistent data structure that provides an interpretation of the situation for the eval function to use, so that the same metaposition will yield different results depending on the accompanying structure – the *plan*. Plans could be made of elementary subplans, such as securing a particular square, or moving a piece to safety, or promoting a certain pawn. Mechanisms for generating and updating plans and their priority could range from a knowledge base to a neural network, and the evaluation function would turn into a *fitness* function of the metaposition with the current plans and *instincts* (default plans that are always active, such as protection instinct). This way, the searching power of metapositions would be combined with far greater potential for *dynamic* analysis; pruning would also benefit immensely, as its heuristics could involve fitness to the dominant plan in that moment.

Another area that should be taken into account is that of *opponent modelling*. The data gathered from the ICC are encouraging, showing that many among the better players make use of established patterns, especially during the opening stage. Some go as far as repeating the same sequences almost every time. Ideally, a savvy opponent will remember this and play consequently, though the usual problem of common knowledge arises: if my opponent knows I do opponent modelling, he might very well bluff to exploit my belief.

Appendix A

Kriegspiel Rules at The Gambit

These are the English rules enforced at the Gambit Club in London, as communicated by Miss E.C.Price e published in F.Dickens and G.White, Chess in Bedfordshire, Whitehead and Miller, Leeds, 1933.

Rules for Chess Kriegspiel

1. The game of Chess Kriegspiel shall be governed in every respect by the Rules of Chess comprised in the BCF Revised Law of Chess except so far as they are necessarily altered, modified or abrogated by the following rules.
2. Two players shall play on two separate boards with two separate complete sets of men and shall be so placed that neither player shall be able to see his opponent's board or men, whether the latter are on the board or removed therefrom after capture.
3. There shall be a third board with a third complete set of men, and this board and men whether the latter are on the board or removed therefrom after capture shall be so placed as not to be visible to either player. This third board shall be controlled by an umpire.
4. The men shall be placed in proper order on all three boards before the commencements of the game.
5. An Umpire shall repeat in their proper sequence on the third board the moves made by both players on their respective boards.

6. On a legal move being made by the player of the White men, the Umpire shall announce the fact to the player of the Black men by saying “White” and conversely “Black” upon the answering legal move being made.
7. The position on the third board shall govern the legality of any move.
8. If a player make a move which is illegal, the Umpire shall announce the fact; the illegal move must be retracted and a legal move made.
9. If a legal move result in a capture the Umpire shall announce the fact without stating which man has made the capture or has been captured, but shall announce the square (calculated from the side of the board occupied by the player of the captured man) on which teh capture has been made.
10. If a legal move result in a check, the Umpire shall announce the fact in the following terms:
 - (a) “Long check” if given on the one of two diagonals of which the king occupies the square common to both, which is composed of the greater number of squares.
 - (b) “Short check” if given on the other diagonal.
 - (c) “Rank check” if given on any rank.
 - (d) “File check” if given on any file.
 - (e) “Knight check” if given by a Knight.
11. When it is a player’s turn to move, he may ask the Umpire if any of his Pawns are in a position to effect the capture, and the Umpire shall reply “Yes” or “No” as the case may be. If the reply be “Yes” the player *must* try to capture with one Pawn, but if the move be illegal he is at liberty to make a legal move with any man.
12. The Umpire shall declare the game won by the player whoi effects a checkmate and shall declare the game drawn
 - (a) When neither player has sufficient mating force;

- (b) When either King is in a position of stalemate;
 - (c) When either player makes the same two successive moves three times and the position of the other player's men is such that he has no power to effect a mate unless the two said moves are varied.
13. The Umpire shall not communicate or indicate to either of the players any detail or particular of the position of the pieces or the play except those permitted by the foregoing rules.
 14. The player that deliberately views the Umpire's or his opponent's board and men shall forfeit the game.

Appendix B

Kriegspiel Rules at RAND

These are the “Kriegspiel Rules at RAND”, quoted by Ferguson as [36]: they probably date to the late '40s.

Standard chess rules, with the following additions and elucidations:

1. Personnel: Two players, referee, kibitzers.
2. Each player has a complete chess set (board, black and white men).
3. A player (e.g., W) may freely rearrange the men of opposite color (e.g., B) on his board; these men have no official role in the game.
4. The players may not see each other's boards and men.
5. A referee monitors the game (preferably using a third chess set, which neither player may see), and announces:
 - (a) whose turn is to move;
 - (b) on which squares the mover's pawns have currently valid options ('tries') to make captures;
 - (c) each rebuff ('no') experienced by the mover in attempting to move;
 - (d) the fact that a capture has taken place, where it happened, and the category of the man captured (according to the dichotomy 'pawn' or 'piece');

- (e) checks, which are announced by whichever of the following is (are) correct:
 - i. check on a long diagonal,
 - ii. check on a short diagonal (the diagonals considered are the pair which intersect at the king),
 - iii. check on a rank (or ‘horizontal’),
 - iv. check on a file (or ‘vertical’),
 - v. check by a knight;
 - (f) the fact that a pawn promotes (but not the piece promoted to or the location of the promotion),
 - (g) checkmate and stalemate.
6. The referee does not review announcements more than one move old, and does not recapitulate losses.
 7. The referee does not announce in the usual manner rebuffs obtained by attempts to move which are illegal *per se* (E.g., moves to or through a square occupied by one’s own man, diagonal movement of rook, attempts to remove check other than by moving king, interpose or capture compatible with the announced character of the check). A special rebuff (e.g., ‘hell, no’) is used here.
 8. The referee tries to eliminate errors. E.g., if the player misidentifies the square named in an announcement, the referee will correct him, if it can be done without significant information accruing to the player. Referee’s blunders range from trivial to catastrophic: the remedies include general reprimand, reverse play, declaring a game void.
 9. A player may, before moving, demand a count of the rebuffs (no’s) sustained by his opponent on the last move; he may in fact demand recapitulations before the opponent completes his move.
 10. A player may attempt any move which is compatible with his own situation (men and deployment) and with the referee’s current announcement. He is not required to have memory of previous plays, or to make logical inferences.

11. A move is completed when a piece touches the board or a presumed enemy piece on a legally admissible square.
12. *en passant* capture options are announced in the same manner as other options are announced; the fact that they are *en passant* is not specified.
13. Castling may be done in the presence of the enemy provided the king does not use or transit an affected square.
14. The referee's arrangement is always final in the event of a dispute over the position of players' pieces. Indeed, all referee's decisions are final.
15. Check by a pawn is announced as it were a bishop or a queen, but without stating it is a pawn.
16. When a check exists, only those pawn options ('tries') are announced which if taken will eliminate the check.
17. The designations used for squares are with reference to the mover, e.g., if W captures on his king's-bishop eight, the event takes place on B's king's-bishop square.
18. Kibitzers. The game is a spectator sport *par excellence*, and everything is done to keep it so. The kibitzers have the right to criticize the play, the players and the referee. However, the ethics of the situation require that the kibitzers never intentionally give useful information to the players. Probably the game breaks down as the number of kibitzers increases indefinitely; even with a half dozen, a pinned pawn has but small chance of not being found in a false try situation.
19. it is considered ethical for a player to capitalize on blunders and all unsolicited information received from referee and kibitzers; he may solicit 'information' from his opponent or otherwise heckle.
20. For the ladder games, the stalemate rule is currently modified: the stalemated player loses.

Appendix C

Kriegspiel Rules “Cincinnati style”

These are the “Cincinnati rules” according to D. Moeser [25]. They inspired the ruleset currently played on the Internet Chess Club.

The idea of Kriegspiel is this: The player on move attempts either to make a legal move, or to make moves called “tries” which have as their prime objective the obtaining of insight as to what the real position is. Any move or “try” which really is a legal move stands as that player’s actual move in the game.

On his own board, each player’s own pieces are “official”, and the Kriegspiel version of “if you touch you move” applies to them. They are for real, and their position must be identical to that shown on the Referee’s board.

The opposing pieces have no formal standing. Each player may set up pieces of the opposing color anywhere he wants, or not set them up at all; and he may move them any time he wants. (The Referee must completely ignore opposing-color material on each player’s board. It is not “official” and it may be on wrong or even absurd squares. The Referee must not allow himself to be confused by it.)

The Kriegspiel “touch” rule applies only to each player’s own material: The Player is not allowed to touch his own material except for the purpose of making a try. That is, *trying* to make a legal move. If a “try” is legal, it stands as the Player’s real move. If it is not a legal move, the Player is not required to move (or try) that piece again.

The best procedure is for the Referee to require players to acquit (let go of) the piece they make a “try” with, and to say nothing until they do so. This requirement eliminates the kind of sticky situation where the Referee blurts out some information but the player says “Oh, I didn’t really mean to play that move!” and insists on the right to take it back because he is still holding onto the piece.

After each legal move is made, and before the Opponent begins his turn, the Referee records the move, makes it on his own set, and makes the announcements required by these rules. It’s important that the Referee announce all the information required by the rules – and nothing else. All announcements must be heard by both players. Also, spectators must not talk openly about the game, for even casual comments can give away valuable information. Finally, the Referee must endeavor to be 100% correct, or else the game is likely to be ruined.

If a “try” is a legal move, the Referee simply announces that the Player has moved. “Black has moved”, he says. Or “White to move”. After a while the Ref is likely to abbreviate this notification: “White”. “Black”. “White”, he’ll say.

However, if a “try” isn’t a legal move, the Referee says “No” or “Illegal”, and that try must be retracted. If the move is impossible, and the player must know it for some reason (like trying to capture one’s own pieces), the Referee’s appropriate response is “Nonsense”. The “Nonsense” announcement discourages a Player from wasting time or attempting to use the Referee to mislead the Opponent.

The player continues to try to make a move until he finds one that is legal. When a legal move is completed, the Referee announces whichever of the following information is appropriate:

1. If a capture has been made, the fact of the capture and the square the captured piece is to be removed from. (Keep this wording in mind for an en passant capture.)
2. Whether the captured material was a pawn or a piece – but if a piece, not what kind of piece.
3. If a check has been made, the fact that the Opponent is now in check,

and the direction of check with respect to the Opponent's King: whether that King is in check on a file, on a rank, on the long diagonal, on the short diagonal, or by a Knight.

4. a. If any of the pawns of the Player on move can capture anything, the Referee announces that the Player now has a "pawn capture", which means that at least one of his pawns can legally capture something (anything) of the Opponent's. The Referee does not tell where on the board any such capture is. The Player may now try to make captures with his pawns, or he may not. He may make tries with pieces, then tries (either possible moves or possible captures) with pawns, go back to pieces, then go back to pawn tries, etc.
- b. Due to the pawn's unique capturing power (its capturing "vector" is different from its move), a Player who attempts pawn captures when the Referee has not announced there are any is trying nonsense. In this instance the Referee announces "Nonsense" so the Opponent is not unfairly confused. (For example, the Opponent would be led to believe the Player still has a lot of material on the board when he really does not.)
- c. Knotty point: If a Player is in check and has a pawn capture on the board, but no such pawn capture will remove the check, does the Referee announce the pawn capture? Answer: no.
- d. As long as at least one pawn capture is on the board, the Referee must announce that a pawn capture is possible – and continue to do so every time the Player on move has a pawn capture available. (Note: This is because announcements apply only to the move on which they are announced.)

The following rules elaborate on certain prohibitions:

1. If a piece (non-pawn) can capture something, that is not announced.
2. Promotion of pawns is not announced. Each player should be supplied with extra "promotion material" at the beginning of the game.

3. The Referee may not give a count of material on the board during the game.
4. On the Referee's board such moves as castling or pawn promotion must be made silently and without any noticeable delay, so as not to reveal the nature of these unique movements. It is in the players' interest to do likewise. This is why the players should have promotion material available at the start of the game!

Appendix D

Rules for Kriegspiel on the ICC

These are the rules for playing on the Internet Chess Club, where this chess variant has been active since 1996.

Kriegspiel (wild 16) is a chess variant in which you cannot see your opponent's pieces. You can only see your own pieces, and you have to guess where your opponent's pieces are. When you try to make a move, ICC may tell you that your move is illegal, in which case you should make another move instead.

To play Kriegspiel, just match someone for a wild 16 game:

```
match Fred 2 12 kriegspiel
```

The referee makes the following announcements where appropriate:

"White's move"

"Black's move"

"Pawn at <square> captured"

"Piece at <square> captured"

"Rank check"

"File check"

"Long-diagonal check"

(the longer diagonal from the king's point of view)

"Short-diagonal check"

(e.g. for a king on e1, the short diagonal is e1 to h4)

"Knight check"

"<number> pawn tries"

(number of legal capturing moves using pawns)

When you try an illegal move, you are simply told “Illegal move”, whether it is moving into check or moving through an enemy piece. Your opponent is not told anything when you try an illegal move.

Moves must be entered in dumb-computer format, e.g. “e2e4”. Input strings such as “nxq” which might be interpreted differently depending on the enemy position are not allowed, with one exception: “px” is allowed, to save you the trouble of trying a dozen possible diagonal pawn moves when you know that precisely one of them is legal. Other acceptable forms include “e2-e4”, “o-o”, and “f7g8=N”. Moves like “Rd3” are currently not accepted (because there are a few cases where they could be context-dependent). Most graphical interfaces generate strings like “e2e4” for you when you make moves with the mouse. (But see the note below about pawn captures under xboard and slic3).

Opponent’s moves show up in the form “?” or “?xf3”. This might break some interfaces.

You cannot observe rated Kriegspiel games. This is to prevent people from logging in with a second account, and seeing all the pieces while they are playing on another account! You *can* observe unrated Kriegspiel games, if you’re registered.

In examine mode you can see all the pieces and moves. E.g. if you have `examine=1`, at the end of the game you’ll be able to see your opponents pieces (you may have to “refresh”), review the move history, etc. The illegal moves tried are not recorded.

The clients xboard 3.4, slic3 22f, and probably some other interfaces will not allow you to even attempt a diagonal pawn move with the mouse when they can’t see a piece to take. The move is not even being transmitted to the server in this case; it’s just being rejected by the client. So with those interfaces, you should type in pawn capturing moves, e.g. “px” or “d5c4”. Naturally these problems are not the fault of the interface writers! We sprang kriegspiel on them with no warning (sorry guys). Ziics happens to work well as is.

Notes and known problems:

1. Pawn captures must be done by keyboard on some interfaces.

2. Channel 116 is the Kriegspiel channel.
3. Kriegspiel with lag is painful. Even with timestamp, your clock will run while you wait to hear that your move is illegal.
4. If you start a game and your opponent complains that he can't see your pieces, offer to abort, and suggest that he read this file.
5. If you disconnect or get disconnected during a Kriegspiel game, the game is a loss. Kriegspiel games are not adjourned.

Bibliography

- [1] S. Akl and M. Newborn. The principal continuation and the killer heuristic. In *Proc. ACM Nat. Conf.*, pages 466–473, 1977.
- [2] A. Bolognesi. Analisi e progettazione di un programma di gioco ad informazione imperfetta, 2002.
- [3] A. Bolognesi and P. Ciancarini. Computer Programming of Kriegspiel Endings: the case of KR vs K. In J. van den Herik, H. Iida, and E. Heinz, editors, *Advances in Computer Games 10*, pages 325–342. Kluwer, 2003.
- [4] A. Bolognesi and P. Ciancarini. Searching over Metapositions in Kriegspiel. In J. van den Herik and H. Iida, editors, *Computer and Games 04*, volume (to appear) of *Lecture Notes in Artificial Intelligence*, pages –. Springer, 2004.
- [5] J. Boyce. A kriegspiel endgame. pages 28–36.
- [6] J. Burger. UMPIRE: An automatic kriegsspiel referee for a time-shared computer. In *Proc. 22nd ACM National Conference*, pages 187–193, Washington, USA, 1967. Association for Computing Machinery.
- [7] P. Ciancarini. *La Scacchiera Invisibile: Introduzione al Kriegspiel*. To be published, 2003.
- [8] P. Ciancarini, F. DallaLibera, and F. Maran. Decision Making under Uncertainty: A Rational Approach to Kriegspiel. In J. van den Herik and J. Uiterwijk, editors, *Advances in Computer Chess 8*, pages 277–298. Univ. of Rulimburg, 1997.

- [9] S. Edwards. Portable Game Notation Specification and Implementation Guide, 1994.
- [10] Tom Ferguson. Mate with Bishop and Knight in Kriegspiel. *Theoretical Computer Science*, 96:389–403, 1992.
- [11] Tom Ferguson. Mate with two Bishops in Kriegspiel. Technical report, UCLA, 1995.
- [12] D. Fogel, T. Hays, S. Hahn, and J. Quon. A Self-Learning Evolutionary Chess Program. In *Proceedings of 2004 Congress on Evolutionary Computation*, pages 1427–1432, Piscataway, NJ, 2004. IEEE Press.
- [13] I. Frank and D. Basin. Search in games with incomplete information: A case study using bridge card play. *Artificial Intelligence*, 100(1-2):87–123, 1998.
- [14] D. Fudenberg and J. Tirole. *Game Theory*. MIT Press, 1991.
- [15] M. Ginsburg and S. Weisband. Social capital and volunteerism in virtual communities: The case of the internet chess club. *HICSS-35, Virtual Communities Minitrack, Hawaii, January 2002*, 2002.
- [16] <http://chessclub.net>. The internet chess club website.
- [17] <http://en.wikipedia.org/wiki/PGN>. Pgn information on wikipedia.
- [18] <http://www.cs.unibo.it/cianca/wwwpages/chesssite/kriegspiel/kriegbase.pgn>. Kriegspiel game database from the icc.
- [19] <http://www.very-best.de/pgn-spec.htm>. Pgn full file format specification.
- [20] W. Jamroga. A Defense Model for Games with Incomplete Information. In *Proceedings of the Joint German/Austrian Conference on AI: Advances in Artificial Intelligence*, pages 260–274, London, UK, 2001. Springer-Verlag.

- [21] H. Kuhn. Extensive games and the problem of information. pages 193–216, 1953.
- [22] M. Leoncini and R. Magari. *Manuale di Scacchi Eterodossi*. Tipografia Senese, Siena, 1980.
- [23] R. Magari. Scacchi e probabilità. In *Atti del Convegno: Matematica e scacchi. L'uso del Gioco degli Scacchi nella didattica della Matematica*, pages 59–66, Forlì, 1992.
- [24] G. Michael. An Interview with Charles Wetherell and Tom Buckholtz. <http://www.nersc.gov/deboni/Computer.history/Buckholtz.html>, 1999.
- [25] D. Moeser. Kriegspiel Cincinnati Style. *J'Adoube - Chess Magazine*, March 30 1977.
- [26] O. Morgenstern and J. Von Neumann. *Theory of Games and Economic Behavior*. Princeton University Press, 1947.
- [27] D. Moriarty and R. Miikkulainen. Discovering Complex Othello Strategies Through Evolutionary Neural Networks. *Connection Science*, 7:195–209, 1995.
- [28] J. Nash. Equilibrium points in n-person games. *Proceedings of the National Academy of the USA* 36, pages 48–49, 1950.
- [29] A. Parker, D. Nau, and VS. Subrahmanian. Game-Tree Search with Combinatorially Large Belief States. In *Int. Joint Conf. on Artificial Intelligence (IJCAI05)*, page (to appear), Edinburgh, Scotland, 2005.
- [30] E. Rasmusen. *Games and Information*. Blackwell, 2006.
- [31] S. Russell and J. Wolfe. Efficient belief-state AND-OR search, with application to Kriegspiel. 2005.
- [32] M. Sakuta. *Deterministic Solving of Problems with Uncertainty*. PhD thesis, Shizuoka University, Japan, 2001.

- [33] J. Maynard Smith. *Evolution and the Theory of Games*. Cambridge University Press, 1982.
- [34] C.S. Wetherell, T. Buckholtz, and K.S. Booth. A Director for Kriegspiel, A Variant of Chess. *The Computer Journal*, 15(1):66–70, 1972.
- [35] C.S. Wetherell, T. Buckholtz, and K.S. Booth. A Program to Referee Kriegspiel and Chess. *The Computer Journal*, 18, 1975.
- [36] J. D. Williams. Kriegsspiel rules at RAND, 1950.