Arimaa Is PSPACE-Hard

Benjamin G. Rin[1]
Utrecht University, The Netherlands

Atze Schipper
Utrecht University, The Netherlands

Abstract

Arimaa is a strategy board game developed in 2003 by Omar Syed, designed to be hard for AI to win because of its large branching factor. In this paper, its theoretical complexity is considered. We prove that Arimaa (suitably generalized to an $n \times n$ board) is PSPACE-hard. This result is found by reducing a known PSPACE-hard variant of Generalized Geography to a variant of Arimaa that we call Arimaa′, which in turn is then reduced to $(n \times n)$ Arimaa. Since the game is easily seen to be solvable in exponential time, it follows that its complexity lies somewhere between being PSPACE-complete and EXPTIME-complete.

2012 ACM Subject Classification Theory of computation → Problems, reductions and completeness

Keywords and phrases Arimaa, complexity theory, PSPACE-hardness, board games, Generalized Geography

Digital Object Identifier 10.4230/LIPIcs.FUN.2024.27

Acknowledgements We would like to thank Rosalie Iemhoff for her useful feedback and suggestions, as well as Colin Caret, Dominik Klein, Johannes Korbmacher, Jaap van Oosten, and all other attendees at The Utrecht Logic in Progress Series for their comments on a presentation of an early version of this paper. We also would like to thank the anonymous referees for their feedback.

1 Introduction

Arimaa is a two-player strategy game played on an 8×8 board. Like chess, checkers, and Go, it is a perfect-information, deterministic, turn-based game with at most one winner. Arimaa has been a focus of much attention in applied AI research, as it was originally designed purposely to be difficult for computers to play [18]. The present paper considers the more theoretical question, "Given a position in Arimaa and specified player, does that player have a winning strategy?" We call this the *Arimaa decision problem.* As with chess and checkers, this question is trivially solvable in constant time if the game is played on a standard 8×8 board (albeit for an impractically large constant). Accordingly, we focus instead on the question for the generalized case of an $n \times n$ board (cf. [8], [11], [12], [13]). For the aforementioned strategy games, the corresponding question is known to be PSPACE-hard.[2] The present paper shows that the same is true for Arimaa. We prove this result by reduction from the known PSPACE-hard problem Generalized Geography, adapting a technique from [16] originally used to show the PSPACE-hardness of $n \times n$ chess.[3] We do not prove here that

[1] Corresponding author. Authors listed alphabetically.

[2] Results showing PSPACE-hardness of strategy and puzzle games continue to be produced to this day, with some recent examples including [1], [3], and [4]. Depending on the details of how the game is generalized – e.g., concerning rules such as the 50-move rule in chess – the question may in fact be shown to be EXPTIME-hard (see, e.g., [8], [12], and [13]).

[3] Our proof has its origins in the bachelor thesis of the second author, written under supervision of the first [15].

$n \times n$ Arimaa is in PSPACE. However, it is not hard to see that it falls in EXPTIME.[4] In fact, we suspect it is EXPTIME-complete (see Conjecture 1), which we leave for future research.

1.1 History

In the last twenty-five years, AI has been increasingly defeating humans in strategy games such as chess and Go. This emerging trend inspired computer engineer Omar Syed to design Arimaa as a game intentionally hard for computers while still relatively easy to understand and fun to play for humans. It is played on a chess board with chess pieces, but the mean branching factor is around 17,000 moves per turn [9] (compared to about 35 for chess).

In 2003, the game was ready and published in The International Computer Games Association (ICGA) Journal [18], together with a prize of $10,000 USD for the first person or organization to create a computer program that could beat the best human players before the year 2020.[5] The challenge sparked much research into the development of Arimaa bots and the practical complexity of Arimaa from people like Brian Haskin [9], Christ-Jan Cox [5] and experienced Go programmer David Fotland [6]. Fotland had won the Arimaa computer tournament five times, but could never manage to win the contest against the best human players. Many approaches were tried by different people, resulting in numerous academic papers, theses and technical reports before the challenge was finally completed (e.g., [7], [17], [19]). In 2015, David Wu with his bot *SHARP* [22] did the unthinkable and defeated the top human players. The March 2015 ICGA Journal issue [20] was themed around the Arimaa challenge being won.

All this research is on the practical complexity of making a machine play Arimaa well. But its theoretical complexity is relatively unexplored. As the editorial board of ICGA wrote in the March 2015 issue:

> The last scientific challenge, establishing the game-theoretical value of Arimaa, is still waiting for its solution. Arimaa is now in the class of games in which computers outperform the best human players, just like chess. Yet, the ultimate question is: can we solve the game? As readers of this Journal know we distinguish between weakly solving and strongly solving a game (introduced by Allis, 1994). So, we would like to encourage all researchers to continue the development of advanced techniques to find the ultimate truth of Arimaa. [20]

1.2 Main theorem

The present paper proves the following result.

▶ **Theorem 1.** *The Arimaa decision problem is PSPACE-hard.*

---

[4] In [8] it is stated that $n \times n$ chess is decidable in exponential time by brute force. A similar case can be made for $n \times n$ Arimaa. There is an upper bound 13^{n^2} on the number of possible board states, since each of the n^2 squares must be empty or occupied by one of the six pieces of either player. Since each position is caused by one of two players, and since a position can occur no more than three total times in a game (by the threefold repetition rule), the game tree has not more than $6 \cdot 13^{n^2}$ nodes, which is exponential in the number n^2 of squares. Optimal moves can then be found by, for example, alpha-beta pruning.

[5] The prize was later raised to $12,000 USD [2].

We will reduce a known PSPACE-hard variant of Generalized Geography, which we call $\text{AGG}_{1.1}$, to a variant of Arimaa we call Arimaa′, which is then reduced to $n \times n$ Arimaa[6]. These reductions are both computable in polynomial time, so it will follow that Arimaa is PSPACE-hard.

The proof in this paper follows in the footsteps of Storer in [16] for $n \times n$ chess. Unlike chess, however, Arimaa has no pieces that can move long distances in a single turn. Accordingly, one of the main challenges of our proof, and arguably one of its main contributions, is in showing how to apply Storer's proof method for games with only short-range pieces like Arimaa.

Since it is straightforward to see that the Arimaa decision problem is in EXPTIME for reasons similar to those for chess (see fn. 4), it follows that the complexity of Arimaa lies somewhere between PSPACE- and EXPTIME-completeness. In the future, we hope to settle the following conjecture.

▶ **Conjecture 1.** *The Arimaa decision problem is EXPTIME-complete.*

2 Background

2.1 Game rules

Arimaa is played on a chess board using chess pieces, but otherwise has little to do with chess. The pieces are renamed as follows to make it easier to remember their hierarchy (listed in order of decreasing strength): *elephant* for king, *camel* for queen, *horse* for rook, *dog* for bishop, *cat* for knight, and *rabbit* for pawn. The 8×8 board remains the same except for the squares c3, c6, f3 and f6. These squares are *trap* squares, where pieces can be captured.

2.1.1 Setup

▶ **Definition 2.** *The* home ranks *of a player are the first two ranks on the board from that player's perspective.*

▶ **Definition 3.** *The* goal rank *of a player is the first home rank of his opponent.*

The board starts empty, with Gold being the first player to choose the setup in which he[7] would like to start the game. After he has positioned his pieces (this is visible to Silver), Silver may choose her starting setup. Both players may place their 16 pieces in any order they want on their home ranks. After the setup, Gold will be the first player to make a move.

2.1.2 Playing

▶ **Definition 4.** *A* position *describes all piece locations and which player is next to move. Switching pieces of the same team and strength does not result in a different position.*

Figure 1 shows a sample Arimaa position. The game is shown from Gold's point of view. Traps are represented by × symbols. For readability, we use numbers to denote the different pieces in figures throughout this paper. The numbers correspond to the pieces' places in the hierarchy – e.g., an elephant is denoted by the number 6, a camel by 5, etc. Hereafter, let us call the 1s, 2s, and 3s the *weak* pieces and the 4s, 5s, and 6s the *strong* pieces.

[6] Hereafter, we may refer to $n \times n$ Arimaa simply as Arimaa, if confusion is unlikely.
[7] For referential convenience, in this paper we (alphabetically) use "he" for Gold and "she" for Silver.

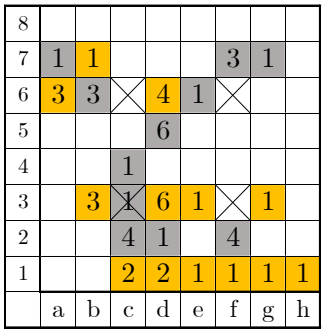


Figure 1 An example Arimaa position from [21].

All pieces move the same way, one step at a time and only in cardinal directions. The only exception to this rule is the rabbit: like the pawn in chess, the rabbit cannot move itself backward. The only way a rabbit can move back is by being pushed or pulled by an enemy piece (see below). The biggest difference from other board games such as chess, checkers, and Go is that players are allowed to distribute up to four moves per turn over their pieces. For example, one may let four pieces move one step, one piece move four steps, or anything in between, as long as the turn ends with a net change to the position. Pieces can only step onto an adjacent square if it is not occupied by another piece, unless the piece making the move is pushing the other piece away.

Pieces may *push* or *pull* a single enemy piece that is strictly lower in the strength hierarchy. This costs two steps, as two pieces are moving one square, and the piece performing the push or the pull must be directly adjacent to the enemy piece. Any square in one of the cardinal directions of the piece being pushed can be chosen to push that piece onto as long as it is able to move, and the pushing piece will move on to the square previously occupied by the pushed piece. Observe the position depicted in Figure 1. The gold dog on a6 could, for example, push the silver rabbit on a7 to a8, moving to a7 itself. Pulling is similar: the piece performing the pull can move to any square it is allowed to move to, and the pulled piece will move to the square previously occupied by the pulling piece. For example, the gold elephant on d3 could pull the silver rabbit on d2 to d3, and could then pull it to d4, ending the turn on e4 itself.

Another game mechanic is *freezing*. Any piece that is next to an enemy piece that is higher in hierarchy is *frozen* and therefore not allowed to move (thus also not allowed to push or pull). It follows automatically that the elephant, the highest ranked piece, cannot be frozen. A piece that is currently not frozen is allowed to move to a square next to a bigger enemy piece, resulting in freezing itself. As an exception, a piece can protect itself from being frozen by being next to a friendly piece. In that case, neither piece can be frozen. Likewise, a piece can unfreeze a currently frozen friendly piece by moving to a square next to it. In the example position in Figure 1, the pieces on a7, b7 and d6 are currently frozen, while the piece on c1 is not.

Another way to immobilize a piece is to *blockade* it. A piece is blockaded when it is surrounded by pieces it cannot push. Examples include the c3 rabbit in Figure 1 and the elephants in Figure 5. So, while an elephant can never be frozen, it can be blockaded if surrounded by two pieces in each direction.

The trap squares are the only way for pieces to be removed from the board (captured). Trap squares can be *controlled* by having at least one friendly piece next to it, meaning your pieces are safe to stand in the trap square without being captured. However, as soon as all

four squares next to the trap are either unoccupied or are occupied by enemy pieces, any friendly piece standing in the trap will be removed from the game. Shared trap control is thus also possible, when both friendly and enemy pieces are standing around a trap, resulting in the trap square being safe to occupy for both players.

All consequences of moving pieces are effective immediately, not at the end of the turn. For example: if a piece is frozen after making a certain move, the player is not allowed to make any more moves with that piece, even if not all four moves in the current turn are used yet. However, pushing and pulling are moves considered to happen simultaneously. This means that a player can, for instance, move a friendly piece onto an uncontrolled trap square while pulling a weaker enemy piece. Although this results in the friendly piece getting captured, the pull still gets to be completed. See, for example, the gold dog on b3: the moment it moves south next to the silver horse on c2 to pull the c3 silver rabbit west, it is frozen, but the pull gets completed anyway.

The final movement rule is the *threefold repetition rule:* a player may not end a turn having created the same position as one that he or she has previously created twice.

2.1.3 Winning conditions

There are three ways for a player to win the game:

1. Ending a turn with at least one of the player's rabbits on the goal rank.
2. Making the other player lose all his or her rabbits. If both players lose their last rabbit in the same turn, the player who took the turn wins.
3. Giving the other player no legal moves to play (say, by freezing and/or blockading all his or her pieces).

2.2 Generalization to $n \times n$ Arimaa

As already discussed, 8×8 is trivially computable in constant time, so we generalize to an $n \times n$ board. However, stretching the board to variable dimensions raises an important decision on how to place the trap squares. Standard Arimaa has four trap squares on an 8×8 board, placed on squares c3, c6, f3 and f6. We have opted to generalize this by concatenating copies of the standard board (with standardly placed trap squares) in all directions, resulting in an arbitrarily large board with many groups of four trap squares (see Figure 5 for an example). More on trap placement can be found in Section 4, where we discuss other ways of distributing trap squares over a large board.

2.3 Generalized Geography variations

Our main theorem is proved by reduction from a known PSPACE-complete variant of Generalized Geography. We present the definition of this variant in the present section.

Geography is a game in which two players take turns in naming cities. The first letter of the next city must be the same as the last letter of the previous city, and the next city must not have been named before. The first player who cannot think of a new city loses. Generalized Geography (GG) is a generalized version of the game. GG is played on a directed graph in which the two players (usually called White and Black, but we may say Gold and Silver) each take turns placing markers on nodes. The game starts with Gold placing a marker on the start node, after which it is Silver's turn. The players then take turns placing markers on nodes with an incoming edge from the last node chosen. This continues until a player places a marker on a node already marked, after which the game ends and that player loses.

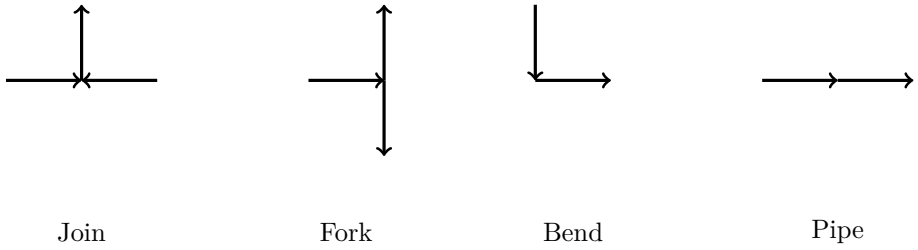


Figure 2 The four node shapes.

The problem of determining which player has a winning strategy in GG is proven to be PSPACE-complete by Schaefer [14]. In [10], some restrictions to the graph are added while preserving the complexity of the game. Storer uses this restricted definition of GG in his proof on the complexity of chess [16], where he calls it Restricted GG (RGG). Based on RGG, Storer defines another variant called Array GG (AGG), wherein further restrictions are added and the winning condition is reversed: a player who marks a previously marked node now *wins* the game. In detail:

▶ **Definition 5.** *The* AGG problem *is defined as: "Given a directed graph G and start node s, with the features described below, does Gold have a winning strategy (see rules above)?"*

The nodes in G have total degree 3 or have indegree 1 and outdegree 1. The start node *s is the only exception, with indegree 0 and outdegree 1.*

The nodes of G form a finite subset of $\mathbb{Z} \times \mathbb{Z}$ *in the Cartesian plane. Edges must be either vertical or horizontal, and may only connect pairs of nodes that have no other nodes between them (see Figure 3).*

Other restrictions on the graph are:

1. *The graph is planar (no crossing edges when the graph in embedded on a plane).*
2. *The graph is bipartite (no odd-length cycles).*
3. *The graph is connected.*
4. *There are no self-loops.*

Storer proves that AGG is PSPACE-complete in [16], by showing how to transform an RGG problem (given an instance (G,s) of RGG) into an AGG problem. This is done by scaling up the graph when needed by adding new nodes (preserving the original parity of the game by adding them in pairs) and changing the length of edges. The transformation is done in such a way that we only end up with nodes of the forms depicted in Figure 2, and straight edges between them.

In Corollary 1.1 from Storer's paper it is shown that AGG remains PSPACE-complete even if we add the further restriction that every non-start node must be one of the types depicted in Figure 2, and the nodes of each type other than the bend are also in the orientations depicted. Let us call this restricted problem $\text{AGG}_{1.1}$. This means that we need only eleven different types of nodes, other than the starting node, to build any $\text{AGG}_{1.1}$ graph: the join, fork and pipe in one orientation each, and bends in all eight orientations (only north-to-east is depicted here).[8] As in AGG, edges may be in any cardinal direction. Note that there can be a *long* edge connecting two nodes that are distant in the Cartesian plane, if no node lies between them.

[8] Storer shows how a grid graph with nodes of other types can be simulated by one with only these types.

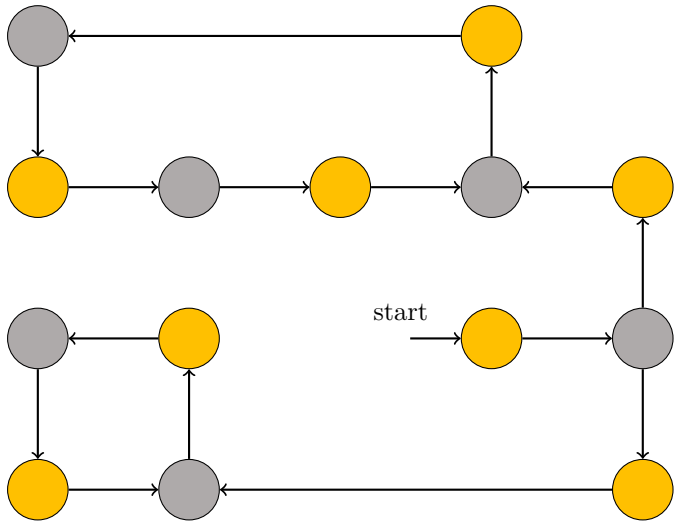


Figure 3 An $\mathrm{AGG}_{1.1}$ example graph. Node color indicates which player can mark that node.

▶ Remark 6. Because the graph is bipartite, and because players alternate turns choosing nodes adjacent to previously chosen nodes, the nodes can be partitioned into two disjoint subsets – one for each player. Since it is predefined which player will mark the start node, each node is already determined at the beginning of the game to be potentially markable by precisely one particularly player. And given any node, its incoming *edge(s)* are also deterministically assigned as potentially usable for exactly one player before the game starts. See Figure 3 for an example of an $\mathrm{AGG}_{1.1}$ graph.

2.4 Arimaa′

To simplify the proof of the main theorem, we show that a variant of Arimaa called Arimaa′ reduces to $n \times n$ Arimaa, and we then show that this variant is PSPACE-hard. We define Arimaa′ in the present section.

▶ **Definition 7.** *A* board state *is an arrangement of pieces on the board, without consideration of which player is moving next.*

▶ **Definition 8.** Arimaa′ *is the problem of determining whether Gold has a winning strategy given an $n \times n$ Arimaa board state P' under the following conditions:*

1. *There is a new threefold repetition rule: players may now legally cause a position to occur for a third time, but a player who does this immediately loses.*[9]
2. *The rule making players lose when starting a turn with no movable pieces is removed.*
3. *The rule preventing players from passing their turn without making a net change to the position is removed. Passing a turn in any position still counts as causing that position, with respect to the threefold repetition rule.*

[9] This change is strictly to make it easier to discuss repetition, so that we may sometimes say a player repeats the position and loses, rather than say the player cannot repeat the position.

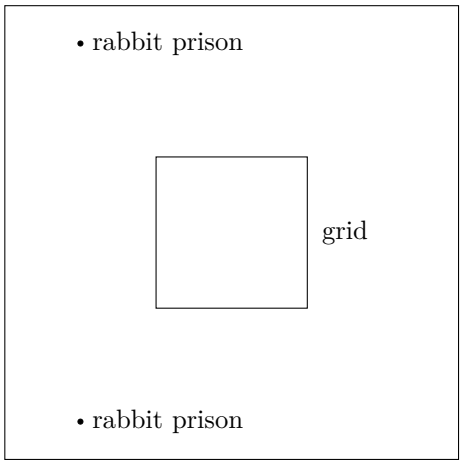


Figure 4 Overview of an Arimaa′ position. In the area labeled "grid" is a simulated $\mathrm{AGG}_{1.1}$ graph.

4. *In the board's center is a simulated $AGG_{1.1}$ graph with its start node already marked, the details of which will be specified in the proof of Theorem 1. (We will see that Gold will have a winning strategy in P' iff the Gold player in the simulated $AGG_{1.1}$ game also has a winning strategy.) Outside this area, the board is empty, save for two gadgets called* rabbit prisons *defined later. See Figure 4.*
5. *The most recent turn was Gold's and was a pass. Before that, Silver moved a piece.*

▶ Remark 9. By condition 8.3, if a player (say, Gold) moves a piece and causes a position X, after which the opponent (Silver) passes the turn, then Gold cannot profitably pass back. If he tries, then Silver can pass again, after which Gold cannot pass back again without instantly losing from having caused position X to occur for the third time. Therefore, a player's decision to pass the turn cannot be trivially undone by the opponent. Indeed, once a player moves a piece, the opponent can repeatedly pass and force the player to make another piece move, for as many turns as desired.

2.4.1 Big picture

▶ **Definition 10.** *The* simulation grid*, or simply* grid*, is the simulated $AGG_{1.1}$ graph located in the middle of the Arimaa′ board. The* simulation *is the ongoing process of players making moves that simulate placing markers on nodes of the $AGG_{1.1}$ graph.*

In Figure 4 we see the grid in the center and the rabbit prisons (see Section 2.4.3) near their respective goal ranks. The Gold player in Arimaa′ will simulate the actions of Gold in the $\mathrm{AGG}_{1.1}$ game. As detailed later, the Arimaa′ players simulate the marking of a node by moving a cat into the corresponding gadget on the board, after which it becomes captured or otherwise made irrelevant and simultaneously frees a previously frozen enemy cat. An $\mathrm{AGG}_{1.1}$ node marked by Gold (Silver) will therefore correspond to an Arimaa′ gadget with a gold (silver) cat entering it and a silver (gold) cat leaving. The latter cat will then travel to an adjacent node gadget. In case of the start node, a silver cat is leaving, so we can think of it as being already marked by Gold.

Surrounding the grid are *walls,* which we describe in the next section. Outside these walls are two rabbit prisons, whose construction and purpose we describe in the section after.

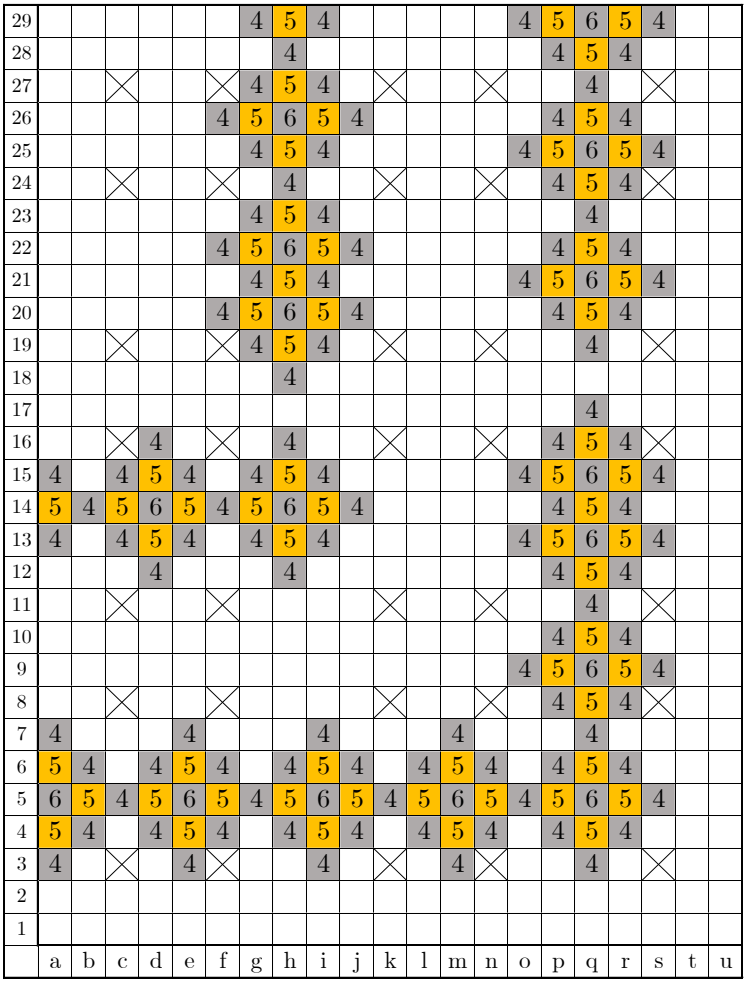


Figure 5 A part of a silver corridor containing a 90° turn.

2.4.2 Walls

▶ **Definition 11.** *A* free *piece is a piece that is allowed to move, because it is not frozen or blockaded by other pieces.*

▶ **Definition 12.** *A* stable *group of pieces is one containing no free piece.*

A wall, as the name suggests, is a special kind of continuous line of stable pieces that is designed to contain and constrain the motion of nearby free pieces. Specifically, walls constrain only *weak* pieces (recall that this means pieces of strength 3 or less). A wall is designed for just one color; its main function is to stop weak pieces of the appropriate color that are on one side of the wall from crossing over to the other side. As we explain below, no appropriately colored weak piece can get through them or break them down. Although a wall is designed for only one color, pieces of both colors are needed to construct it. The color of weak pieces that the wall constrains is always the opposite of the wall's outermost pieces (the 4s – see Figure 5). We say a wall is *gold* (*silver*) if its outer 4s are gold (silver). For example, the walls in Figure 5 are silver walls.

The main building block of a wall is a 6 with four enemy 5s and eight friendly 4s around it. The 6 is unable to push or pull because of the double layer of pieces around it, and the 5s and 4s are frozen, so the group of pieces as a whole is stable. These building blocks can be laid out in any desired direction to build walls of various shapes.

The parallel walls in Figure 5 jointly form a *corridor,* inside which a free piece may move. The inside of the corridor shown spans from row 8 to 11 and from column k to n. The 90° turn is included for demonstrative purposes; corridors can extend in a single direction for an arbitrary length. Straight corridors will correspond to edges in our reduction from $\mathrm{AGG}_{1.1}$.

Observe how a silver (gold) wall constrains the movement of weak gold (silver) pieces, since the weak piece gets frozen the moment it touches any outer 4. In the case of Figure 5, the walls depicted have gaps on squares h17 and q18, but since each gap is only one square wide and has 4s next to it, a weak gold piece cannot pass through it without getting frozen. However, any strong gold piece can move through these gaps. If its strength is 5 or greater, that piece can even destabilize the wall by pushing/pulling 4s. Furthermore, *any* free *silver* piece can also move through the gaps and can destabilize the wall by unfreezing a 4.

It will normally hold that only weak gold (silver) pieces get free near silver (gold) walls.[10] Since we always know which player may use a given edge in $\mathrm{AGG}_{1.1}$ (see Remark 6), we can carefully construct corridors of correct colors accordingly. All corridors will have gaps similar to the two in Figure 5, but they are impenetrable by the weak free pieces.

In addition to color, walls can also be categorized by their *parity*. We define the parity of a wall based on the color of pieces that lie on the diagonal of the upper-left/bottom-right traps (in the groups of four traps). If the color of the pieces on that diagonal is silver, we say the wall has *standard* parity (regardless of the wall's color). Otherwise, the parity is *nonstandard*. In Figure 5, the wall depicted has standard parity.

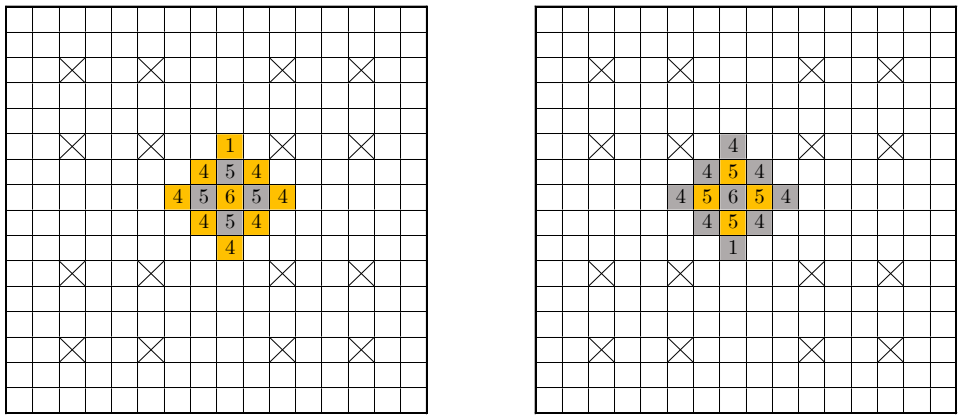
Gold walls are made in almost the same way as silver walls, but aside from reversing the piece colors, we also shift the wall one square to maintain standard parity. This is because reversing the piece coloring changes the wall's parity, so the shift is done to undo that change. We desire all walls to be built with the same parity so they can connect properly. More on connecting corridors can be found in Section 3.2.

2.4.3 Rabbit prisons

The rabbit prisons are similar to wall building blocks, but with a 1 in place of a 4. See Figure 6. Each player's rabbit prison is located near that player's goal rank. The prisons' sole purpose is to ensure that players comply with the $\mathrm{AGG}_{1.1}$ simulation in the Arimaa′ game, by giving the opponent of a non-complying player a way to punish them.

Specifically, a player who makes a move in the Arimaa′ game that breaks the simulation will let the other player get a loose 4 or 6. (Details are explained in Section 3.3.2.) Such pieces are strong enough to pass through gaps in the walls and walk outside the grid. When this happens, the loose 4 or 6 can reach the appropriate rabbit prison and unfreeze the rabbit there, letting it step onto the goal rank and win the game.

[10] The only exception occurs when a player fails to uphold the simulation of the $\mathrm{AGG}_{1.1}$ game within the Arimaa′ game. In Section 3.3.2, we see how any player who does this allows the opponent to have a free piece near a same-colored wall, leading to a forced win for that opponent. This dynamic serves to ensure the simulation is upheld until the end.



(a) (b)

Figure 6 A gold and a silver rabbit prison.

2.4.4 Viable winning conditions

As detailed in the game rules (Section 2.1.3), there are multiple ways to win at Arimaa. However, some winning conditions have been altered in Arimaa′. In the setup presented here, in which Arimaa′ is used to simulate an $\mathrm{AGG}_{1.1}$ game, the normal method of victory has a player move a piece into a node gadget for the second time (always a join, the only node type with in-degree 2 – see Figure 2), which leads, as we will later see, to an eventual threefold repetition for the opponent. As discussed above, there is also an alternative winning method possible against an opponent who does not comply with the $\mathrm{AGG}_{1.1}$ simulation.

The second Arimaa winning condition – capturing all opposing rabbits – can be disregarded in Arimaa′. This is because, apart from all the rabbits in the simulation, the one in the rabbit prison must also be captured. In order to do this, a player must get a free piece outside the grid, and as soon as a player achieves that, it is equally possible to win by freeing the player's own imprisoned rabbit and walking it to the goal rank.

The third Arimaa winning condition has been simply removed from Arimaa′, as noted in Definition 8.2. But note that in Arimaa′, if a player moves a piece and then *ends* (rather than begins) their turn with no movable pieces, then their opponent can force a win by passing twice and causing them to lose the game by the new repetition rule.

2.4.5 Reducing Arimaa′ to Arimaa

We now show that reducing $\mathrm{AGG}_{1.1}$ to Arimaa′ suffices to prove our main theorem.

▶ **Lemma 2.** *$n \times n$ Arimaa is PSPACE-hard if Arimaa′ is PSPACE-hard.*

Proof. For any Arimaa′ position with board state P', we show how to convert it to an $n \times n$ Arimaa board state P such that Gold wins in P if and only if Gold wins in P' (with the same player to move). This works in polynomial time. We begin by introducing the concept of a *cat cage*, seen in Figure 7. The cat cage depicted is empty, but in the actual construction a (silver or gold) cat is placed on one of the two empty squares in the middle. A cat cage is *gold* (respectively, *silver*) if a gold (respectively, silver) cat is put into one of the two central empty squares.

In P, the board state is identical to that of P' except that a gold and a silver cat cage are placed outside the grid. By the cages' design, a cat in one of the two middle squares of a cage is only able to move back and forth between those squares, and all other pieces in the

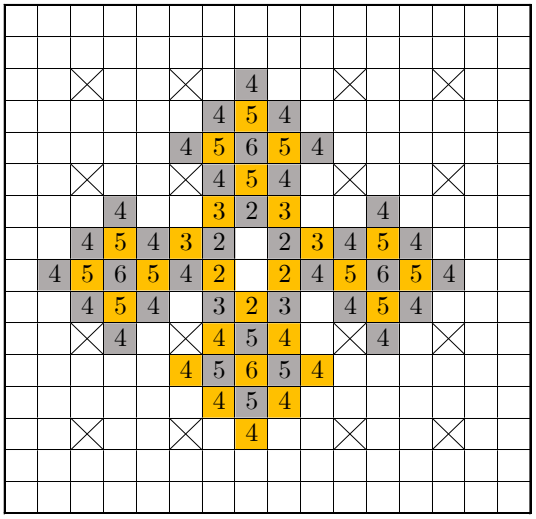


Figure 7 An empty cat cage.

cage remain immobile no matter the position of the cat. Further, note that there will never be interaction between the pieces of the cat cages and those of the grid, because any player who would somehow get a free piece outside the grid would just move it to the rabbit prison to free the rabbit and win. These considerations lead us to the following definition.

▶ **Definition 13.** *A* grid state *is an arrangement of pieces on the grid, without consideration of which player is moving next or the caged cats' positions. A* cat cage move *is a turn in which the player simply moves the caged cat to the other accessible square and ends the turn. Thus, a cat cage move leaves the grid state intact and changes only a caged cat's location.*

We now argue that the rule differences between $n \times n$ Arimaa and Arimaa′ described in conditions 8.2-8.3 do not change the outcome of the game. By assumption, the Arimaa players can play the same grid moves as the Arimaa′ players. Passing moves are not legal, but these can be simulated by cat cage moves. The cat cages also ensure that players will not lose the game by beginning a turn with no free pieces. Still, we must show that our remark about Arimaa′ players losing if they *end* a turn with no free pieces applies to the Arimaa game.

During the course of play, suppose the Arimaa board is in an arbitrary board state and a player (without loss of generality, Gold) has just ended a turn with no free non-cage pieces. We will show that Gold loses in this situation.

Assume without loss of generality that each cat is on the south square of its cage. Given a grid state C, let C ⇓⇓ denote the board state consisting of grid state C with both caged cats on the south square. The current position can then be written gC ⇓⇓, denoting that Gold has just caused board state C ⇓⇓. From position gC ⇓⇓, Silver can make a cat cage move (moving its cat to the north square), resulting in position sC ⇓⇑. As Gold has no more legal moves left in the grid, he is forced to make a cat cage move, resulting in position gC ⇑⇑. Because Gold has no other options, Silver can force the following sequence: gC ⇓⇓ (initial position), sC ⇓⇑, gC ⇑⇑, sC ⇑⇓, gC ⇓⇓ (second occurrence), sC ⇓⇑ (second occurrence), gC ⇑⇑ (second occurrence), sC ⇑⇓ (second occurrence). From here, Gold has to create position gC ⇓⇓, but this is illegal in Arimaa as it repeats the position a third time. Accordingly, Gold has no legal moves and loses by Arimaa rules.

We now see that the presence of cat cages and the use of the repetition rule compensates for the difference between Arimaa and Arimaa′ caused by the 8.2-8.3 rule changes. On the other hand, while passing can be simulated by with cat cage moves, Arimaa player can actually make cat cage moves back and forth for longer than players can pass back and forth in Arimaa′ (compare the move sequence above in this proof with Remark 9). However, just as in Arimaa′, players cannot trivially undo an opponent's decision to simulate passing the turn with cat cage moves; we have seen that if a player's opponent insists on only cat-cage movement, then the player must eventually progress the simulation with a move in the grid. ◀

▶ Remark 14. In the course of play, there may arise positions in which the grid contains arrangements of pieces that function similarly to a cat cage. That is, there may arise groups of pieces internally within the grid such that a cat imprisoned inside is mobile, but can only move through a fixed region of squares within. We call such a region a *synthetic cat cage*. This may occur after a player uses a turn switcher gadget (see Section 3.3.2).

We observe that synthetic cages have no effect on the outcome of the game, for either the Arimaa′ or Arimaa players discussed above. If an Arimaa player makes a synthetic cage move, the result is equivalent to making a cat cage move.[11] If an Arimaa′ player makes a synthetic cage move, the opponent can respond with a pass, forcing the first player to do something productive.

2.4.6 Reachability from the starting position

Before we proceed to the proof of the main theorem, we note that any simulated $\mathrm{AGG}_{1.1}$ position on an Arimaa and Arimaa′ board can be reached from the start of the game. This is easy to see, as pieces can move any direction (except rabbits, which cannot move backward). The board must be large enough that the home ranks can contain all pieces needed for the simulation. Then the players offer each other no resistance when moving pieces to form the simulated $\mathrm{AGG}_{1.1}$ position. Excess pieces can be captured in traps.

3 Proof

In this section we present the formal proof of Theorem 1:

▶ **Theorem 1.** *The Arimaa decision problem is PSPACE-hard.*

It will be clear that the reduction from $\mathrm{AGG}_{1.1}$ to Arimaa′, like the reduction from Arimaa′ to Arimaa, is polynomial-time computable.

3.1 Movement through the grid

The simulation of the $\mathrm{AGG}_{1.1}$ game in Arimaa′ works by always having only one player, called the *active player,* incentivized to move pieces, while the other player, called the *inactive player,* simply passes the turn repeatedly, until the active player completes a series of moves that simulate marking a node. This process involves a partial gadget called a *turn switcher* (see Section 3.3.2). After this, the roles switch as the second player becomes incentivized to move pieces while the first player repeatedly passes the turn, and so on. This continues until the simulated $\mathrm{AGG}_{1.1}$ game is complete.

[11] The reader can verify that if both players are making (synthetic or non-synthetic) cage moves in a sequence similar to the eight-turn sequence in the proof above, then switching mid-sequence to moving a cat from a different cage does not help.

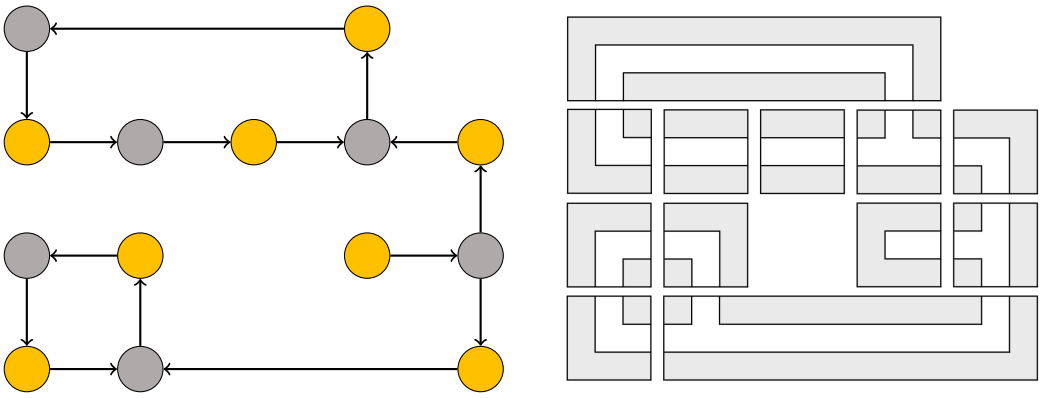


Figure 8 An $AGG_{1.1}$ graph (left) and sketched representation by Arimaa′ gadgets (right).

▶ **Definition 15.** *The key piece in the grid is the* loose cat*, a free cat that is the only piece on the board able to travel. This cat is capable of moving from one gadget to the next. For as long as the simulation is running, the active player will have a loose cat.*

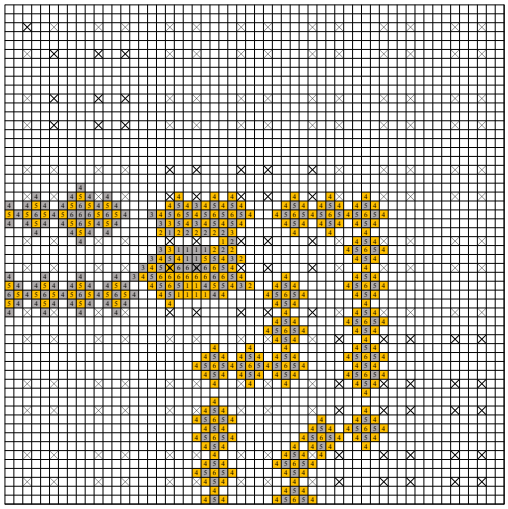
The simulation begins with Silver active (see Section 3.3.1). Players keep their roles until forced to switch in a turn switcher (see Section 3.3.2). We consider two cases. If the active player has a winning Geography strategy, the active player will have incentive to continue making moves that progress the $AGG_{1.1}$ simulation. If the active player does not have a winning strategy, the inactive player will keep passing turns and thereby force the active player to keep making moves in the simulation. In this case, while the active player may initially make irrelevant moves, the threat of threefold repetition will eventually force the loose cat to make progress in the simulation. This principle ensures that the losing player keeps moving through the corridors and turn switchers. So, in either case the simulation will eventually reach its natural conclusion.

3.2 Connecting gadgets

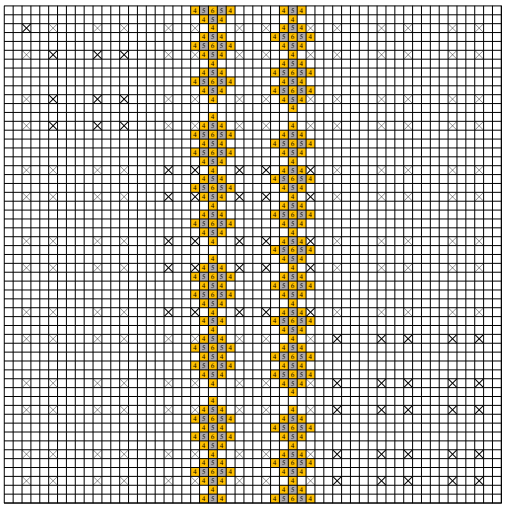
Before we present the details of the gadgets representing the different nodes of the $AGG_{1.1}$ game, we look at how they are stitched together to form the grid. In Figure 8 we see how the example $AGG_{1.1}$ graph from Figure 3 looks once transformed into square blocks. The example is made out of a starting node, one fork, two joins, two pipes, and eight bends. These blocks will become the gadgets in the reduction. For some examples, see Figure 9. The gadgets each use 49 concatenated standard Arimaa boards, arranged in a 7×7 square. The connecting corridors are all centered and have standard parity, ensuring that the gadgets connect correctly to each other. The detailed figures in the rest of this section cut off some of the corridor and empty space, leaving only the most critical parts of the gadgets or partial gadgets visible.

3.3 Gadgets

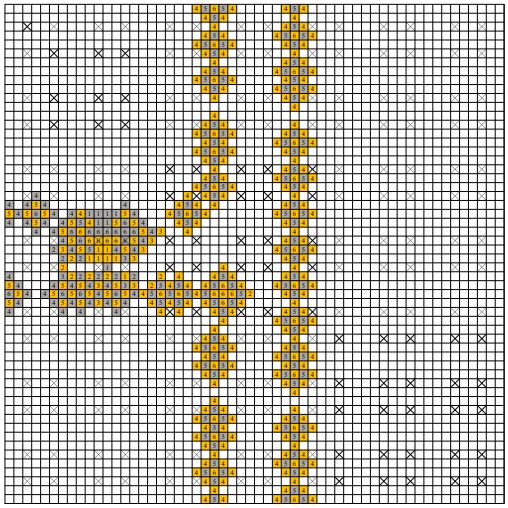
All the stable structures forming the simulation are built in standard parity, except for the turn switcher. Most gadgets in this section are depicted in just one color, but each can be transformed by reversing the piece colors, while preserving the parity by shifting the wall position one square as discussed in Section 2.4.2. The turn switcher poses a more difficult challenge, as we will see in Section 3.3.2.



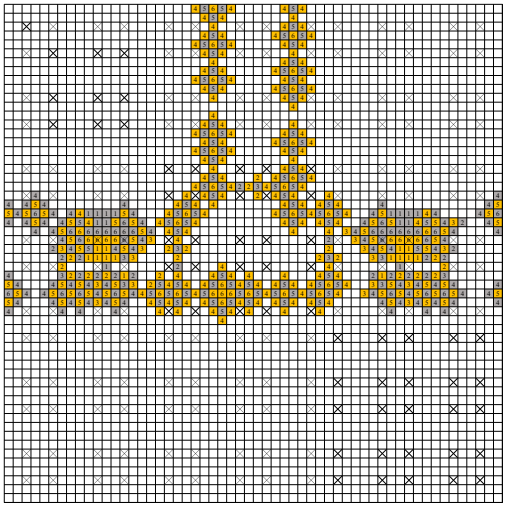
(a) A silver south-to-west bend gadget.



(b) A long vertical silver edge.



(c) A gold fork gadget.



(d) A gold join gadget.

Figure 9 Complete gadget examples.

3.3.1 Starting gadget

Recall that an $\mathrm{AGG}_{1.1}$ game starts with Gold marking the start node and Silver then marking the next node. The player exiting the gadget simulating the start node should therefore be Silver. This gadget representing the start node is nothing more than a corridor that is closed on one side, with the other side connecting to the gadget representing the second node (see Figure 10). In it is a loose silver cat to start the simulation. This is the only free piece at the outset; the other pieces on the board are either part of the grid or rabbit prisons, hence stable.

3.3.2 Turn switcher

This is the most important and complex partial gadget, responsible for switching the active and inactive player. In broad terms, it activates when a loose cat approaches and frees a friendly cat that then pushes an opposing rabbit down a narrow passage. At some point,

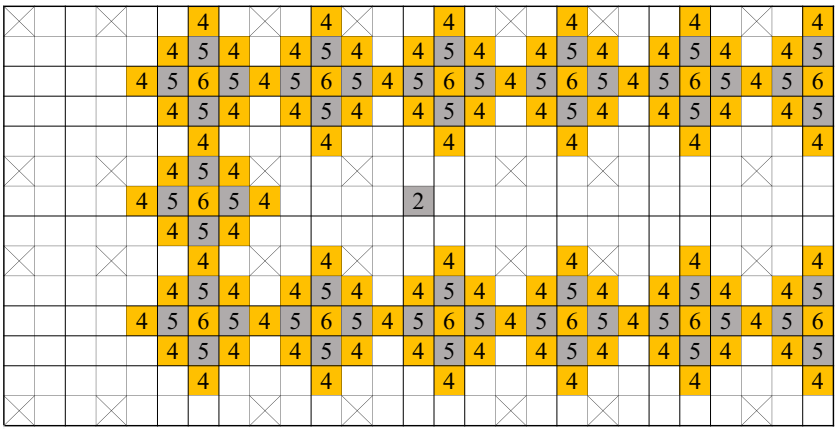


Figure 10 The critical part of a starting gadget.

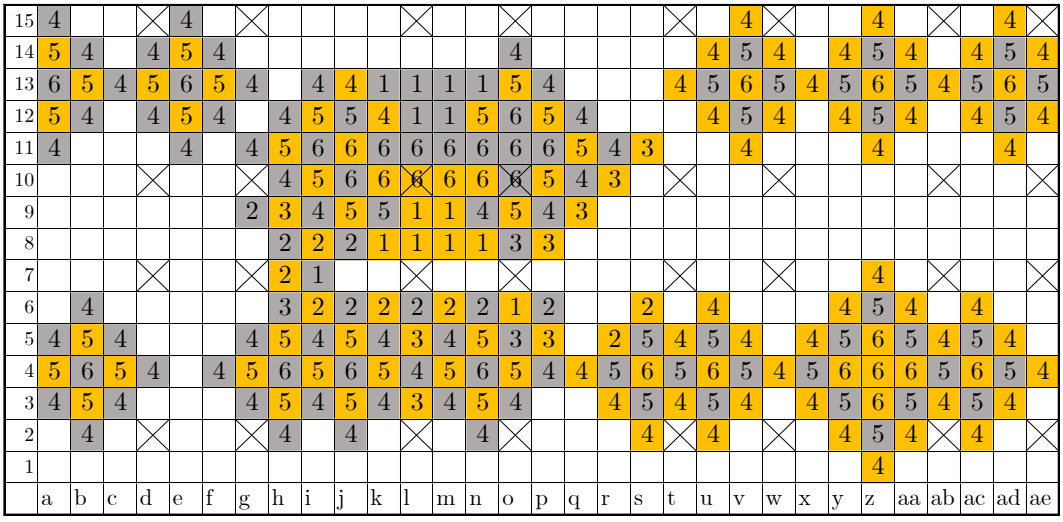


Figure 11 A turn switcher changing Gold as active player to Silver.

the inactive player will have to respond, as otherwise the second cat will be able to push all the way through the passage and end up on the other side of the turn switcher, where it will be able to free an elephant that can be used to win the game. Now follows a detailed explanation.

Figure 11 shows a turn switcher, which we see is completely stable before it gets activated. Avoiding repetition, a loose cat from Gold will eventually enter the switcher via the left corridor, where it will have no choice but to step on the g7 trap square and free the cat on h7. The h7 cat then has no choice but to push the silver rabbit on i7 east, causing the g7 cat to be captured. The remaining free cat is then forced to stay east of h7, because on h7 it would get frozen again, allowing Silver to win by repeatedly passing the turn. At some point, Gold will have to push the silver rabbit further right to j7 (case 1), k7 (case 2), l7 (case 3), m7 (case 4), n7 (case 5), o7 (case 6), and/or p7 (case 7). As Gold can push the rabbit up to twice per turn before Silver gets to move, Gold must choose which squares to allow the rabbit to be on when Silver starts her turn. For example, if Gold wants to avoid leaving the rabbit on m7, then Silver will necessarily have turns in which her rabbit begins on l7 and n7.

We consider all cases below. In each case, we will assume Gold keeps the loose cat adjacent to the silver rabbit between j7 and p7. If Gold instead decides to abandon the silver rabbit and bring the cat back west, nothing changes in how Silver should act.

Case 1 (j7)

The silver cat on j8 is now free and can push the k8 rabbit, freeing the elephant on j10 and allowing it to break down the wall and complicate the position. However, this series of events would never happen, because this requires both players to cooperate; Gold would have to choose to push the rabbit to j7 and end the turn, and Silver would then have to move the j8 cat. Whichever player has a winning strategy in the $\mathrm{AGG}_{1.1}$ simulation could decline to let this sequence of moves occur.

Case 2, 4 (k7, m7)

The silver rabbit is frozen, so Silver simply passes and soon Gold must push the rabbit again.

Case 3 (l7)

Similar to cases 2 and 4. The rabbit can only move one square east before it gets frozen again, but it will have no incentive to do so as Gold can reply by immediately starting to pass all turns, causing Silver to lose the game. Meanwhile, the silver cat on l6 is technically unfrozen now, but remains still unable to move because it has become blockaded.

Case 5 (n7)

Similar to case 1, with the n6 cat replacing the j8 cat.

The interesting cases are 6 and 7, as Silver will be forced to stop passing and make a move in the simulation. At the end of case 6, the roles of active and inactive player will have been switched. Case 7, we will see, will never occur.

Case 6 (o7)

In this position, we claim that Gold is now making a game-winning threat, forcing Silver to stop passing and instead move her now free rabbit. The threat is that if Silver passes now, Gold can double-push the rabbit east to q7 in his next turn, after which Silver is unable to stop Gold from going to the elephant prison and winning the game. We now explain how Gold would accomplish this, after which we describe Silver's answer to the threat.

The elephant prison is centered around the square z4. The purpose of the elephant prison is to give the active player a way to free an elephant without freeing the opponent's pieces.[12] In the event that Silver has not stopped the loose cat of Gold from walking all the way to p7 and pushing the silver rabbit all the way to q7, Silver now has no free pieces in this turn switcher. This allows Gold's cat to keep the rabbit frozen and push it into a trap at his leisure, then move toward the gold 4 on z7. Unfreezing the 4 activates the elephant prison. The 4 can move aside to give room to the elephant on z5 to push out the 5 on z6 (keeping it always frozen). This 5 may be captured with any of the available traps, leaving the elephant completely free. The silver 5s on y5 and aa5 stay frozen by the gold elephants below them.

[12] As it turns out, this construction is more elaborate than strictly needed. It would suffice to replace the elephant prison with normal wall, letting one of the wall's outer 4s play the elephant's role, as 4s are also large enough to escape the grid and win. Still, we aesthetically favor using the strongest piece.

After the elephant is freed, it can move through the gaps in the walls and escape from the grid to the large empty space outside. Once there, it will go to its friendly rabbit prison to free the rabbit there and win the game.

Returning to our initial case 6 position, Silver can and must prevent all this by moving her rabbit from o7 to p7, which unfreezes the cat on p6 so that it can push the gold rabbit on o6 to the o7 trap.[13] The purpose of Silver pushing the gold rabbit from o6 onto o7 is to restrict the mobility of Gold's n7 cat. If Silver doesn't do this, Gold's n7 cat would have the freedom to travel east and escape the turn switcher, which could benefit only Gold. So, having now used three out of four movement steps this turn, Silver can either end the turn now or move her cat back to p6 and then end the turn. Assume for simplicity that she chooses to move now to p6, since we will see that the cat must move to p6 soon anyway.

After this turn by Silver, it is Gold to move. In this position, we claim Gold now has no choice but to pass his turns. Observe that the rabbit now on the o7 trap square is immobile, as it is blocked laterally and to the north, and gold rabbits can never move south. Further observe that if Gold moves his n7 cat, this rabbit will be captured by the trap, and Gold would only end up making irrelevant moves with his cat between i7 and n7 (moving all the way to h7 would just get it frozen). Meanwhile, Silver would simply be able to pass the turn repeatedly and make Gold lose the game by repetition.[14] So, with Gold forced to keep his cat and rabbit still, we see that he must pass, at which time Silver becomes the active player and the silver cat now located on p6 becomes the loose cat. It is now free to go into the rest of the gadget this turn switcher is a part of and continue the simulation. The gold cat, meanwhile, stays permanently imprisoned between i7 and n7 and will never again be relevant in the game (see Remark 14).

As an aside, note that Silver cannot leave the switcher with both her newly loose cat and her rabbit, as each cannot free the other without freezing itself. (The useless rabbit is anyway confined to only six squares – see fn. 13.) So there is always only one loose piece.

Case 7 (p7)

Gold will not allow this to occur. If he does, Silver wins as follows. First, Silver captures the cat on the o7 trap by pulling the adjacent o6 rabbit with her p6 cat. Then she freezes that rabbit by pushing it back to o6. Gold has no choice now but to pass. The silver cat now pushes the rabbit to the o7 trap, capturing it, and travels backwards through the turn switcher to the west side. Careful to avoid the g7 trap, it frees the frozen 3, which frees a 4, which then escapes the grid and wins the game, similarly to gold's threat from case 6.

▶ Remark 16. In the construction of turn switchers, there are some subtleties in changing the switcher's color and/or direction, since almost every piece in it has its own function and since it is precisely built around and onto the trap squares. It also has an internal parity

[13] It would be a losing strategy for Silver to move her rabbit further east than p7, as that would allow Gold's cat to catch up and freeze the rabbit, move it into a trap and win via the elephant prison. In any case, silver rabbits cannot move north, so the s6 cat will keep it forever confined to squares q7, r7, s7, q6, r6, and q5.

[14] Note that even if this were an Arimaa game rather than Arimaa′, Gold would still lose the game. It might seem at first glance that Gold's initial westward movement of his cat is itself a sort of synthetic cat cage move, making Gold the initiator (and therefore winner) of an eight-move repetition sequence (as he shuffles his cat between m7 and n7), but this is not the case. Gold's first movement to m7 causes the o7 rabbit to be captured, creating a necessarily new Arimaa position and allowing *Silver* to be the first to mark time in a cat cage.

change[15] where there is a double column of adjacent rabbits and elephants of the same color. The upper left and the entire bottom side of the turn switcher in Figure 11 are of standard parity, so they need no adjustment to connect to walls of standard parity. The upper right corner, however, is not, so an extra gap of space is left between that part of the turn switcher that is not of standard parity and the wall that is. The gap is not wide enough for enemy cats to go through, as they would still be frozen by the pieces making up the wall.

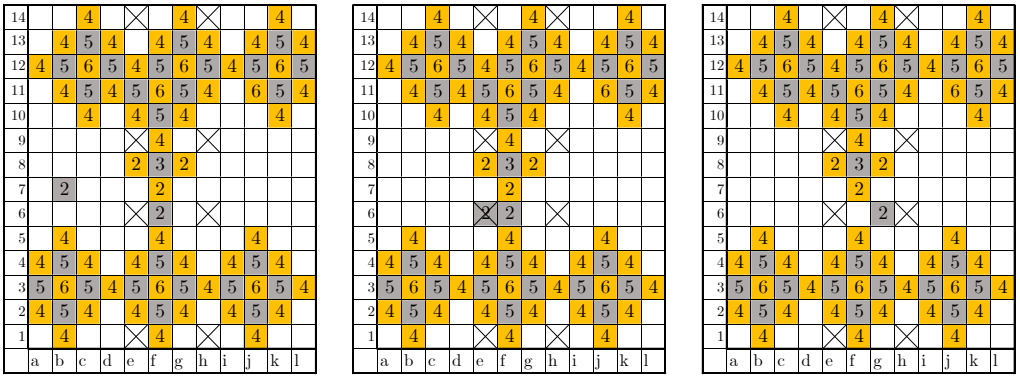
Our reduction requires four types of turn switchers: rightward gold-to-silver (as in Figure 11), leftward gold-to-silver (as in Figure 13, right turn switcher), rightward silver-to-gold (not depicted), and leftward silver-to-gold (as in Figure 9a). All are constructed similarly, but with the following adjustments. Changing the turn switcher's direction requires mirroring it over its vertical axis, resulting in a parity change. Changing its color requires switching its piece colors and mirroring it over the horizontal axis (since gold and silver rabbits are mirrored in their vertical movements). In all cases, a wall gap similar to the one in s12-t12 in Figure 11 is included where needed.

3.3.3 Pipes and Bends

A pipe is easily constructed by a straight corridor with a turn switcher. A bend is similar, but with a 90° corner. Bends can be made in any direction, with the turn switcher in whichever of the two corridors connected by the corner is horizontal. See Figure 9a for an example.

3.3.4 Joins and Sleds

A join can be constructed from a two facing turn-switchers that feed into a common central area that leads into a vertical corridor. Within the central area are two occurrences of a small partial gadget that we call a *sled*. We now describe the workings of sleds, after which we discuss the join itself and how it uses sleds.



(a) Before. **(b)** During. **(c)** After.

Figure 12 A sled between two walls in a corridor in different stages of use.

Sleds are built between two corridor walls. They are designed to let just a single loose cat pass a single time. See Figure 12 for a sled in action. In 12a we can see a loose cat (b7) traveling east inside a corridor, encountering a sled. The sled functions like a dead end for

[15] The reason for this internal change of parity is that the turn switcher needs to be in standard parity both above and below the entrance – i.e., both h6 and h8 have silver pieces in Figure 11 – in order to keep the h7 cat initially frozen. By contrast, o6 and o8 have pieces of opposite color, as do p6 and p8, in order for the turn switcher mechanism to work.

this cat, and eventually it will have no choice but to move onto the trap square e6, as seen in Figure 12b. Now the cat on f6 is unfrozen and has no choice but to move east, becoming the new loose cat as the e6 cat gets captured (see Figure 12c). This new loose cat is free to go north (g7) and east (h7), then continue movement through this corridor. Observe that the sled can only be used in its intended direction and only one time.[16]

Recall that the main feature of an $AGG_{1.1}$ join is that it is the only type of node that may be marked twice, and subsequently the player who marked it wins the game. The Arimaa′ join gadget is designed to work similarly. In Figure 13 we see a join ready to be marked by Gold.[17] So Gold enters this join and Silver leaves it. Without loss of generality, assume Gold enters from the west. Silver emerges from the turn switcher and has no choice but to use the sled on the west side. With the loose silver cat now in the middle of the join, the only place it can go is the vertical corridor exiting the gadget toward the turn switcher of the next node gadget (north of this join gadget), since the east sled cannot be used from this side. The next time this join gadget is used, Gold will activate the turn switcher on the east to enter. Silver again emerges and has no choice now but to use the eastern sled and again go to the vertical corridor leading to the next gadget. But since the next gadget's turn switcher cannot be used twice (the entrance works like a sled), Silver is now unable to get past this point. Stuck in a dead end, she will eventually run out of new moves and lose the game by repetition.

3.3.5 Forks

Forks gadgets represent nodes with indegree 1 and outdegree 2. The fork shown in Figure 14 is a gold fork. The silver loose cat that will leave the turn switcher may choose to take either the northern or southern outgoing corridor. Backtracking at a later stage in the simulation to also use the other direction of the fork is impossible, as each node the fork feeds into has a turn switcher, which a cat cannot go through in reverse.

3.4 Proof conclusion

We see that every gadget works as required. This completes the proof of Theorem 1. ◀

[16] The metaphor of a sled is imperfect, since the cat entering the contraption is not the same physical cat as the one coming out. But it is our attempt to convey a form of uni-directional travel that (for a cat, at least) can be used only once.

[17] This depiction is modified from the one in Figure 9c, in order to show its functioning more clearly. The official join from 9c is more compressed and has only one elephant prison that is shared by both turn switchers.

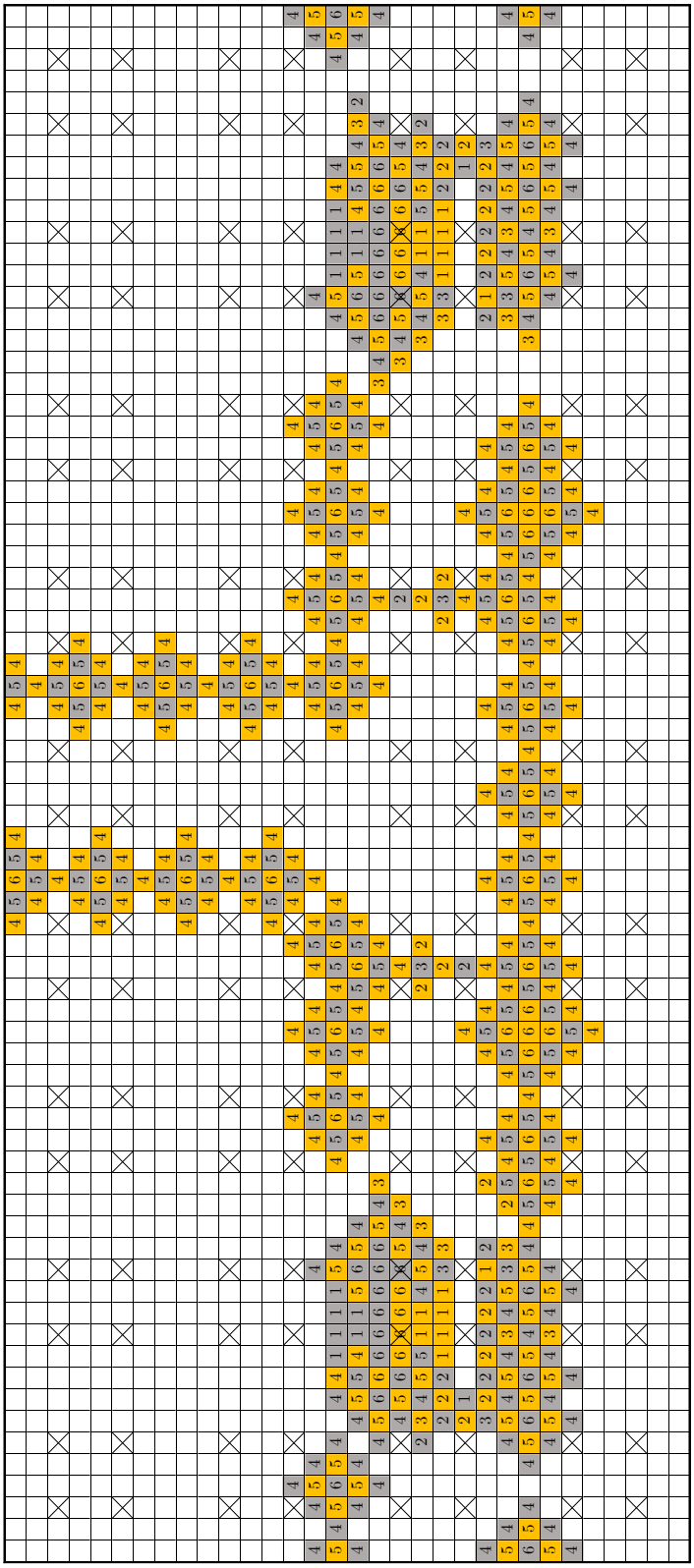


Figure 13 The critical part of a join gadget, expanded.

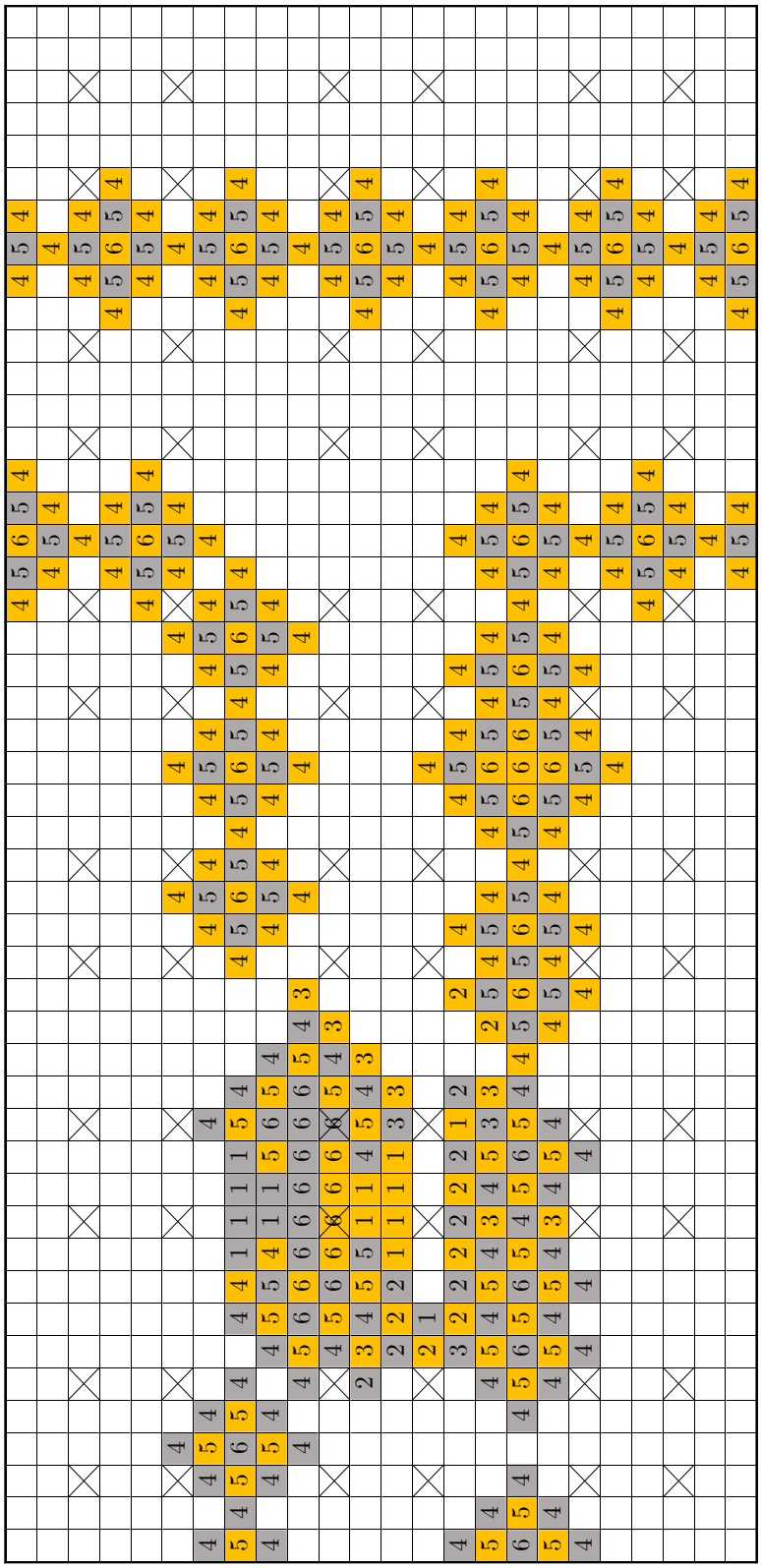


Figure 14 The critical part of a fork gadget.

4 Discussion and Conclusion

4.1 Overview

The inception of the game Arimaa has inspired many researchers to explore the practical complexity of the game when developing their bots. The result of this paper on the theoretical complexity of the game proves that the problem of solving Arimaa is intractable (under the assumption P $\subsetneq$ PSPACE). It also gives us a way to compare the game to other games in terms of difficulty, and of course adds one more problem to the library of known PSPACE-hard problems.

As remarked earlier, one notable feature of this proof is that it shows PSPACE-hardness of a game whose pieces only move short distances in a single turn. While our reduction strategy is largely based on that of [16] for chess, chess does not present this same challenge because some pieces can move arbitrarily many squares per turn. In Arimaa, the strictly local movement of pieces calls for the idea of cat cages and Arimaa′ in our reduction.

4.2 Future research

This paper shows that Arimaa is PSPACE-hard because $\mathrm{AGG}_{1.1}$ reduces to Arimaa′, which in turn reduces to Arimaa. The reduction presented in the proof works with traps placed on the corners of a 4×4 square, with four empty spaces between adjacent groups of traps and two empty spaces separating traps within a group. However, another way to generalize the standard 8×8 Arimaa board could be to place traps with two empty spaces between them in all directions. For this alternative version of $n \times n$ Arimaa, the reduction in this paper would no longer work, as it is impossible to create any kind of stable building block like the one we used here for the walls. This does not rule out all other possible reductions, but at the moment we do not know how they would look. With minor adjustments, our reduction can work with a constant number of three or more empty squares between traps, although that might be less in line with the spirit of the original game. In future research it would be interesting to see a reduction for a version containing traps spaced with two squares between them in every direction (rather than alternating gaps of two and four squares), if possible.

Further, as discussed in this paper's introduction, we expect that it is possible to prove Arimaa is EXPTIME-hard, perhaps using a reduction similar to that of [8]. Raising the lower bound of Arimaa's complexity to match the upper bound, this would make Arimaa EXPTIME-complete, just like chess (without a generalized 50-move draw rule) [8], checkers [13], and Go (with Japanese ko rules) [12]. However, we also consider it possible, though less likely, that Arimaa is in PSPACE and is therefore PSPACE-complete.

References

1 Joshua Ani, Jeffrey Bosboom, Erik D. Demaine, Yevhenii Diomidov, Dylan H. Hendrickson, and Jayson Lynch. Walking through doors is hard, even without staircases: Proving PSPACE-hardness via planar assemblies of door gadgets. In Martin Farach-Colton, Giuseppe Prencipe, and Ryuhei Uehara, editors, *10th International Conference on Fun with Algorithms, FUN 2021, May 30 to June 1, 2021, Favignana Island, Sicily, Italy*, volume 157 of *LIPIcs*, pages 3:1–3:23. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPICS.FUN.2021.3.

2 Arimaa.com. The 2015 Arimaa Challenge. http://arimaa.com/arimaa/challenge/2015/, 2015. [Online; accessed 20-02-2024].

3 Davide Bilò, Luciano Gualà, Stefano Leucci, Guido Proietti, and Mirko Rossi. On the PSPACE-completeness of Peg Duotaire and other Peg-Jumping Games. In Hiro Ito, Stefano Leonardi, Linda Pagli, and Giuseppe Prencipe, editors, *9th International Conference on Fun with Algorithms (FUN 2018)*, volume 100 of *Leibniz International Proceedings in Informatics*

(LIPIcs), pages 8:1–8:15, Dagstuhl, Germany, 2018. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.FUN.2018.8.

4 Edouard Bonnet, Florian Jamain, and Abdallah Saffidine. Havannah and TwixT are PSPACE-complete. In H. Jaap van den Herik, Hiroyuki Iida, and Aske Plaat, editors, *Computers and Games - 8th International Conference, CG 2013, Yokohama, Japan, August 13-15, 2013, Revised Selected Papers*, volume 8427 of *Lecture Notes in Computer Science*, pages 175–186. Springer, 2013. doi:10.1007/978-3-319-09165-5_15.

5 Christ-Jan Cox. Analysis and implementation of the game Arimaa. *MSc diss., Universiteit Maastricht, The Netherlands*, 2006.

6 David Fotland. Building a world-champion Arimaa program. In *International Conference on Computers and Games*, pages 175–186. Springer, 2004.

7 David Fotland. Computer Arimaa: The beginning. *International Computer Games Association Journal*, 38(1):12–18, 2015. doi:10.3233/icg-2015-38103.

8 Aviezri S. Fraenkel and David Lichtenstein. Computing a perfect strategy for $n \times n$ chess requires time exponential in n. *J. Comb. Theory, Ser. A*, 31(2):199–214, 1981. doi:10.1016/0097-3165(81)90016-9.

9 Brian Haskin. A look at the Arimaa branching factor. http://arimaa.janzert.com/bf_study, 2006. [Online; accessed 20-02-2024].

10 David Lichtenstein and Michael Sipser. Go is polynomial-space hard. *Journal of the ACM*, 27(2):393–401, apr 1980. doi:10.1145/322186.322201.

11 Stefan Reisch. Hex ist PSPACE-vollständig. *Acta Informatica*, 15:167–191, 1981. doi:10.1007/BF00288964.

12 J. M. Robson. The complexity of Go. In R. E. A. Mason, editor, *Information Processing 83, Proceedings of the IFIP 9th World Computer Congress, Paris, France, September 19-23, 1983*, pages 413–417. North-Holland/IFIP, 1983.

13 J. M. Robson. N by N checkers is Exptime complete. *SIAM Journal on Computing*, 13(2):252–267, 1984. doi:10.1137/0213018.

14 Thomas J. Schaefer. Complexity of decision problems based on finite two-person perfect-information games. In Ashok K. Chandra, Detlef Wotschke, Emily P. Friedman, and Michael A. Harrison, editors, *Proceedings of the 8th Annual ACM Symposium on Theory of Computing, May 3-5, 1976, Hershey, Pennsylvania, USA*, pages 41–49. ACM, 1976. doi:10.1145/800113.803629.

15 Atze Schipper. $n \times n$ Arimaa is PSPACE-hard. Bachelor's thesis, Utrecht University, 2021.

16 James A. Storer. On the complexity of chess. *Journal of Computer and System Sciences*, 27(1):77–100, 1983. doi:10.1016/0022-0000(83)90030-2.

17 Omar Syed. The Arimaa Challenge: From inception to completion. *International Computer Games Association Journal*, 38(1):3–11, 2015. doi:10.3233/ICG-2015-38102.

18 Omar Syed and Aamir Syed. Arimaa - A new game designed to be difficult for computers. *International Computer Games Association Journal*, 26(2):138–139, 2003. doi:10.3233/ICG-2003-26213.

19 Gerhard Trippen. Plans, patterns, and move categories guiding a highly selective search. In H. Jaap van den Herik and Pieter Spronck, editors, *Advances in Computer Games, 12th International Conference, ACG 2009, Pamplona, Spain, May 11-13, 2009. Revised Papers*, volume 6048 of *Lecture Notes in Computer Science*, pages 111–122. Springer, 2009. doi:10.1007/978-3-642-12993-3_11.

20 Jaap van den Herik. Table of contents / editorial. *International Computer Games Association Journal*, 38(1):1–2, March 2015. doi:10.3233/icg-2015-38101.

21 Wikibooks. Arimaa/Playing the game. https://en.wikibooks.org/wiki/Arimaa/Playing_The_Game, 2019. [Online; accessed 20-02-2024].

22 David J. Wu. Move ranking and evaluation in the game of Arimaa. Bachelor's thesis, Harvard College, 2011. URL: https://api.semanticscholar.org/CorpusID:504661.