Bachelor's thesis

7.5 ECTS

Balanced asymmetry in general game playing

November 11, 2020

Thomas Dingemanse

t.dingemanse@students.uu.nl

3985628

Supervisor

Jan Broersen

Second reader

Michael De

Kunstmatige Intelligentie

Utrecht University

Universiteit Utrecht

Contents

1 Introduction

Since the early days of artificial intelligence, strategy games like chess have been used as a benchmark for the evaluation of intelligent systems (Genesereth et al., 2005; Silver et al., 2018), functioning as a proxy for more complex real-world environments. In 1992b, Barney Pell laid the groundwork for general game playing (GGP), a field focused on systems that can play a variety of previously unseen games well, based only on a formal description of the rules that is provided at the start of the game. These systems, called general game players, cannot utilize the training or self-play that are essential for the performance of machine learning algorithms for game playing such as AlphaZero,[1] because they only have a few minutes at most between receiving the rules and starting the game. There are many different types of general game players, see section 2.2.

1.1 Symmetry and balance

Before we delve deeper into the subject of general game playing, it is necessary to clearly define and differentiate between two related concepts: *symmetry* and *balance*. Symmetry means the game is played in the same way for each player,[2] while balance means all players have the same probability of winning. Symmetric games are always balanced, but balanced games are not necessarily symmetric. Creating balanced asymmetric games can be very difficult, but much effort is put into it nonetheless, think of multiplayer video games where different player characters have their own unique strengths and weaknesses.

Pell initially described the goal of general game playing as seeing "programs which can analyse and play *any* games that humans could play" (1992a, p. 2). He also decided it was more productive to focus on a smaller subclass first: "symmetric, chess-like games". Since the introduction of Game Description Language II which supported incomplete information games, some work has been done on information asymmetry (Schiffel & Thielscher, 2014), but not on other kinds of asymmetry or on balance.

If general game players are to be relevant to the field of artificial intelligence, they should be able to play the kinds of strategic games that humans find interesting and

[1] Section 2.3 provides additional information about AlphaZero and MuZero and compares them to general game playing.

[2] Note that the meaning of symmetry in this thesis is different from its meaning in a number of other general game playing papers, where it is used as a noun to refer to a structural aspect of a game representation as a finite state machine that can be exploited to reduce the search space for general game players (Schiffel, 2010).

challenging. This includes asymmetric games. To evaluate whether general game players are truly general, they should play asymmetric games equally well in any of the player roles, provided that the game is balanced. Therefore, finding whether balanced games cause differences in performance between asymmetric player roles could provide a new and valuable insight into the differences and similarities between human game playing and current computational approaches like GGP. Additionally, these findings could help to improve general game players that specifically perform badly at asymmetric games.

1.2 Current research

One of the four definitions of AI described by Russell and Norvig is *thinking rationally*. Rational agents should be "acting so as to achieve one's goals, given one's beliefs" (Russell & Norvig, 2009, p. 7). This involves domain-independent problem solving, a problem that GGP in general and the current research in particular contribute to, as stronger general game players are also better general problem solvers in other contexts.

The question I want to answer in the current research is "How does asymmetry in balanced games affect general game players?" The Viking board game hnefatafl is well suited to answer this question: it is asymmetric in terms of starting positions, number of game pieces, and goals, but its rules have evolved in such a way that the probability of winning is roughly equal for both players, so the game is fun to play for both the defender and the attacker. This differentiates the game from other asymmetric games that have been formalised for GGP, which are asymmetric, but also unbalanced.

To answer the research question, I will take an approach based on the one proposed by Pell (1992b): a tournament between general game players. The performance of the players will be evaluated in two ways. First, their performance against an opponent playing random moves: *intelligence*. Second, the difference in performance between player roles in games against themselves: *generality*. An intelligent and truly general game player should not manifest a bias towards either player role in balanced games, but it should win most games against an opponent playing random moves. The null hypothesis is that all players perform equally well for both roles, and they are evenly matched against the random player: the players are roughly as intelligent and as general as the random player. The alternative hypothesis is that some players are better at one role than the other, or that some players can consistently win the majority of matches against the random player, or both. Since there is no prior research into this topic, it is

not possible to say in advance which players will be good at playing which role.

To investigate the research question, this thesis will adhere to the following structure. First, section 2 (General game playing) contains a more detailed overview of the field and the various general game players, and compares general game playing to autonomous planning, AlphaZero, and MuZero. Next, in section 3 (Hnefatafl), I will describe the history, rules and complexity of hnefatafl, along with some practical considerations for its GGP implementation. The methods are outlined in section 4: formalization of the game rules, development of the agents, configuration of the tournament and how the agents will be evaluated. Then section 5 (Results) will report the outcome of the tournament. Furthermore, section 6 (Discussion) will evaluate the hypothesis based on the results from the previous section and discuss if the methods were appropriate and sufficient for the research aims. Finally, I will discuss how these results are relevant to the broader field of AI and provide some pointers for further research in section 7 (Conclusion).

2 General game playing

For simple games with relatively few states, classical search is useful, but it quickly becomes impossible as the state space increases. For more complex games like chess, partial search on the state space such as planning ahead a limited number of turns and choosing the best outcome, is generally a good strategy, but requires accurate heuristics to evaluate outcomes. However, hand-crafting evaluation functions of states for every game in advance requires human supervision, which is not always an option. One solution is to generate an evaluation function automatically based on the rules of the game. Another is to exploit the structure of the state space using constraint satisfaction, logical reasoning, or a tree search algorithm. All of these solutions fall into the domain of GGP, a general problem-solving approach to game playing. Today, much of the work in GGP revolves around the annual International General Game Playing Competition (IGGPC) at AAAI, and other GGP competitions. GGP has two main aspects:

1. Logically formalizing the rules of different kinds of games. Game Description Language (GDL) is used to specify a logical formalization for any finite turn-based game in terms of its basic elements (such as cells on a board), legal actions, state transitions, players, goals, and terminal states (Thielscher, 2011). See section 2.1 below.

2. Developing programs that can perform well across a wide range of games, based on such formalizations. These programs are called general game players, discussed in section 2.2.

In contrast to machine learning solutions, the aim of GGP is not just to get the most accurate results for a certain kind of problem, but to pinpoint the general knowledge or insight that allows humans to solve many different kinds of previously unseen puzzles, games and problems.[3] As a result, GGP may offer more insight into the inner workings of successful problem-solving strategies than the usually opaque nature of trained machine learning models.

2.1 Game descriptions

In GGP, it is common to model games as finite state automata (Schiffel, 2010). The structure of such a model depends on the rules of the game, which are described in GDL. GDL has two functionally equivalent variants: prefix GDL and infix GDL. Prefix GDL has a syntax comparable to Datalog,[4] whereas infix GDL is syntactically similar to Prolog.[5] Both variants are strictly declarative, like Datalog but unlike Prolog. GDL is based on first-order logic (FOL) with the closed-world assumption, also called full information, applying negation as failure. This means that logical reasoners can be employed for inference on game descriptions (Björnsson & Schiffel, 2013).

Currently, GDL has three versions: GDL-I, GDL-II, and GDL-III. GDL-I can describe any turn-based, finite, deterministic, perfect information game (Genesereth et al., 2005). GDL-II also supports non-deterministic games and imperfect information games (Thielscher, 2011). The most recent extension of the language, GDL-III, added support for epistemic games (Thielscher, 2017). In this thesis all three variants are simply called GDL, since the additional functionality added in GDL-II and GDL-III is not relevant for our purpose.

[3] Humans display the capacity to solve many different kinds of novel problems, but there are different views on what it is that underpins this ability. This question is best addressed from the perspective of cognitive science and philosophy of mind.

[4] Datalog is a strictly declarative logic programming language mostly used to query databases and designed to be read by computers rather than humans.

[5] Prolog is a logic programming language. Infix GDL is not based on Prolog itself, but on Epilog, a language for epistemic logical reasoning that is in turn based on Prolog.

2.2 General game players

General game players are defined by Genesereth and Björnsson (2013, p. 107) as "systems able to play strategy games based solely on formal game descriptions supplied at run time. In other words, they don't know the rules until the game starts." The somewhat vague term *strategy games* merits some additional explanation. According to Genesereth and Thielscher (2014, p. 1), "general game players should be able to play simple games (like Tic Tac Toe) and complex games (like Chess), games in static or dynamic worlds, games with complete and partial information, games with varying numbers of players, with simultaneous or alternating play, with or without communication among the players, and so forth."

There is an inherent uncertainty in GGP: general game players have no knowledge of the future actions of other players, and only rarely a guaranteed path to a goal state can be found. This means that general game players need to account for different possible outcomes of their actions. Additionally, time limits and state space size make finding a guaranteed path to a goal state impossible from most states.

A variety of general game players has been developed implementing different approaches, including logic programming (Schiffel & Thielscher, 2007), generating heuristic evaluation functions (Clune, 2007), constraint satisfaction (Gent et al., 2008; Koriche et al., 2016), and Monte Carlo tree search (MCTS) (Björnsson & Finnsson, 2009; Méhat & Cazenave, 2010; Möller et al., 2011). The latter, introduced in 2008 by Finnsson and Björnsson, and its variant Upper Confidence bound applied to Trees (UCT), have become ubiquitous in GGP because of their unparalleled performance (Genesereth & Björnsson, 2013). However, this may change again as last February a deep learning-based approach outperformed UCT (Goldwaser & Thielscher, 2020).

2.3 Comparable approaches to problem solving

GGP is not the only approach to intelligent problem solving and game playing. Two other approaches will be discussed below.

First, autonomous planning. In a short 2013 IGGPC review paper, Genesereth and Björnsson compare GGP to autonomous planning. They note similarities, such as domain-independence and the goal description of reaching one of the specified terminal states, but also some differences (table 1). In GGP there are opponents, which complicates decision-making, and there is an execution environment, allowing agents to switch be-

	Autonomous planning	GGP
Domain	domain-independent	domain-independent
Goal	specified state(s)	specified state(s)
Language	PDDL, STRIPS, and others	GDL
Opponents	no	yes
Execution environment	no	yes
Time constraints	no	yes

Table 1: *Similarities and differences between autonomous planning and general game playing. Similarities are shaded, differences are unshaded.*

tween planning ahead and acting in the current situation. Another difference is that autonomous planning is usually not subject to strict time constraints, whereas in GGP, agents tend to have only a few seconds or sometimes minutes to decide what their next move will be. Note that this is still a very long time compared to the few milliseconds commonly used in reactive planning: in GGP, agents have more time to analyse possible moves and plan multiple moves ahead, focusing more on deliberation than immediate action.

Second, the current state of the art machine learning algorithm for game playing MuZero and its predecessor AlphaZero, both developed by DeepMind, are in some ways comparable to GGP as well (table 2). For instance, all three methods apply algorithms like MCTS to play various complex strategy games. AlphaZero and MuZero both use MCTS to choose actions during self-play, applying respectively model-free and model-

	AlphaZero	MuZero	GGP
Knowledge of rules	yes	no	yes
Game playing	MCTS	MCTS	typically MCTS
Training time	days	days	minutes
Reinforcement Learning	model-free	model-based	no

Table 2: *Similarities and differences between AlphaZero, MuZero, and general game playing. Similarities are shaded, differences are unshaded.*

based reinforcement learning. The self-play data is then used to train a neural network. While most current GGP approaches also use MCTS, they don't use reinforcement learning or neural networks, as their aim is to acquire an understanding of the rules through inference, as opposed to learning the best moves from experience, i.e. self-play. Moreover, AlphaZero and MuZero require days of training, while GGP agents have to act in a matter of seconds after being provided the rules of a game, due to the limited time available before the game starts and between turns. AlphaZero and general game players have access to the rules of a game in advance. In contrast, MuZero must learn the rules based on gameplay data from its self-play phase. In this regard, MuZero (but not AlphaZero) is similar to Inductive General Game Playing (IGGP), a GGP variant where players must infer the rules from gameplay data (Cropper et al., 2019).

In conclusion, all three approaches have their own aims, strengths and weaknesses, and therefore their own place in the field of artificial intelligence. The place of GGP is real-time game playing without prior knowledge of the rules or training, with opponents and strict time constraints. While this is much more difficult than other approaches, this makes GGP uniquely suitable to investigate intelligent domain-independent deliberation as opposed to applying complex learned patterns or static planning.

3 Hnefatafl

Strategic board games have been a popular pastime for millennia (Sebbane, 2001, p. 213). One such game is hnefatafl,[6] a game in the tafl family that was played by Vikings in Europe before chess took over in popularity (Ashton, 2010, p. 1). The exact way hnefatafl was historically played is an ongoing debate. The best available resources on this topic are a number of translations of famous taxonomer Carl Linnaeus' travel journal from 1732, written in Latin and Swedish. It describes a similar tafl game: (Saami) tablut. Due to the terminology used in this manuscript, the defending pieces are sometimes called the Swedes, and the attacking pieces Muscovites or Russians.

Hnefatafl, like other tafl games, is a deterministic, asymmetric, 2-player complete information game played on a square grid. The defending player tries to move the king to an edge or corner of the board while protecting it with their other pieces, while the attacking player closes in from all sides, trying to prevent the king from escaping. It's important to note that even though the game is asymmetric, many variants turn out to

[6]The word *hnef* means *fist* and refers to the king piece. *Tafl* means *table* or *board*.

be relatively balanced.

This thesis focuses on 13×13 historical hnefatafl using the Parlett (1999) board layout without restricted corner squares (from now on simply called 13×13 hnefatafl), the most balanced hnefatafl variant.[7] It has no measurable bias towards either player if we focus only on strong players playing full matches, meaning that each player plays as both the attacker and the defender. When partial matches and weaker players are also included, it has a 3% bias in favor of the attacker. By comparison, the more popular 11×11 Copenhagen hnefatafl has a 42% bias towards the defender. A general observation about tafl games is that more experienced human players appear to be better at defending, while inexperienced players tend to be better attackers. This may be due in part to the relative ease and lack of planning required to move the king towards the edges compared to capturing the king.

An upper bound on the size of the state space of tablut was estimated to be 1.4×10^{27} by Galassi (2019), comparable to Othello but much smaller than chess. However, the state space of 13×13 hnefatafl is likely to be much higher, as it is played with similar rules, but on a much larger board. In summary, 13×13 hnefatafl is asymmetric but balanced, and sufficiently complex to pose a challenge to general game players.

3.1 Rules

13×13 hnefatafl has a game board with 13 rows and 13 columns (figure 1). The center square is the throne. The defender has 17 game pieces: 16 regular pieces and one king. The attacker has 32 regular game pieces, twice as many as the defender.[8] First, I will discuss the rules of the game. Then, I will address some practical considerations that become relevant when the game is formalised as a GDL game description.

The attacker begins and the players then take turns, moving one of their pieces each turn. Each piece can move any distance in a straight line as long as all squares on its path are empty, like the rook in chess. The king moves in the same way as regular pieces, with one exception. The throne is *restricted*: regular pieces can pass through the throne when it is empty, but only the king can end his turn on the throne.

All pieces, including the king, can participate in capturing opponents. A piece is captured if it is closed in on both sides (either vertically or horizontally) by opposing

[7] Based on statistics from http://aagenielsen.dk/tafl_balances.php.

[8] This ratio between attacking and defending pieces (excluding the king) is typical for tafl games.

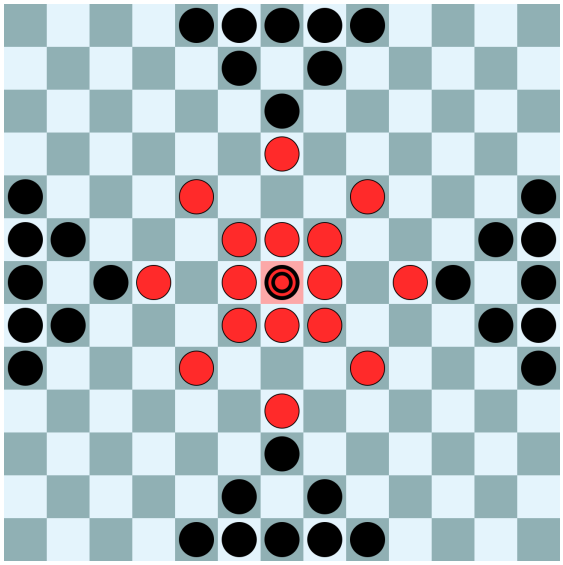

Figure 1: *Initial positions of 13 × 13 historical hnefatafl using the Parlett board layout without restricted corner squares. Red pieces are defenders, black pieces are attackers. The king occupies the throne (center square, red background).*

pieces, but only if this is the result of the opponent's move. Moving one of your own pieces in between two opponents does not result in its capture. A piece is also captured if it is closed in between an opposing piece and the empty throne. Captured pieces are removed from the board. The king cannot be captured if he is on the throne or next to it. Instead, he can be killed: if the king is surrounded on all four sides by opposing pieces, or by three opposing pieces and the throne, he is removed from the game.[9]

If the king ends its turn on one of the edge squares, the defending player wins the game. The attacking player wins if the king is captured or killed, or if all defending pieces are surrounded by attacking pieces, so that they cannot escape unless an attacking piece moves away. If a player has no legal moves available on their turn, they lose the game. If either player forces the same board position to be repeated for the third time, they also lose the game.

For the GDL implementation of this game, a 500 turn limit was added, because GGP games must be finite. However, games should rarely (if ever) take that long in practice,

[9]This difference between killing and capturing the king has only become apparent after a recent (yet unpublished) translation of the tablut rules that fixes a previous mistake. See http://aagenielsen.dk/tablut_translations.html and http://www.tsalo.fi/tablut.html.

as human matches of 13×13 hnefatafl take about 57 turns on average.[10] Additionally, the rule where the attacker wins by surrounding all defenders was left out in the game description, because it would lead to intractable computational costs. This is not an issue, because this situation is uncommon and unlikely to lead to a win for the defender. In the perpetual repetitions rule, where a board position cannot occur three times during the same game, the player who has more choice in preventing the repetitions is normally marked as the aggressive player, and he or she is the one who loses the game. This is not precise enough for a game description, so the rule was implemented by letting the player who directly causes a board position to be repeated for the third time by taking a move on their own turn lose the game. These changes address practical concerns of the implementation while preserving the essence of the game, because the strategy remains almost entirely unaffected.

4 Methods

4.1 Software

A link to the source code for 13×13 hnefatafl is included in the appendix, section A.2. The rules were formalized in infix GDL, saved as a `.hrf` file, converted to a prefix GDL (`.kif`) file using the Stanford GDL converter[11] and validated using Eclipse with the Griddle plugin.[12]

A selection of four players to take part in the tournament was created with the criterion of including different approaches to GGP: `MCTS` is a simulation-based approach, `Minimax` uses the minimax algorithm with α-β pruning, `Heuristic` uses heuristic evaluation function generation, and `Random` selects a random legal move. All players were generated by the Configurable Game Player software[13] and represent the game as a state machine using a logical prover based on the FOL resolution procedure (Russell & Norvig, 2009, p. 345). The open-source ggp-base project[14] was used to implement a locally hosted tournament between the general game players.

[10] Based on statistics from http://aagenielsen.dk/tafl_spillaengder.php.
[11] http://ggp.stanford.edu/public/gameconverter.php
[12] https://github.com/AlexLandau/griddle
[13] http://www.ggp.org/cs227b/player.html
[14] https://github.com/ggp-org/ggp-base

4.2 Design

The experiments were designed to explore the two properties mentioned in the introduction, section 1.2: intelligence and generality. In all experiments, the independent variables are player type and player role assignment (attacker vs. defender), and the dependent variable is victory ratio (% of matches won). First, to test the intelligence of general game players, the `Random` player was used as the baseline performance measure. Every other player type played 10 matches against the `Random` player, 5 as the defender, 5 as the attacker. Then, to test the generality of general game players, each one played 10 matches against another player of the same type, again equally distributed between the two player roles. Another possible design is a counterbalanced tournament, but this would require a large number of matches for each player combination, making it much less practical within the scope of this thesis.

Note that the `Random` player is not expected to perform equally well in both player roles, even though the game is balanced for humans. The likelihood of the king randomly ending up at one of the 44 edge squares is higher than that of two attacking pieces being placed on either side of the king before it moves again. For more intelligent players (up to human-level intelligence), the victory ratio should shift towards 0.5, indicating a balanced game.

4.3 Procedure

All matches used a start clock of 30 seconds and a play clock of 15 seconds. Matches were never run in parallel to make sure enough computational resources were available for both players in every match. Finally, the number of won and lost games was recorded for each player in both player roles. Four matches, each of which lasted for over three hours and more than 500 turns, were terminated and omitted from the results. Continuing these matches was not practically feasible and unlikely to lead to a victory for either player, because the turn duration kept increasing with every turn.

5 Results

Initially, ten matches of historical 13×13 hnefatafl were recorded with the `Random` player as both the attacker (🗡) and the defender (🛡). The results are displayed in table 3. In all matches, the turn duration increased steadily over the course of the match,

Match	1	2	3	4	5	6	7	8	9	10
Turns	258	404	90	412	489	242	278	270	342	267
Winner	⛊	⛊	⛊	⛊	🗡	⛊	⛊	⛊	⛊	🗡

Table 3: *Results of 10 matches of historical 13 × 13 hnefatafl,* `Random` *vs.* `Random`*.*

starting at multiple turns per second and approaching the play clock of 15 seconds around turn 300 on average. Subsequent turns did not result in timeout errors, even though the turn duration (sometimes greatly) exceeded the play clock. For all other player types, all matches resulted in timeout errors on every turn, to the extent that they were functionally equivalent to the `Random` player.[15] As a result, only the generality of the `Random` player was measured. The generality of other players could not be determined as none of the corresponding matches produced meaningful results. Player intelligence is measured relative to the `Random` player, and since this was the only player without timeout errors, the intelligence of the players remains unknown.

6 Discussion

To summarize the results, 8 of the 10 `Random` vs. `Random` matches were won by the defender. This trend, while not significant, could indicate that 13 × 13 hnefatafl is not balanced when moves are selected randomly. If future results confirm this effect, it is likely that the observed balance in human games is not an emergent property of the game's structure or rules themselves, but arises from intelligent decision making by the players. Whether the degree to which a game such as hnefatafl is balanced is directly linked to the level of intelligence of the agent remains an open question. Different problem solving approaches (such as different general game players) are directly comparable in the sense that their win ratio in a tournament can be compared, but this does not imply that matches between more intelligent players will always be more balanced. It is entirely possible that a certain agent is particularly good at attacking, but merely average at defending, or the other way around. The human approach to problem solving apparently leads to balance in this specific game, and while this does require a certain level of intelligence, it does not mean that more intelligent players will play more balanced

[15] On a timeout error, the server picks a random move for the player, see section 6.1.

games in general. However, we cannot compare the human approach to other approaches since humans are much more intelligent than any other agent, and because much is still unknown about how human intelligence actually works.

6.1 Turn duration

If the server does not receive a legal move from a player before the turn timer runs out, the server picks a random legal move for that player. If this happens often enough, the match ceases to be representative of the player's skill. Since the Monte Carlo tree search players frequently timed out in these experiments, the experiments do not provide any meaningful information about how balanced the game is for two `MCTS` players. This holds for any player attempting to look ahead in the game tree. Since planning is a fundamental aspect of intelligence and essential for complex games like hnefatafl, none of the experiments contribute to answering the research question, with the exception of the `Random` vs. `Random` matches.

In these matches, the play clock was not being used by the players to submit their moves, because the submitted moves were already displayed at the very start of each turn while the turn timer was still running, sometimes long after exceeding the play clock of 15 seconds. Also, the `Random` player simply picks an arbitrary move from the list without calculating any future states, so it only requires minimal computation. Therefore, the increasing turn duration is likely due to the computational cost of calculating the next state based on the current state and the most recent move, which becomes increasingly more complicated as the game progresses, and the state history increases in size.

The obvious question is: why does the turn duration increase so rapidly? The first hypothesis, discussed in section 6.1.1, is that the game description contains an error. The second hypothesis is discussed in section 6.1.2: either GDL or ggp-base has some limitation that makes it unsuitable to formalize certain kinds of games.

6.1.1 Game description analysis

First, I attempt to determine whether the game description contains an error by systematically checking all of its components. A number of variant problems were generated to determine which, if any, of the components of the game description was causing the issue using a process of elimination. This section will occasionally refer to these variant problems, which are detailed in the appendix, section A.1. I will discuss all game

description components separately below.

1. **Roles.** The player roles (one attacker and one defender) are trivial and explicitly defined in a way that is functionally and syntactically equivalent to numerous other game descriptions that are not subject to this issue. Therefore, the player roles cannot be the cause of the problem.

2. **Base propositions.** The base propositions provide constraints for other predicates by defining a set of valid variable assignments, reducing the time needed to check variable assignments by limiting the options in advance. An important rule for these base propositions is that they must include all valid variable assignments for a given predicate that could theoretically occur during a game. Additionally, good base propositions include little else, so that they efficiently prevent the repeated checking of invalid combinations. The only way to know if the base propositions could be any more efficient than they are is to actually make them more efficient, but they were already carefully designed with the goal of being as efficient as possible. While inefficient base propositions could be slowing down the game considerably, simply removing them would likely make the game even slower. This makes the base propositions difficult to analyse. Nonetheless, some base propositions were removed in various variant problems. None of them resulted in a measurable decrease in turn duration, except variant problem 11, which removed the state history.

3. **Inputs.** The inputs are similar to base propositions, but for the legal moves of the game. An input defines what kinds of actions are valid, without taking into account the state of the game. For instance, in 13×13 hnefatafl, moving a piece from an existing position on the board to another position should correspond to one of the input predicates, while moving it to (55, 108) should not be included in any input predicate. Whether a move with a valid input is actually legal in a certain specific game state is determined by the legal moves section below, but the legal moves in any given state must be a subset of the input predicates. The available moves could be calculated very quickly, because `Random` vs. `Random` matches had a very short turn duration, even though `Random` players also need a list of possible moves in a given game state.

4. **Initial state.** The initial state includes the initial board positions. All cells with

their initial values are explicitly defined by grounded statements and inspection shows these statements to be trivially correct. Furthermore, grounded statements like these cannot be the cause of the problem, since they require no variable matching. Finally, the problem occurs in most states, not just the initial state. Therefore, no changes to the initial state were made in any of the variant problems, unless required to accommodate other changes, such as a 9×9 board.

5. **Legal moves.** The legal moves are a subset of the inputs. They describe which moves a player is allowed to do in a specific state. The legal moves were changed in a number of variant problems that either restricted them (7, 8) or allowed for more freedom instead (10). However, none of them resulted in a noticeable turn duration decrease. This is surprising, as a significant increase or decrease in possible actions should radically change the state space size and thus slow down the game, but no such effect was observed. It is possible that some other factor has such a great effect on the turn duration that even a significant expansion or reduction of the state space has little effect in comparison.

6. **Update rules.** The update rules determine what the game state will look like after a state transition; together, they fully describe the resulting state (a specific set of predicates) based on the moves of the players and the current game state. The update rules are not only used after an actual player move, but may also be used by players to plan ahead by inspecting states that would result from hypothetical player moves. That makes this component of the game description a likely candidate for the cause of the problem. Almost every update rule is tested in some variant problem (4, 5, 6, and 10) as a side effect of testing other aspects of the game description. Crucially, one particular predicate could not be tested: cells where nothing happens on a given turn (no pieces moves into it or away from it, and no piece is captured or killed there) should remain unchanged and keep their old value. This rule includes a negation, which uses negation as failure (NAF) in GDL. However, removing this rule is not possible as it would result in an incomplete game state where almost all of the cells are removed from the game entirely after the first turn. Even though the relevant part of the game state is limited to $13^2 = 169$ cells, the application of NAF to explicitly represent the lack of change to all but a few of the cells each turn can be seen as an instance of the frame problem. Interestingly, game descriptions for other games seem to employ

similar rules based on NAF for this purpose, and they do not run into issues with the turn duration. Therefore, this rule is unlikely to be the culprit.

7. **Terminal states.** The terminal states specify the conditions that cause the game to end. Every terminal state should have a corresponding goal (see below), so that the players can be rewarded utility depending on how the game was terminated. The terminal states worked correctly in all observed matches, and only become relevant at the end of the game. Still, these predicates must be checked for every state to see if it is terminal, so this could well be responsible for a slower transition between states. The perpetual repetitions rule was removed in variant problem 11, since the entire state history was removed. Other variant problems removed the terminal states corresponding to capturing (5) or killing (6) the king, or moving the king to the edge of the board (7), as well as the terminal state where the player whose turn it is has no legal moves left (3).

8. **Goals.** The goals define rewards for certain specified conditions, most commonly terminal states. The goals for hnefatafl are similar to those of many other games described in GDL: the winner gets 100 points, the loser gets none. These statements are trivial and similar to many other game descriptions, so they were not changed in any of the variant problems.

9. **Supporting concepts.** Supporting concepts include rules for capturing opponents, killing the king, movement, etc. All of these rules are referred to by other predicates in at least one of the other sections of the game description. A number of variant problems were created to test specific supporting concepts by removing them, such as capturing regular pieces (4), capturing the king (5), killing the king (6), and capturing pieces by simply moving into their space (10). None of these variant problems resulted in faster state transitions.

10. **State history.** The state history is used to keep track of the number of repetitions of any given state since the start of the game. This is required to prevent endless loops of game states, thereby keeping the game finite; a strict requirement for GGP game descriptions. A rule preventing perpetual repetitions is also included in the rules of this particular variant of hnefatafl. In variant problem 11, the state history was removed, which resulted in a somewhat reduced increase in turn duration, but still caused timeout errors on every turn for `MCTS` players.

In summary, the increased turn duration could be partially explained by the state history, possibly compounded by other predicates in the base propositions, update rules, or supporting concepts. The other components (roles, inputs, initial state, legal moves, terminal states, and goals) are unlikely to be the cause of the problem.

6.1.2 Limitations in general game playing

Second, I investigate the possibility that the increased turn duration is due to shortcomings in the GDL language specification, the GGP specification, or its implementation in the ggp-base software. As the preceding section shows, a few possible explanations for the increasing turn duration still remain in the game description, so it would be frugal to assume the problem lies with GDL or GGP itself. Moreover, a great number of game descriptions in GDL are hosted online and appear to function quite well with ggp-base. However, it would be equally remiss to ignore the possibility entirely that the problem lies with GGP or GDL. GDL as a language must be sufficiently expressive to represent any kind of game, which is why it is based on (epistemic) logic, but this comes at a cost of increased complexity when parsing, checking and applying its rules. As a result, troubleshooting these issues can be incredibly time-consuming because the GDL code is not actually executed in a runtime environment, but interpreted by the general game players. Understandably, ggp-base is not developed with the intent of making it easy to see how much time certain rules take to compute and, more importantly, why.

Computational complexity is another possible cause for the increased turn duration and timeout errors. Perhaps using a 13×13 board for a game as deep and complex as hnefatafl is simply too demanding for use in GGP on current hardware. Two variant problems were generated to test this hypothesis: hnefatafl on a 5×5 board (1) and hnefatafl on a 9×9 board (2). The former is too trivial to offer meaningful results and the latter did not result in faster state transitions and caused timeout errors for `MCTS` players just as in 13×13 hnefatafl. Thus, it seems that the issue is not one of complexity of the state space.

7 Conclusion

Despite its name, GGP is not only applicable to game playing. Advances in GGP can be applied to other domains, since GGP aims to develop domain-independent problem solving methods for novel problems. In this way, GGP is one of many approaches to more

general artificial intelligence. However, there is a long way to go for GGP as it is much worse at learning how to play games than state of the art machine learning approaches such as MuZero. On the other hand, its niche of understanding the rules logically and reasoning to their implications is still valuable, since it is a fundamentally different task than efficient pattern recognition based on data. Like many fields in AI have done in the past, GGP raises the bar for what we can and should expect from artificial intelligence, and shows that symbolic AI still has an important place in science today.

In attempting to put the generality of various general game players to the test, this thesis ended up testing the generality of GDL and GGP instead: "can 13×13 hnefatafl be represented using GDL and played as a GGP game?" This remains an open question, and the answer so far is "no". The main research question of this thesis was "How does asymmetry in balanced games affect general game players?" To answer it, GGP must be able to handle 13×13 hnefatafl correctly, or possibly another asymmetrical balanced game, but these are few and far between.[16] These insights are a small step in the right direction, even though the main research question has not been answered.

Further research might look into other ways to answer the research question. One possible approach would be a combination of measures that mitigate the effect of exponentially increasing turn duration. For instance, hnefatafl on a 7×7 or 9×9 board without a state history, perhaps simplifying or removing other rules as well. Another possibility is to find a way to modify the game description such that it still represents 13×13 hnefatafl, but can be used to compute state transitions much more efficiently. Yet another option is to look into other balanced asymmetrical games, or to observe under what conditions asymmetrical games result in balanced matches for certain general game players.

8 References

Ashton, J. C. (2010). Linnaeus's game of Tablut and its relationship to the ancient Viking game Hnefatafl. *The Heroic Age: A Journal of Early Medieval Northwestern Europe*, *13*, 1526–1867. https://www.heroicage.org/issues/13/ashton.php

[16] There are no statistics to support this, but Root is a popular modern board game claiming to be asymmetrical, yet balanced. If empirical research into the alleged balance of the game is an option, this could be an interesting avenue for further research.

Björnsson, Y., & Finnsson, H. (2009). Cadiaplayer: A simulation-based general game player. *IEEE Transactions on Computational Intelligence and AI in Games*, *1*(1), 4–15.

Björnsson, Y., & Schiffel, S. (2013). Comparison of GDL reasoners, In *Proceedings of the IJCAI-13 workshop on general game playing (GIGA'13)*.

Clune, J. (2007). Heuristic evaluation functions for general game playing, In *AAAI*.

Cropper, A., Evans, R., & Law, M. (2019). Inductive general game playing. *Machine Learning*, 1–42.

Finnsson, H., & Björnsson, Y. (2008). Simulation-based approach to general game playing., In *AAAI*.

Galassi, A. (2019). *An upper bound on the complexity of tablut* [Unpublished manuscript]. Unpublished manuscript. http://ai.unibo.it/sites/ai.unibo.it/files/Complexity_of_Tablut_0.pdf

Genesereth, M., & Björnsson, Y. (2013). The international general game playing competition. *AI Magazine*, *34*(2), 107–107.

Genesereth, M., Love, N., & Pell, B. (2005). General game playing: Overview of the AAAI competition. *AI magazine*, *26*(2), 62–62.

Genesereth, M., & Thielscher, M. (2014). General game playing. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, *8*(2), 1–229.

Gent, I. P., Nightingale, P., Rowley, A., & Stergiou, K. (2008). Solving quantified constraint satisfaction problems. *Artificial Intelligence*, *172*(6-7), 738–771.

Goldwaser, A., & Thielscher, M. (2020). Deep reinforcement learning for general game playing, In *Proceedings of the AAAI conference on artificial intelligence*, New York, AAAI Press.

Koriche, F., Lagrue, S., Piette, É., & Tabary, S. (2016). General game playing with stochastic CSP. *Constraints*, *21*(1), 95–114.

Méhat, J., & Cazenave, T. (2010). Ary, a general game playing program, In *Board games studies colloquium*.

Möller, M., Schneider, M., Wegner, M., & Schaub, T. (2011). Centurio, a general game player: Parallel, Java- and ASP-based. *KI-Künstliche Intelligenz*, *25*(1), 17–24.

Parlett, D. S. (1999). *The Oxford history of board games*. Oxford University Press, USA.

Pell, B. (1992a). Metagame in symmetric, chess-like games. In J. van den Herik & V. Allis (Eds.), *Programming in artificial intelligence: The third computer olympiad*. USA, Ellis Horwood.

Pell, B. (1992b). Metagame: A new challenge for games and learning. In J. van den Herik & V. Allis (Eds.), *Programming in artificial intelligence: The third computer olympiad.* USA, Ellis Horwood.

Russell, S., & Norvig, P. (2009). *Artificial intelligence: A modern approach* (3rd). USA, Prentice Hall Press.

Schiffel, S. (2010). Symmetry detection in general game playing, In *Twenty-fourth AAAI conference on artificial intelligence.*

Schiffel, S., & Thielscher, M. (2007). Fluxplayer: A successful general game player, In *AAAI.*

Schiffel, S., & Thielscher, M. (2014). Representing and reasoning about the rules of general games with imperfect information. *Journal of Artificial Intelligence Research*, *49*, 171–206.

Sebbane, M. (2001). Board games from Canaan in the early and intermediate bronze ages and the origin of the Egyptian senet game. *Tel Aviv*, *28*(2), 213–230.

Silver, D., Hubert, T., Schrittwieser, J., Antonoglou, I., Lai, M., Guez, A., Lanctot, M., Sifre, L., Kumaran, D., Graepel, T., Et al. (2018). A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science*, *362*(6419), 1140–1144.

Thielscher, M. (2011). GDL-II. *KI-Künstliche Intelligenz*, *25*(1), 63–66.

Thielscher, M. (2017). GDL-III: A description language for epistemic general game playing, In *The IJCAI-16 workshop on general game playing.*

Graphical assets

🗡 Sword by IconMark from the Noun Project.

🛡 Shield by Kimmi Studio from the Noun Project.

Appendix A Implementation

A.1 Variant problems

For each of the variant problems indicated in this section, one or more matches between two `Random` players were observed, as well as between two `MCTS` players. All variant problems are identical to 13×13 hnefatafl in every regard except for the modifications outlined below. The issue was prevalent in all of the variant problems.

1. Hnefatafl on a 5×5 board. The king is in the center, at (3,3), with attackers at (1,3), (3,1), (3,5), and (5,3), and defenders at (2,3) and (4,3). The computational complexity of this variant problem is more similar to tic-tac-toe than to 13×13 hnefatafl. `Random` vs. `Random` matches take 3-20 turns each and `MCTS` vs. `MCTS` matches take 2-8 turns each. More importantly, `MCTS` does not cause timeout errors in 5×5 hnefatafl, presumably because the game is trivial enough that a few moves is usually sufficient to reach a terminal state for both `Random` players and `MCTS` players. Therefore, looking ahead in the game tree does not incur computational penalties to the point where it becomes practically infeasible.

2. Hnefatafl on a 9×9 board. This variant problem caused timeout errors for `MCTS` players. While turn durations did increase over time for `Random` players just as in 13×13 hnefatafl, matches of 9×9 hnefatafl tend to terminate in fewer than 300 turns. Thus, the increased turn duration does not impede empirical experiments for `Random` players playing 9×9 hnefatafl, however these would be less interesting since 9×9 hnefatafl is not balanced for humans.

3. Hnefatafl where the win condition that the other player has no legal moves left is removed. In this variant problem, `MCTS` players caused timeouts on every turn. The increase in turn durations for the `Random` player was similar to matches of 13×13 hnefatafl.

4. Hnefatafl where none of the pieces can be captured. In this variant problem, `MCTS` players caused timeouts on every turn. The increase in turn durations for the `Random` player was similar to matches of 13×13 hnefatafl.

5. Hnefatafl where the king cannot be captured, only killed. In this variant problem, `MCTS` players caused timeouts on every turn. The increase in turn durations for the `Random` player was similar to matches of 13×13 hnefatafl.

6. Hnefatafl where the king cannot be killed, only captured. In this variant problem, `MCTS` players caused timeouts on every turn. The increase in turn durations for the `Random` player was similar to matches of 13×13 hnefatafl.

7. Hnefatafl where the king is not allowed to move. In this variant problem, `MCTS` players caused timeouts on every turn. The increase in turn durations for the `Random` player was similar to matches of 13×13 hnefatafl.

8. Hnefatafl where none of the game pieces is allowed to move. In this variant, the third turn always ends the game, because is is the third repetition of the same board configuration, causing the first player to lose the game.

9. Hnefatafl where none of the game pieces is allowed to move and the no perpetual repetitions rule is removed as well. Now the game continues indefinitely, and every new turn is identical to the first, offering very little information on what is causing the issue.

10. Hnefatafl where movement of the game pieces is not limited to a straight line that is not blocked by other pieces. Instead, the pieces can move to any square, removing any piece already present in that square.

11. Hnefatafl without a state history. In this variant problem, the number of previous occurrences of every unique board configuration is removed. Therefore, there is no way to check for repeated board posititions, an important rule similar to remise in chess. Additionally, this variant problem is no longer guaranteed to be finite unless an arbitrary turn limit is added, which is a strict requirement for GDL game descriptions. Timeout errors were present for `MCTS` players on every turn, just as in 13×13 hnefatafl. While the turn duration did increase with every turn for `Random` players in this variant problem, the effect was much less pronounced than in 13×13 hnefatafl, reaching 2 seconds around turn 700 and 3 seconds around turn 800. This seems to indicate that the turn timer still increases exponentially, even after the removal of the state history which turned out to slow down the game significantly. It is possible that the state history is just one of the issues leading to timeout errors in the `MCTS` vs. `MCTS` matches.

A.2 Game descriptions

All game descriptions used in this thesis, as well as a number of supplementary files, are made publicly available as a git repository hosted on GitLab.[17]

[17] https://gitlab.com/thomas.dingemanse/ggp-balanced-asymmetry