UCLouvain

École polytechnique de Louvain

The chess literature is for the most part of a purely technical nature. It deals with the play and not with the player and his way of thinking; it treats the problem and not the problem solver.

Adriaan D. de Groot

Contents

Summary

The field of General Game Playing (**GGP**) usually yearns to create agents that are able to win any game as efficiently as possible. Utility functions are then easy to find: **winning the game**, or achieving a better score, nets you better results. This resulted in game playing becoming a testbed for artificial intelligence. New techniques are often tested on games such as Chess, Go or Checkers where the environments are easy to formally define, while the task is still being tied closely to human intelligence.

Superhuman-level has been achieved in a great number of games past the 2010s. The first notable instance was Chess world champion Garry Kasparov losing to DeepBlue, a chess playing computer, in 1997 during a six-game match. Researchers became interested in games where heuristics were harder to encode, such as Go. Only in 2016 did AlphaGo become the first computer Go program to beat a 9-dan professional player, Lee Sedol. It did so using novel techniques, such as Deep Neural Networks (DNN) trained using self-play to evaluate positions and Monte-Carlo Tree Search (MCTS) to search the game tree, which are now staples of superhuman-level game playing agents.

Such level of play was only achieved in agents playing specific games. In such instances, an optimized game representation and evaluation functions can be built by leveraging domain-specific knowledge. **General Game Playing (GGP)** aims to generalize game playing agents so that they can pick up **any game**. In this more general setting, domain-specific knowledge can only be used scarcely, leading to more polyvalent yet weaker agents. Research in the domain of GGP has been kicked off by Stanford's GDL (Game Description Language), a standard language based on first-order logic used to describe games.

While superhuman-level game playing is still a goal of GGP, this report investigates another approach: agents that *mimic human behavior* in games. Game playing agents are often described as "black boxes", human players cannot comprehend what their plans are when playing against them. This makes such agents unable to provide:

- an enjoyable yet challenging experience for opponents
- realistic game playing data
- deeper understanding of games through studying their insights

The first point would be of interest to online platforms. While tentatives have been made to limit the strength of AI opponents through artificial means, like limiting search depth. This indeed reduces the overall strength of the agent, but does not make their playing less convoluted. The second point would help game designers test their creations cheaply and efficiently using human-like agents as gatherers of experience. The third point would be of utmost importance to professional and competitive players trying to better understand the game they play to make a living. This would place AI agents as companions to gain

deeper insights, rather than unbeatable opponents used as a basis for the quality of actions played.

The later points are of particular interest to the **Digital Ludeme Project (DLP)**, which aimed to **model**, **reconstruct** and **map** ancient games and their transmission across history and cultures. Modelling and reconstructing such games was done using the **Ludii Game Playing System**. It features a description language for games that is easier to use than Stanford's GDL (1300 games implemented in Ludii for only 52 in Stanford's GDL) for modelling games. Reconstruction is important because ancient games are often only partially recovered. Rules, equipment, or records of playing data can be missing. In order to fill this gap, Ludii uses artificial agents to generate, test, and analyze plausible rulesets. This methodology can be criticized because of one key flaw: agents used in this process do not mimic human behavior.

Human motivations for playing games can be vastly more diverse than simply winning. From getting into tricky situations for educational purposes, to asserting strength over a weaker player by showcasing skillful plans, or losing on purpose to avoid upsetting the opponent, there are many situations where **winning is not everything.** The general theory of cognition of humans playing games is also vague and thus difficult to formalize. The literature on these questions is sparse and difficult to get into. It encompasses research domains such as psychology and cognitive science at the frontier of game theory. This report aims to provide a survey of the current techniques, roadblocks and concepts that have been explored to create more human general game playing agents.

1

Introduction

The field of General Game Playing (**GGP**) [1] [2] usually yearns to create agents that are able to win any game as efficiently as possible. Utility functions are then easy to find: **winning the game**, or achieving a better score, nets you **better results**.

This led to the field becoming a perfect testbench for new Artificial Intelligence (AI) techniques. The environments are controlled, varied and yet non-trivial. This is mainly due to the fact that games present *long-term rewards* that are difficult to grasp without advanced planning. After all, the ability to play Chess, Go, Shogi and many other tabletop games is the sign of a tactical, logical mind in many cultures [3].

As it turns out, superhuman-level game playing agents are now common ground, and yet said agents deviate markedly from human behavior [4]. After all, human motivations can be vastly more diverse than simply winning. From getting into tricky situations for educational purposes, to asserting strength over a weaker player by making them fall into traps, or losing on purpose to avoid upsetting the opponent, there are many situations where **winning is not everything**.

1.1 Overview

The field of game playing has been a testbed for artificial intelligence ever since its inception. The earliest examples of artificial agents playing games traces back to 1951 in computerized versions of the games of Nim, Chess and Checkers. In this early phase, game playing agents were already capable of achieving sufficient skill to challenge amateur players [5].

This skill level quickly ramped up, with the next milestone for AI in games being achieved in 1997, when the chess computer DeepBlue [6] defeated then world champion Garry Kasparov. This was the first instance of a game playing agent achieving superhuman strength. This trend of superhuman-level agents continued, and became commonplace around the 2010s, culminating with AlphaGo [7] beating 9-dan professional Go player Lee Sedol in 2016. The program used a combination of **Monte-Carlo Tree Search** and **Deep Neural Networks** trained using **Reinforcement Learning** and self-play, a combination of techniques that is now common place and will be explained in further detail in section 2.2.2.

1

It can be argued whether game specific agents really are ”AI”. Artificial intelligence often assumes that the input data to the agent is arbitrary, with a goal to generalize knowledge. In this regard, game specific agents such as DeepBlue, AlphaGo and others can be seen as nothing more than sophisticated domain-specific heuristic rules to drive underlying search algorithms.

1.1.1 General Game Playing

General Game Playing (GGP) aims to cross that gap by designing agents that can play **any game**. Domain-specific knowledge is then eliminated, or reduced to high-level features common to a subset of games i.e. tabletop games. Agents in GGP are Game Playing Systems (GGS), programs that take as input a domain-specific language describing the rules of a game (often called a Game Description Language or GDL) and are then expected to be able to play said game.

The first recorded GGS was **MetaGame** [8], a system able to define, generate and play chess-like games automatically. The real kick off for widespread research and interest in the field of GGP happened with the General Game Playing Stanford project in 2005 [9], organizing competitions to emulate progress.

These systems rely on **game theory** as a general model of games. This field of computer science and mathematics devises a theory of ”games” in the general sense. In common language, what we refer to as ”games” are actually *abstract games*, such as:

- board games, such as chess, Go or Checkers
- video games which are played on a computer
- card games, like Uno or Poker
- and many others

However, game theory encompasses the broader setting of multiple agents or players that can affect their environment or state through their actions and receive rewards based on said actions. Ehrenfeucht–Fraïssé games [10] can be used to prove inexpressibility in first-order logic, protein folding has been expressed as a game [11], social interactions can be modelled this way as well [12]. The formalism of game theory will be defined in finer details in later chapters.

1.2 Importance

While the prospect of human-like game playing agents has appeared in the literature before, applications and techniques are left sparse and there is a lot we don’t yet understand about defining agents that align with human behavior [4].

For video games, human-like agents may be used to replace disconnected players in online settings, or to create believable non-playable characters [13]. Notably, agents that are trained solely for the sake of ”winning” are known to perform terribly when paired with humans in cooperative environments [14].

Other domains, such as human-robot interaction, require agents that are able to cooperate effectively when paired with humans. This problematic requires agents that are able

to understand human motivations, as well as feeling ”humane” enough themselves for us to be willing to cooperate with them.

Our field of interest, General Game Playing, has focused on human-like agents less. One explanation is that while human-likeness is somewhat of a requirement when cooperating with humans, it is not the case for games. Most tabletop games are opposing environments: one player wins by making the other one lose. Worrying about how ”naturally” it plays is not a concern if we simply want to achieve superhuman level.

However, human-like agents could be of use in this field by:

- providing more parameters to tune the strength of the agent believably, allowing superhuman-level agents to match the strength of weaker players while preserving a natural playstyle.
- generating realistic game playing data.
- being a framework for explainable AI, providing deeper understanding of games.

Such agents would be used as **gatherers of experience** rather than simply being incredibly strong opponents.

1.3 Aim

The literature on these questions is sparse and difficult to get into. It encompasses research domains such as psychology and neuroscience at the frontier of game theory and artificial intelligence. While human-like agents have been a topic of research in many neighboring fields such as human-robot interaction (HRI) [15] [16], video games [14] [13] [17] [18] [19], conversational agents [12] or game-playing agents [4] [20] [21], we lack a comprehensive survey of these methods to have a full view of *the state of human-like agents in the litterature*.

This report aims to fill this gap by providing a survey of the current techniques and concepts that have been explored to create more humane agents in other fields, and envision how they may be applied to GGP. The end-goal of this approach is to design and implement a GGP agent that is human-like in its behavior (Figure 1.1).

1

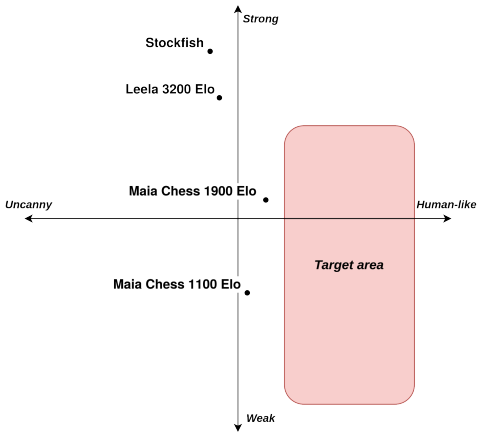

Figure 1.1: Some artificial agents for chess are plotted on this graph based on their performance in the move matching task [4]. The x-axis goes from 0% of human moves predicted (uncanny) to 100% (human player). The y-axis measures playing strength against the current top engine, Stockfish, with a measured Elo of 3641 on the CCRL scale at the time of writing. Current approaches tend to predict human moves with around 50% chance, and the playing strength of an agent does not correlate to it being more or less human. Our aim is to target the red area, focusing on human-likeness rather than playing strength.

1.4 Structure

This report aims to be a comprehensive aggregate of the techniques, methods and open questions in the field of human-like GGP agents for board games. These notions will be explored in the context of the Digital Ludeme Project [22], GameTable Cost Action [23] and more specifically the Ludii game playing system [24], which focus on the preservation, reconstruction and study of cultural heritage through games. Introductory reminders on these notions will be found in chapter 2. Chapter 3 will focus on the question of what it means for an agent to be human-like. This will put forth a few abstract and general criteria which will be used to measure human-likeness, as well as present previous work in this domain. Chapter 4 will then present a new approach based on all the aforementioned results, criterion and methods, applied to the field of general game playing.

2 Context

This chapter will present the general context of this report. It will be structured as follows:

- Section 2.1 will present related projects and organizations, notably the **Digital Ludeme Project (DLP)** and the **GameTable** cost action.
- Section 2.2 serves as a reminder of the formalism of **game theory**.
- Section 2.3 will explain the goals, design and overall architecture of the **Ludii game playing system**.

All of these notions are of peculiar importance to this research as they serve as a baseline from which new techniques are derived. It will be referred to throughout the remainder of the report.

2.1 Related projects and organizations

While human-like game playing agents may be useful in a wide variety of scenarios as seen in section 1.3, this research is done in the context of the **Digital Ludeme Project (DLP)**, and as such will be inclined to serve the needs of said project.

The Digital Ludeme Project [22] is a five-year project funded by the European Research Council. It is hosted by and originates from Maastricht University. Its main goal is to devise methods for modelling, reconstructing and understanding ancient games. These goals will be referred to under the umbrella term of *digital archaeoludology* [25], a field which studies ancient civilizations through their use of games in social settings using modern techniques such as AI.

We do not actually know much about ancient games. Such findings are often made through unearthing equipment, partial records of playouts, or depictions of the game in any way. Ancient games often miss parts of the pieces, the board, the rules, and generally lack comprehensive records of playing sessions. Thus, our modern understanding of these games is based on unreliable modern reconstructions and therefore biased. A model for more formal analysis of these ancient games would be needed to reduce this bias.

This is the aim of the DLP: **applying mathematical models to the problem of filling gaps in our understanding of ancient games** (Figure 2.1).

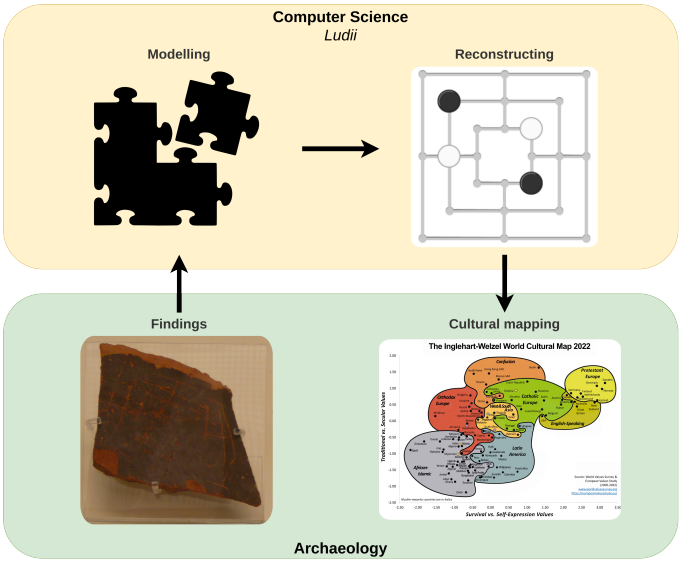

Figure 2.1: The process of reconstructing ancient games requires a cooperation of multiple fields, notably those of computer scientists and archaeologists. Findings made by achaeologists are transmitted to computer scientists, who will model this potentially partial definition of the game into a form that the Ludii game playing system can understand and make use of (**Modelization**). This model is then experimented on, generating potential rulesets and testing their plausibility using AI techniques (**Reconstruction**). Such results are then given back to archaelogists so that they can use the reconstructed game and map it to specific cultures and analyze their social significance (**Mapping**)

This comes into three main phases:

- **Modelization**: implement a huge range of traditional games in a single, easy to use, expandable and playable database.
- **Reconstruction**: fill gaps in our understanding of ancient games in a plausible way.
- **Mapping**: linking the reconstructions back to an archaeological setting.

Currently, this process is done with the help of state-of-the-art AI techniques. A valid criticism that could be made targets the lack of humaneness of agents emerging from these techniques [4]. We can't expect that ancient civilizations were playing games at the level of these agents, or even if our algorithms maximize the right values to begin with. In the past, games could be a medium to assert power, a social activity, and much more. Winning is not everything, and agents mimicking humans could prove deeply useful in reconstructing and understanding ancient games.

The DLP is a **cross-disciplinary project**. It requires expertise from various fields of research and a wide range of abilities. As such, the **GameTable COST action**, directed by its chair Éric Piette, who is also a part of the DLP, provides the cross-disciplinary network of scholars necessary for such an endeavour. It provides a platform on which the DLP can rely when in need of insights from psychologists, archaeologist, historians, or experts in many other fields.

Currently, the GameTable COST action has around 450 members divided into 5 working groups:

- **Search, Planning, Learning and Explainability**, led by Dennis Soemers and Jakub Kowalski, focuses on techniques related to game theory, such as planning, learning, heuristics and AI. Its goal is to develop agents which are explainable and general enough.
- **Cultural Heritage of Games**, led by Walter Crist and Tim Penn, aims to develop new applications for the study of games, and preserve potentailly endangered or lost cultural heritage in the form of games. It is composed of scholars which are specialized in the history of various regions around the globe.
- **Automated Game and Puzzle Design**, led by Swen Gaudl and Younes Rabii, is a working group that focuses on the reconstruction of incomplete games using automatic game generation techniques.
- **Mathematics in Games**, led by Lisa Rougetet and Tiago Hirth, aims to share mathematical aspects of games to better understand the tight bound between games and mathematics. It is mainly interested in the perception, study and use of mathematics in relation to games specifically.
- **Implementation, Dissemination, and Education**, led by Theodora Moullou, has the responsibility of connecting all other working groups and disseminating the results of the COST action to the general public.

This report falls into the first working group, **Search, Planning, Learning and Explainability**.

2.2 Game theory: an introductory reminder

At the center of this work is the notion of **games**, formalized through the mathematical framework of **game theory**. The section will mainly serve as an introduction to important concepts used throughout the report.

Notably, we will present:

- formalisms used to **describe games**, more precisely how to describe sequential games as they are the main subject of this report. Our results can however be generalized to other forms of games, such as stochastic or simulataneous games.
- **modern techniques** for agents to play games, ranging from search algorithms to deep neural networks trained with reinforcement learning and self-play.

2.2.1 Describing games

While there are many formal descriptions for different types of games, this report will focus on three of them: Markov Decision Processes (MDP) [26] (section 2.2.1), Markov Games (MG) [27] (section 2.2.1) and Extensive-Form Games (EFG) [28] (section 2.2.1).

MDPs are a common formalism to describe sequential decision-making problems. They are commonly used in Reinforcement Learning (RL) (described in detail in section 2.2.2)

and describe *single-agent* problems. This is akin to a single player being modelled, able to take actions, observe its environment and potentially learn how to maximize a *reward function*. MGs extend this problem to *multiple agents*, which may or may not have conflicting reward functions.

EFGs are another formalism that is more common in game theory and literature on tree search, and represents specifically sequential decision-making problems i.e. a game where players take turns making actions. Most common tabletop games can be represented in an extensive-form e.g. Shogi, Go, Connect-4, etc.

Markov Decision Process

A MDP [26] is described as a tuple $< S, A, p, r >$ where:

- S is the set of all possible states the environement can be in at any given point. For games specifically, this could be all configurations of a 15x15 Go board.
- A is the set of all possible actions that the agent can take. For games, we often refer to agents as *players* and actions as *moves*. On top of that, some actions can only be legal in certain states. We denote $A(s) \subseteq A$ the set of actions that are legal in a state s.
- $p : S \times A \times S \mapsto [0,1]$ is a function that, given two states $s, s' \in S$ and an action $a \in A$, returns a probability noted $0 <= p(s'|s,a) <= 1$ of the environment transitioning from s to s' after executing a. As an additional constraint, we require $\forall s \in S \forall a \in A \sum_{s' \in S} p(s'|s,a) = 1$ i.e. p must be a probability distribution over all states and actions.
- $r : S \mapsto \mathbb{R}$ is the *reward function*, which defines for any state $sinS$ a real-valued reward denoted $r(s)$ that the agent obtains by transitioning to state s.

There are many other formalisms for reward functions, such as defining rewards from state-action pairs, or tuples $< s, a, s' >$ of the origin state, action and resulting state. Rewards can also be made to be non-deterministic. However, during this report, we'll focus on the given definition as it is the most widely used for games.

MDPs are *discrete-time processes*. We call *trial* a sequence of T state-action pairs, where at any time step $0 <= t <= T$, the environment is in state s_t, the agent selects an action a_t, transitions into a state s_{t+1} $p(s_{t+1}|s_t, a_t)$ and receives a reward $r_{t+1} = r(s_{t+1})$. Note that T can be infinite, in which case the trial never ends. In practice, it we cannot define a precise T for any game, since some games can last for various periods of time. A game of chess can end after 4 moves, or last for multiple hundreds of moves. In such cases, it is more practical to talk about these trials being infinite, and assuming that when a terminal state s_T is reached, all actions transition to the same state s_T and have a reward of 0.

A **policy** $\pi : S \times A \mapsto [0,1]$ is a function that, given a state $s \in S$ and an action $a \in A$ returns a probability $0 <= \pi(a|s) <= 1$ of the agent selecting a when in state s. Note that π must be a probability distribution over all actions. In the context of games, we add another constraint that $\forall s \in S \sum_{a \in A(s)} \pi(a|s) = 1$ i.e. π is a probability distribution over sets of legal actions for all states.

We also define the *returns* G_t that an agent at time step t will receive as $G_t = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$ where γ is a *discount factor* added to control the impact of temporally distant rewards. For games, $\gamma = 1$ is often used because the time it takes to win a game is usually irrelevant.

Derived from these returns, we define a *value function* $V^{\pi}(s) = \mathbb{E}_{\pi}[G_t|s_t = s]$ where $\mathbb{E}_{\pi}$ denotes the idea that we take the expectation of returns under the assumption that the agent acts according to policy π for state s onwards. We can also derive a *state-action value function* $Q^{\pi}(s,a) = \mathbb{E}_{\pi}[G_t|s_t = s, a_t = a]$ that gives the expected returns of executing an action a in state s, and then playing the rest of the trial by following policy π.

MDPs are a natural way to model single-agent tasks. In theory, n-player games can also be modelled under this formalism by incorporating the other players as part of the environment. This approach has hard limits, such as the fact that we must either consider a single opponent model or one MDP per combination of opponent model. As such MDPs are not well suited for multi-player games, and we should turn to Markov Games for such problems.

Markov Games

The main difference between Markov Games [27] and MDPs is that the former explicitly models tasks with multiple agents (players). A MG is a tuple $< k, S, A_1, ..., A_k, p, r_1, ..., r_k >$ where:

- $k >= 1$ is an integer denoting the number of players in the game. Note that a MG reduces to a MDP if $k = 1$.
- S is, as in an MDP, the set of possible states.
- A_i denotes the set of possible actions for player i. We denote $A = A_1 \times ... \times A_k$ the joint action space i.e. all actions selected by each player simultaneously.
- $p : S \times A \times S \mapsto [0,1]$ is similar to the probability of transition from MDPs, except each transition requires all players to select an action.
- $r_i : S \mapsto \mathbb{R}$ is the reward function for player i.

Policies and value functions can be defined just as in MDPs, with the difference that all players select an action at every time step.

Markov Games are a natural way to express *simultaneous-move games*, where every player selects their action at once e.g. rock-paper-scissors. It can also model *sequential games* by restricting actions taken by non-moving players to a *pass* action.

Extensive-Form Games

Extensive-Form Games (EFGs) [28] are a more natural way to express n-player sequential games. They are represented as a tuple $< P, T, U, i, D, I >$ where:

- $P = 1, 2, ..., k, n$ is a finite set of $k >= 1$ players and a "nature" player n modelling stochastic events e.g. dice rolls, a deck of cards, etc.
- $T =< S, A >$ is a tree where each node represents a game state $s \in S$. We denote $S_T \subseteq S$ the set of *terminal nodes* and $S_I \subseteq S$ the set of inner nodes such that $S_I \cap S_T = \emptyset$. A state $s' \in S$ is a successors of $s \in S$ if there is a legal action $a \in A$ that transitions from s to s'.

- $U : S_T \mapsto \mathbb{R}^k$ is a reward function over states $s \in S$, which returns a vector of real-valued rewards for each of the k players.
- $i : S_I \mapsto P$ maps the inner states of the game to a player, called the mover.
- $D : \{(s, s') | i(s) = n, s \in S_I, s' \in S\} \mapsto \mathbb{R}$ models the stochastic decisions of the "nature" player, with a probability $0 <= D(s, s') <= 1$ of transitioning to state s' when in state s. D must be a probability distribution over successors of s.
- $I : \{(p, s) | p \in P \setminus \{n\}, s \in S\} \mapsto \mathbb{P}(S)$ returns the *information set* of a player p in state s. This can be seen as the set of states that are indistinguishable from each other from the perspective of player p in state s.

EFGs add support for *imperfect-information games*, where a part of the state is hidden to agents. This makes them able to encode simultaneous-move games by having each player selecting an action during their turn, hiding this information to other players and finally playing all selected moves simultaneously. However, we will only focus on sequential, perfect information games in this report.

2.2.2 Modern techniques

This subsection will briefly introduce techniques that are used in modern game playing agents. Notably, we will focus on Monte-Carlo Tree Search (MCTS) [29] as a search algorithm and Reinforcement Learning (RL) [30] as a method for learning policy.

These two methods are used in many game playing systems in some form today. Game specific programs like AlphaZero [31] have popularized this combination of methods by learning from zero using *self-play* i.e. only playing against itself.

Monte-Carlo Tree Search

Monte-Carlo Tree Search (MCTS) [29] is a family of algorithms and methods to find optimal decisions in a given domain through random sampling, also reffered to as *rollouts* of the action space. It then builds and maintains a *search tree* according to previously sampled results to direct future sampling.

As a search algorithm, it is so-called *best-first*. In opposition to *depth-first* algorithms such as Minimax [32] with or without $\alpha\beta$ pruning [33], best-first solutions such as MCTS build an *unbalanced tree*, focusing on searching promising actions deeper. As such, it works particularily well in game with a **high branching factor** i.e. many legal actions in a given state on average.

On top of that, its reliance on random sampling makes it **aheuristic**, meaning it does not require any heuristic evaluation function in order to run. The evaluation of actions is solely derived from the rules of the game i.e the value of terminal states and results obtained through sampling actions. This makes it a prime candidate algorithm for GGP, as it requires no domain-specific knowledge apart from a game's rules.

We define a *run* of the MCTS algorithm as an iterative process where four steps (in Figure 2.2) are repeated until we reach a computational limit (time, memory, number of iterations, etc):

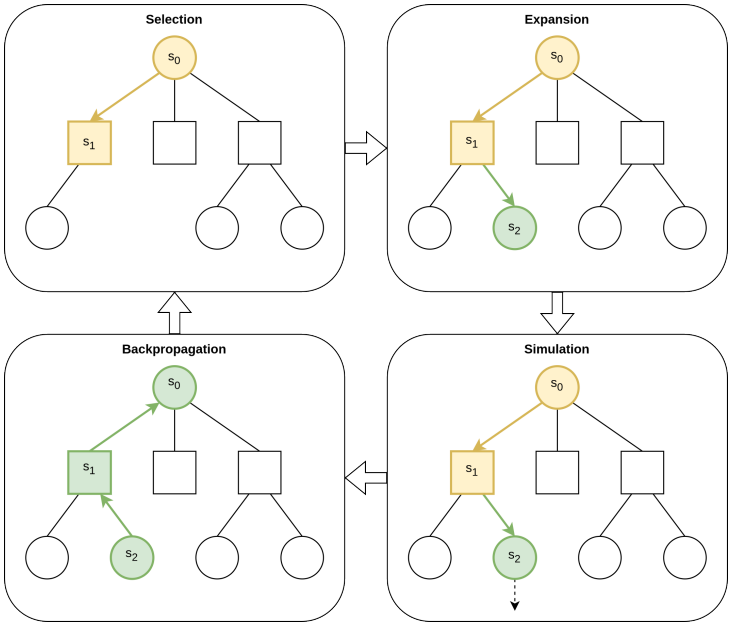

Figure 2.2: Graphical representation of one iteration of the MCTS algorithm. We first **select** a non-expanded node by going down the tree while balancing exploration and exploitation. The selected node is then **expanded**, leading to a sampling of possible results using **playouts**. The sampled result is then **backpropagated** to further direct future iterations of the algorithm.

- the **selection** phase traverses the tree from the root (current state) up to a leaf node following a *tree policy*. The most common of these policies is Upper-Confidence bounds applied to Trees (UCT) [34], which selects at each step the child maximizing the following UCB1 formula.
- the selected node is then **expanded**. One of its children which have not yet been explored is added to the search tree. This child is selected according once again to the *tree policy*.
- a **simulation** or *rollout* is conducted starting from the newly added node. A game is played until completion from the selected state, where actions are usually taken at random. The simulation returns the game-theoric value Δ of the terminal state reached.
- the outcome Δ of the simulation is **backpropagated** through the nodes traversed during the *selection* phase. Each of the traversed nodes updates its total reward X (and potentially other metadata) with knowledge of the simulation's result Δ.

The process being iterative means that MCTS is an *anytime* algorithm: it can be interrupted at any point and yield a usable result.

We can parametrize this base MCTS algorithm through two means:

- the **tree policy**, which defines how we select and expand nodes. Many possible policies exist for selecting nodes, such as GRAVE [35], RAVE [36], etc. When it comes to expansion, the question is mainly that of how many child states should be added to the tree in one iteration.
- the **default policy** used during the simulation phase to decide how actions should be taken. The most common, as stated above, is to randomly select actions until a terminal state is reached. There are however different approaches: one could perform a shallow Minimax search from the expanded node, or place some probability distribution over actions, etc.

Reinforcement Learning

Reinforcement Learning (RL) [37] is an approach to machine learning that takes inspiration from psychology. It aims to simulate the way intelligent life seems to learn from positive and/or negative feedback arising from their interactions with their environment (Figure 2.3).

In usual RL tasks, examples of *good* behavior are not available. Instead, the agent is expected to learn solely from interacting with its environment by taking actions, observing the reward obtained, then adapting its policy.

If the task is modelled as an MDP (section 2.2.1), the aim of RL is to learn an optimal policy π for action selection in the sense that it maximizes its reward.

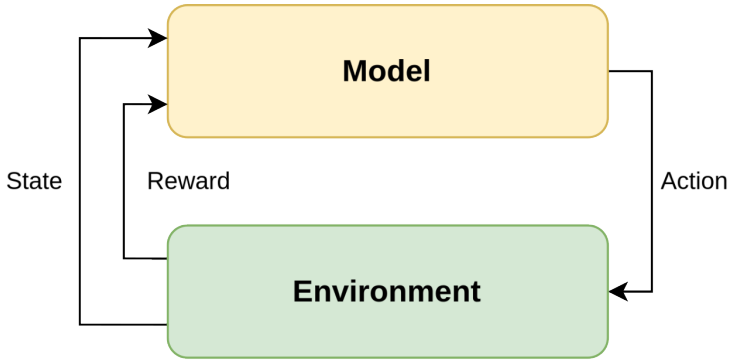

Figure 2.3: Vizualization of the reinforcement learning framework. The agent and environment form a feedback loop where the agent modifies the environment through its actions, receiving a reward and updated observation (state) in exchange. The observed reward and state can then be used to train the agent so it maximizes the reveived reward in future interactions.

2.3 The Ludii game playing system

Ludii [24] is the main component allowing the Digital Ludeme Project to fulfill both modelization and reconstruction of ancient games. It is a general game playing system focused on extensibility and ease of development rather than brute performance.

It is based on the *ludemic approach* to describe games, where games are composed of atoms called *ludemes*. The idea occured to Cameron Brown while compiling his book *Connection Games: Variations on a Theme* [38], during which he noticed that most games were composed of different combinations of the same high-level ideas. This led to the creation of Ludi during Browne's PhD from 2006 to 2009 [39]. 15 years later, this idea was made into a fully-fledged game playing system: Ludii [24].

Ludii is a game playing system applicable to any game. It can be given the rules of any game and be expected to be able to play it. We will further explore how games are described in order to be usable by the Ludii Game Playing System, as well as the means used to learn how to play said games under heavy domain-specific knowledge and time constraints.

2.3.1 Ludii's Game Description Language

Rules of the games are provided in a code-like format, using a domain-specific language which we'll refer to as part of the family of Game Description Languages (GDLs). The term GDL was originally coined for Stanford's GDL, but many other GDLs exist. Ludii rolls out its own Game Description Language (GDL), which we'll refer to as **L-GDL** throughout this report. Instead of extending first-order logic like Stanford's GDL, it opts for a constructive approach based on atomic units of game-related information called *ludemes*. Composing ludemes yields one or more new ludemes.

Some base ludemes are included in the language as keywords. They encapsulate key concepts of games, like the board configuration, pieces, common moves that can occur, etc. Such ludemes are directly derived from their implementation in the Java core of the system [40]. The full grammar of the language is documented in the Ludii Language Reference

[41].

Compared to Stanford's GDL, which is the academic standard, L-GDL is:

- simpler conceptually
- generalizes concepts well
- provides encapsulation

As a simple metric to understand this difference, Stanford's GDL is used in the implementation of around 52 games at the time of writing, while Ludii implements around 1300.

This simple framework does not come at the expense of expressiveness, as both Ludii and Stanford's GDL can express the full range of games [42]. In essence, they are both equivalent to extensive-form games.

The basic structure of an L-GDL program is as follows:

- `(players ...)` is a ludeme used to define players taking part in the game. It can take a single integer when only the number of players is important, or additionnal data such as colours, important zones, directions, etc when needed.
- `(mode ...)` is optional and provides information about the type of game about to be described, notably if it is simultaneous or sequential. If it is ommited, the game is assumed to be sequential.
- `(equipment ...)` is used to defined aspects such as board type, pieces, and other equipments e.g. dices, tokens, etc.
- `(rules ...)` defines how the game is played. It takes as parameters three other ludemes:
  - `(start ...)` rules are actions that are applied before the first turn. It usually constructs the initial board state i.e. initial position of pieces.
  - `(play ...)` rules are used to define legal moves that should be generated given a game state.
  - `(end ...)` rules are conditions under which the game should end for certain players, and defines rewards acquired when such states are attained.

This representation is then compiled down to a `Game` object in Ludii. This object encapsulates and represents static data about the game, such as its state representation, algorithms to generate legal actions or check whether the game ended. Without delving into too much details, it is important to know that boards are represented as *graphs* $G = (V, E, C)$ with V the set of vertices, E edges between said vertices and C cells composed of vertices and edges.

2.3.2 State-action features

During his thesis, Dennis Soemers generalized an approach for learning state-action features for games [43]. State-action features as proposed by Soemers are a *linear function approximator*, as opposed to Deep Neural Networks (DNNs) which are able to express non-linear relationships. While unlikely to match the playing strength of DNN approaches, these approximators have the advantage of [44]:

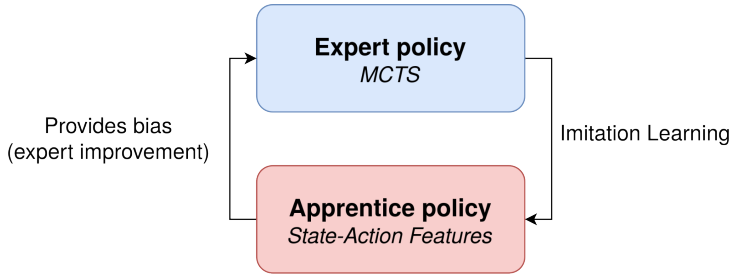

Figure 2.4: Feedback loop of the expert iteration framework. The apprentice learns from the expert by trying to mimick its policy, while the expert itself is biased by the apprentice as it learns. In our specific framework, the expert policy is obtained from an MCTS algorithm, itself biased by learnt state-action features.

- **generality**, as DNNs often require different architectures for different games.
- **iterpretability**, as features are extractible from from the model once learned, and then exploitable by humans [45].
- **lower computation requirements** for training and possibly evaluating.

We describe features $x =< p, a >$ as a tuple of:

- p a *pattern* or set of elements that the features tests for. These elements are in the the *off-board*, *empty*, *friendly*, *enemy*, *owned by player n*, *piece of type t*.
- an action $a =<$ from, to $>$ which the feature weighs. Note that the *from* position is optional e.g. it is not necessary in games such as Go or Tic-Tac-Toe.

Such features are applicable to any game which board's can be represented as a graph [44].

Features are learned using an *expert iteration* [7] [46] is a framework for self-play learning that makes use of two policies: an *apprentice* policy, and an *expert* policy. The expert policy is thought of as a slow, algorithmical model of thought which does not need prior training to function, while the apprentice is a computationally efficient heuristic, but requires training in order to perform.

Both of these policies *affect* and *improve* each other. The apprentice is trained on results obtained through the expert, and then the expert is biased by the apprentice. The usual combination for superhuman-level agents is that of MCTS and DNNs. Here, we'll formalize the combination of MCTS as an expert policy and state-action features as the apprentice (Figure 2.4).

3

Describing and evaluating human-like cognition

Human psyche is a complex enough subject to have multiple fields of research entirely dedicated to it. Throughout this chapter, we'll refer to **cognitive science**, an umbrella term encompassing fields such as psychology, artificial intelligence, neuroscience, etc.

This complexity did not stop researchers in various fields to try to create human-like artificial agents. This is, in fact, one of the subgoals of Artificial General Intelligence (AGI) [47]. Throughout the litterature, scholars have refined methods to evaluate humaneness in an algorithm's decisions, and adapted various results from cognitive science to emulate the human mind as best as possible. [15] [12] [4] [48] [20] [21] [14].

All fields of computer science are not created equal when it comes to research on the topic of human-like agents. Human-Robot Interation (HRI) [15] [16], believable non-player characters [14] [13] [17] [18] [19] in video games or emulating conversation [12] are all *cooperative tasks*. Two or more agents strive to attain a co-dependant objective, whether it is moving through a crowd or making the player or interlocutor engaged.

Game playing, and by extensions General Game Playing, are not commonly cooperative tasks. The most prevalent form of tabletop games is that of two-players zero-sum games, where a player can only gain utility by taking from its opponent. Here, the two agents are in *opposition*. Human-like agents for games have been researched on two occasions: at a time where heuristics were complex, and the human psyche seemed like a stronger model [48] [6], and recently when researchers realized that explainable models might be of use [20] [21].

This chapter will focus on previous results regarding both human thought-process and the application of these results to artificial agents in various fields. The goal is to have a broad understanding of the current state-of-the-art regarding the question of human-like AI, as well as a deeper understanding of the underlying concepts used to make said agents human-like in the first place.

3.1 Human thought: cross-disciplinary perspective

3.1.1 Human cognition in games

Most of the work done in psychology when it comes to understanding how humans think about tabletop games can be attributed to Adriaan de Groot, a Dutch psychologist. His work and experiments are reported in their integrity in his thesis, "Het denken van den schaker" published in 1946 and later translated to english as "Thought and Choice in Chess" [49]. His work focused solely on the game of Chess, but it can be argued that his results are general enough to cover a wide range of board games.

He formalizes the "choice-of-move" problem. That is, given a position, how does the player (subject) selects an action he thinks leads to the best outcome. We are not interested here in whether or not the selected action is objectively the best, but how it was selected. De Groot describes the problem as ill-defined, as there are many considerations to take into account. Notably, he exhibits a difference between laying out a plan and executing one, **planning** and **acting**.

An important factor also formalized was that of *freedom of choice*, aka how many moves can be chosen. De Groot does not translate this to the idea of *legal actions* as thought of in game theory, but rather the number of moves from which the subject really makes his choice, a subset of the legal actions space. Note that his thesis was written before the apparition of computer chess, and while he intuited the notion of "proven good move" to be that of a MiniMax tree, De Groot did not have access to computers for his calculations. However, he still posited from statistical analysis of previous games that there were on average between two and five good moves per position. For reference, chess has an estimated branching factor of between 31 (average over simulated games [50]) and 35 (commonly agreed upon value, altough the original source could not be found).

Other qualitative considerations can be formulated to evaluate how closely a method approximates the human evaluation and planning patterns. Notably, it has been found that humans of different skill levels do not vary as much in depth of search as they do in pattern recognition [51] [49]. This immediately disqualifies "depth limiting" methods as good candidates for a human-like agent. Depth-first algorithms are also deemed bad models, since we tend to focus on a few good moves in best-first fashion. This claim is backed up by the fact that Monte-Carlo Tree Search based search outperforms $\alpha - \beta$ in predicting what moves human players chose in a variety of situations [4].

3.2 Research on human-like agents

Multiple fields have been interested in crafting human-like agents for a long time. The main problem that human-like agents would solve for these is better cooperation with humans. For example, the field of **Human-Robot Interaction (HRI)** has an emphasis on creating intelligent machines that are able to cooperate with human agents to achieve tasks as efficiently as possible.

Sometimes, this cooperation is *implicit*, with a prevalent example being navigating amongst a crowd. [15] Humans make strong assumptions about the behavior of their peers within a crowd in order to not bump into one another while reaching different goals. It has been demonstrated that humans tend to feel more at ease if the crowd behaves in an expected, human-like way. This includes machines that are part of the crowd.

On other occasions, this cooperation is **explicit**. This happens for example in cooperative games, where human and AI are to work together to achieve a given goal. Agents trained through reinforcement learning and self-play achieve superhuman levels when playing together (AI-AI team), but tend to be subpar when playing in a human-AI team. This is alleviated by using more humane artificial agents [14].

This section will present current standard ways of evaluating the human-likeness of an artificial agent, as well as results and approaches from various fields when designing human-like agents.

3.2.1 Evaluating human-likeness

Quantifying human-like behavior is a problem in and of itself. Formally defining such a complex and intricate concept as ”human behavior” is not a feasible goal. However, the literature on human-like agents has come up with a high-level definition that stems from how we test for human behavior in such agents: **a behavior is human-like if it is indistinguishable from that of a human.** This definition is the basis of evaluation methodologies in current research and will be the basis of our own methodologies as well.

3.2.2 Turing tests

The most well known attempt to evaluate human-like behavior is the **Turing test** (Figure 3.1). In said test, human observers are given the task of ranking several unknown agents or observees, some of which are humans themselves, while others are machines, based on their intuition of how ”humane” the agents were. This is a **qualitative evaluation** of human behavior.

This methodology has been widely criticized in the philosophical literature [52] as a test for general intelligence. However, when targeting specific domains, such as tabletop games in our case, it still constitutes a good basis for qualitative evaluation. Another improvement over the basic Turing test is to allow for more open feedback. Humaneness is not based on ranking observees, but rather scaling them (e.g. as a ”percentage of human-likeness). Other qualitative feedback can be obtained that way, like how comfortable the observer felt interacting with the observee, or its aggressiveness when playing a tabletop game. This methodology fits our definition of qualifying indistinguishability and is common in the literature [15] despite being reatively costly to setup in most scenarios. It requires willing participants for the survey, both in the roles of observers and observees, on top of a requirement of anonimity when it comes to the nature of the observees.

3.2.3 Statistical tests

On the quantitative side, previous research on human-like game playing already defined useful ideas. The most prominent method for evaluating an alignment with human behavior is by quantifying the ability of an agent to **predict human actions given a state of its environment** (Figure 3.2).

For example, Maia Chess [4] measures human-likeness of its models by comparing the move chosen by the agent in a given position to the most common moves played by humans. Turnwald et al. [15] employs a similar strategy by measuring the difference in trajectory between humans and robots trying to reach a goal. Other attempts include He

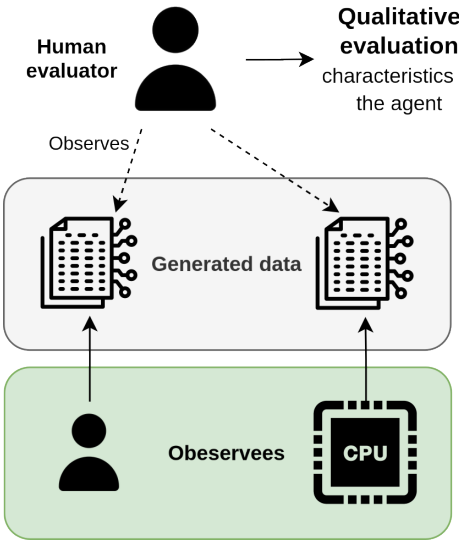

Figure 3.1: Turing Tests involve two groups: the **observers** and **observees**. In Turing's original setup, the observers would interact with observees unknowingly of whether they were machines or humans, then take a guess. Modern approaches are more complete, with qualitative analysis of the observees rather than a simple "human or machine" question.

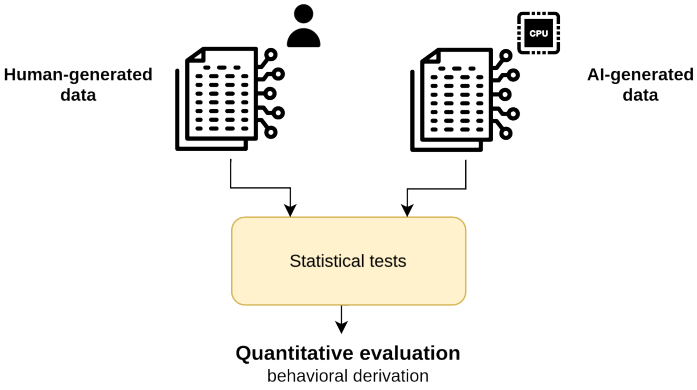

Figure 3.2: Statistical tests compare an expected model of human behavior collected from available data with the behavior emerging from the model. This allows the measurement of a quantitative distance between observed behaviors. This evaluation scheme is heavily dependant on the classification of human-generated data. Said classification can be done by skill level, player archetype (aggressive, passive, tactical), or more involved criteria.

et al. comparison of their dialogue model's decisions with human responses included in conversational datasets [12].

This quantitative evaluation scheme gives tangible results as a measure of distance between behavior, but cannot by itself qualify as a sufficient test for human-likeness. A neural network acting as a classifier from environment states to actions taken by humans could achieve great results under this framework, while it could be argued that it does not exhibit "human-like behavior" by itself. This can be countered by generalizing the framework and using sufficiently big datasets as to avoid this bias. For general game playing, statistical testings over a wide range of games with sufficiently large datasets should help in reducing this bias significantly.

Another important feature of this evaluation scheme is its capacity to compare an agent's behavior to a pre-processed *behavioral cluster*. For example, one could check how different the choices of aggressive human players are from that of our model. Clusters of skill level are also good examples [4]. This allows researchers to fit their models to a specific type of human player rather than *humans* in a general sense.

3.2.4 Past attempts at creating human-like agents

Methods already exist for mimicking human behavior in agents. Most of these methods, however, come from the field of **Human-Robot Interaction (HRI)** or **General Video Game Playing (GVGP)**, and have not been tested or applied to the task of GGP specifically. This section will discuss said existing methods as well as evaluate their applicability to GGP.

It is widely agreed on that modelling humans as optimal agents is counter-productive when designing agents that need to cooperate with us. Most of the work in HRI focuses on planning, evaluating and learning using models of human behavior. How these human models are derived in HRI could be a first step towards understanding and implementing an agent mimicking such a model in GGP [14].

Work of particular interest to our task is that of Choudhury et al. [53], which compares two approach for human modelling: **imitation learning** using data from humans on a particular task, or the more structured and generalizable **"theory of the mind"**.

Experimental results of training agents to cooperate with human models show that the agent is more adaptable to its environment, performing tasks with more varied strategies [14] [15]. Notably, instead of always having a leading/following role in the cooperative task, it tends to switch between these roles when suitable.

Model-free reinforcement learning approaches are often appealing because the agent does not require a "role-model" from which it learns how to interact with the world. However, it often falls short in areas such as explainability, or, more relevant to our problem space, interactibility and humaneness. This has been quantified by Carroll et al. [14] in an experiment where a human-AI team played the cooperative video game Overcooked. Self-trained agents were shown to perform incredibly well when paired together, but poorly when paired with a human. They posit this is due to the "sub-optimal" playing of human players from the point of view of the self-trained agent.

Model-based approaches, instead, allow the agent to learn by observing human actions. In this sense, it is akin to **imitation learning**. An agent is trained on human-generated

data, and is tasked with mimicking the observed behavior. Agents trained this way performed way better when cooperating with humans in Carroll et al.'s experimental setup.

The main drawback of this approach in our case is that we work under the assumption that data arising from human play *does not exist* and/or is heavily limited. On top of that, training for a single game does not generalize well. However, learning how humans perceive the utility of different game states does generalize pretty well and is doable using **inverse reinforcement learning**, a technique which we'll describe more thorougly in subsection 3.2.5.

"Theory of the Mind" (ToM), a term coined by Alison Gopnik [54], advocates for modelling humans using a set of assumptions. It models agency.

Cognitive science literature provides some amount of evidence that we might use ToM as a basis for our own interactions with peers during infancy [55] [56] [57] [58], notably under the assumption that they are rational, and as such make approximately optimal decisions under some objective.

Research done on whether a ToM approach is best for human-agent interaction is often inconclusive due to a few intricate roadblocks arising [53]:

- We do not yet have a precise model of "how humans think". Therefore, any quantitative or qualitative results are derived from using probably subpar models.
- Experiments are often costly, although this point can be somewhat alleviated by using board games as an intermediate.

The problem of designing a ToM approach to general game playing comes in three subproblems:

- defining a **planning algorithm** optimizing for the constrained computational model of humans.
- modelling **rewards** for different tasks, which may or may not be related to winning a game.
- providing a way to **evaluate** heuristics for state-action pairs.

Planning like a human

In GGP, the two predominant family of algorithms for planning are depth-first (Minimax [59], $\alpha\beta$ [33], etc) and best-first (MCTS [29], GRAVE [35], UCT [34], etc). In an effort to mimick human strength, it is often the planning algorithm that is impacted. For example, the maximum depth of the search can be reduced or the program can chose the wrong move randomly.

However, these tweaks do not tend to make game playing systems more "human-like" [4]. Instead, we argue that the currently available algorithms for GGP are not fit to that task and will try to infer a new algorithm from the basis of psychology.

Previous work on the subject of human-like planning has been done, notably for Large Language Models [12] or motion planning in HRI [15] with promising results. These works are based on some form of Monte-Carlo Tree Search, the rationale for such a choice often

being that it is closer to how human cognition works expertimentally [4] [15]. This empirical notion matches the findings from De Groot's experiments on planning in the game of chess [49].

An augmentation of the usual MCTS framework comes from the Dual-Process Theory of Cognition (DPTC) [60] as applied by He et al. to their conversational agents [12]. This theory posits that humans employ two different processes of thinking:

- **System 1** is fast and based on intuition. This sytem is used when the agent is faced with known and learned patterns e.g. endgames in chess, or the eye pattern in Go.
- **System 2** is slow and analytical. It is active mainly when new situations arise and System 1 is unsure of what to do next. System 2 will then take some time to simulate and plan ahead to compensate for the lack of prior knowledge.

The idea posited by He et al. for a more humane conversational agent is to:

- use a policy function as System 1. This policy function is based on a fast neural network.
- use the MCTS algorithm as a fallback when System 1 presents uncertainty, simulating System 2. In their experiment, the forward simulation was done using ChatGPT to create the interlocutor's response in advance.

The notion of uncertainty in the policy function was measured as the value difference between the first and second choices. If both choices are deemed almost equally interesting by the policy function, we assume that it is uncertain about how to proceed. Another idea, probably more applicable to game playing, is to compute the entropy of the returned policy distribution. This is due to the fact that in games, particularly in winning states, more than one action can lead to the desired outcome.

A proposed strategy to still take advantage of the second system in situations where the policy is almost certain that one of two actions is the best is to solely search these two "best" actions. This is done by computing the entropy of the policy, then picking actions to search by removing them from the action set and normalizing the policy. This process is repeated until the policy is too uncertain in regards to an ϵ parameter.

This idea will be explored further in section 4.1.

3.2.5 Human-like agents in tabletop games

The research on tabletop games is not exempt from attempts at creating human-like agents altogether. Experiments on the subject date back to 1991 where Levinson and Snyder designed the *Morph* computer chess program [48]. The idea was to take advantage of pattern matching for the program to learn heuristics. Positions were represented as graphs with attack-defense semantics 3.3. All pieces in the chess board represented a node of the graph, and a vertex was added between pieces if one could attack or defend the other. Weights were then added to patterns in order to produce a heuristic value for states.

The results from this experiment were pretty mixed. On one hand, *Morph* managed to learn values for each piece which were closely tied to the theoretical values given by professional chess players. However, Levinson and Snyder noted that finding the correct

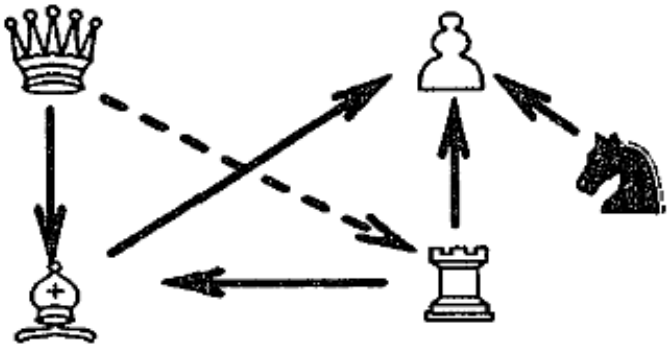

Figure 3.3: Example of an attack-defend pattern as described by Levinson and Snyder [48]. Filled arrows represent a direct path from one piece to another, while dashed arrows represent indirect paths i.e. through a piece.

function to apply to the weights of active features in order to find a good heuristic was a complexe endeavour. Notably, they seemed to be limited by the lack of *non-linearity*.

More recently, Manziuk proposed an approach similarily based on pattern recognition, or *contexts* (Figure 3.4) to find the best moves in connect-4 with a 1-ply search [20].

His approach was based on a neural network, which introduced non-linearity unlike the experiment made by Levinson and Snyder. Results from this approach are also encouraging, with the network performing better than naïve deep neural network approaches when it came to finding the best move to play without any search i.e. through pure pattern recognition.

Maia Chess [4] introduces the task of *move-matching*, a statistical test where the chosen move of the agent is compared to the move chosen by humans on average. The goal here was to maximize the accuracy of the agent in the task of predicting human moves given a game state.

This can be expressed as a *classification problem*. Training a deep neural network on human-generated data obtained from the Lichess platform allowed Maia Chess to outperform other agents such as Stockfish or Leela Chess Zero in the move-matching task.

This approach could be used for GGP by gathering a large quantity of data, something made plausible using the Ludii online platform. However, it is not directly applicable to the task of reconstructing ancient games, since it relies on the availability of human-generated data.

Human-like rewards

Rewards in General Game Playing are often translated one-to-one from scores that can be achieved by completing goals in the current game. For two-player zero-sum games, the typical reward is 1 for winning, 0 for a draw, or −1 for losing. These rewards are solely limited to the game's outcome, and are a useful evaluator of an agent's performance in the literature.

However, human motivation for playing games can be varied. None of the papers cited above [53] [14] [15] deal with this idea even while trying to design human-like agents.

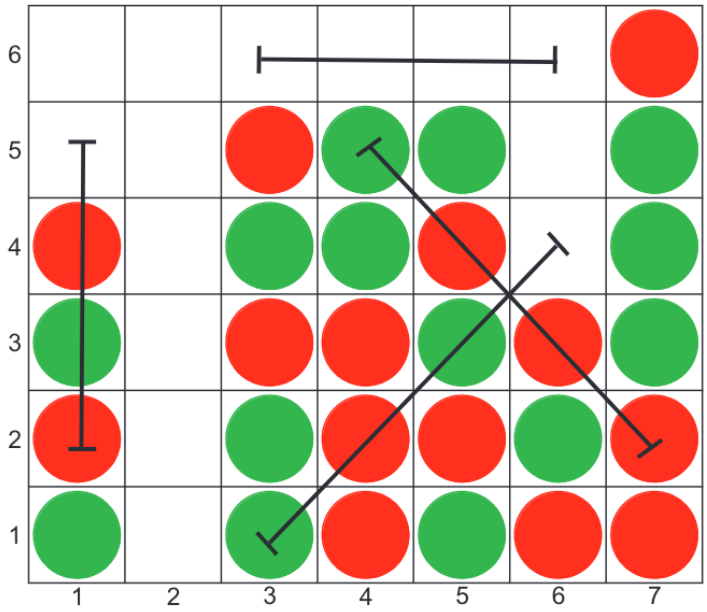

Figure 3.4: An example of the four possible contexts described by Mandziuk in his paper [20]. It matches lines of 4 pieces, in any direction, and for all cells.

Instead, their approaches simply try to maximize the intricate reward (in the case of [14] the score given to players for completing tasks). In the case of human-AI collaboration, this could lead to unpredictable behavior and poor performance when an agent designed solely for maximizing in-game score is paired with a human who wants to simply have fun some other way, or even whose goal is to ruin the AIs plans.

Inverse Reinforcement Learning

Inverse Reinforcement Learning (IRL) [32] [61] is the problem of inferring a reward function from the decisions taken by an agent. Previous works that apply this technique to tabletop games to infer a reward function when it is difficult to describe otherwise [17] or as an opponent model policy [62]. We'd instead want to understand a model trained on human data not to optimize our reward for winning, but instead to make said reward function align better with human considerations. This was one of the potential applications proposed by Russel in its original description of IRL [32]. Some work has been done in this regard for other fields, such as cognitive science [55] or sociology [63], demonstrating the inference of a human's goal by formalizing their actions as an IRL problem.

In our case, IRL could be used to study how humans perceive rewards in games. For example, two-player zero-sum games often consider reward functions to be −1 for the losing player, 1 for the winner, and 0 for both in case of a draw. It would not be far-fetched however to infer that so called "bad loosers" would value losing and winning in an unbalanced way, as would a novice playing against an expert. The potential applications and methods to use will be further described in 4.2.

IRL is often formalized as a Markov Decision Process (MDP) ascribed to the interaction of the observed agent with its environment e.g. a trial for GGP. As a reminder, a MDP is a tuple $< S, A, p, r >$, explained in further details in 2.2.1.

The reward function r of this MDP is unknown, and the observed agent is assumed to be following an optimal policy for the MDP i.e. the policy π that maximizes $V^{\pi}(s)$ for all $sinS$ the set of all possible states. IRL problems are thus expressed as a 3-uple $< S, A, p, D >$. S, A, and p are the same as in MDPs, but we replaced the reward function with: $D = \{(s_0, a_0)_1, ..., (s_T, a_T)_1, ..., (s_0, a_0)_n, ..., (s_T, a_T)_n\}$ the set of observed trajectories. We assume that these trajectories are *perfectly observed*, meaning that we know all actions and states encountered for all timesteps $0 <= t <= T$.

Our goal is then to determine r a reward function that best explains either the policy π if given, or the observed trajectories.

3.3 Closing remarks

This survey of existing methods for human-like artificial agents in different fields and results on human cognition from a general cognitive science perspective help us unveil a few key points on:

- what makes a human-like agent successful in its task.
- the processes from human cognition that come into play, specifically for game playing problems.
- methods to evaluate the humanness of an artificial agent.

4

Human-like model for general game playing

As stated in previous chapters, human-like agents are important to general game playing for different reasons than in the fields where it is usually employed, which often revolve around cooperating with humans.

The goal of humane agents in General Game Playing (GGP) is, in a sense, also to cooperate with humans, altough said cooperation is *indirect*. Notably, such agents could:

- provide deeper insight into strategies they employ, in opposition to usual black box agents.
- generate realistic, human-like playing data in an automated way.
- offer a more interesting, organic and fun challenge to human players.

Such agents will be referred to as **gatherers of experience** rather than the usual black box superhuman-level opponent game playing agents trend towards.

As stated in previous chapters, our main framework for humane agents is that of the Digital Ludeme Project (DLP), more specifically embedding said models in the Ludii game playing system. In this frame, human-like models would be used for the reconstruction of ancient games with missing rules, pieces or records of trials, as well as generating plausible playing data and strategies that we can analyze to better understand the cultures said games originated in.

This chapter focuses on translating methods, approaches and results summarized in chapter 3 to the specific field of general game playing. We will dive into the details of a preliminary model trying to mimick human cognition and behavior, as well as the experimental setup devised to evaluate the humaneness of said model when it plays (Figure 4.1).

This chapter is dedicated to further detailing the ideas behind its architecture, evaluation scheme and how adjustments should be made based on feedback from said evaluations.

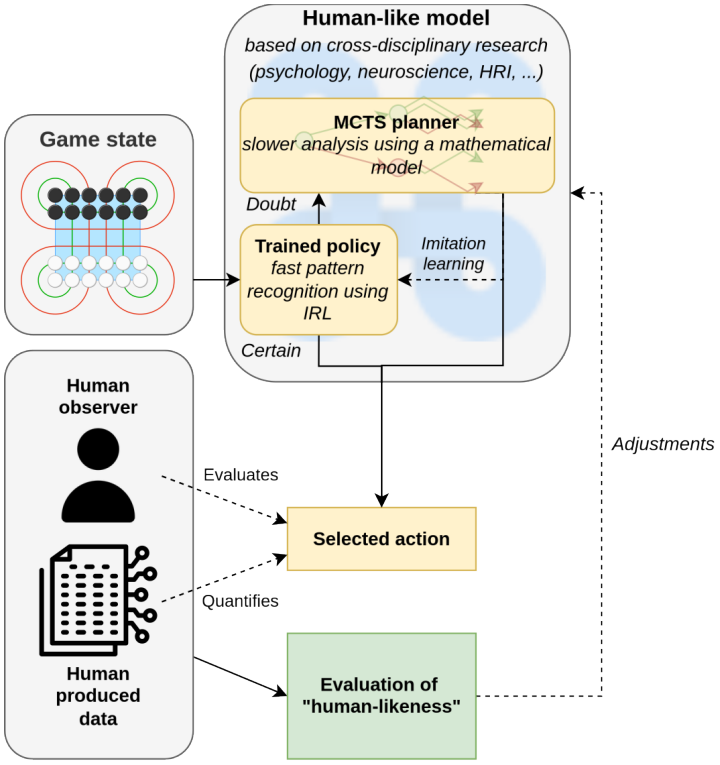

Figure 4.1: A preliminary experimental setup involving an AI model inspired by the dual-process theory of cognition [60]. This model integrates a rapid pattern recognition system developed through Inverse Reinforcement Learning (IRL) and based on state-action features (System 1) that defaults to a Monte Carlo Tree Search model (System 2) when it encounters states of uncertainty regarding the optimal action. The outputs of this model are assessed both qualitatively by humans through a Turing test and quantitatively by comparing them against statistical player profiles. This two-pronged evaluation method aims to gauge the model's "human-likeness" effectively, ensuring a robust measure of its performance in scenarios mimicking human decision-making.

4.1 Architecture

The architecture of our first model is based off of a few core ideas unveiled in the previous chapter:

- double-process theory of cognition
- inverse reinforcement learning for human-like rewards
- selection of a subset of moves to search based on prior knowledge
- best-first search model of planning

Our architecture is composed of two core blocks: the **intuitive brain** (based on System 1 of the Double-Process Theory of Cognition aka DPTC) and the **analytical brain** (based on System 2 of DPTC). Both blocks will work in tandem and exchange information to direct the final decision made by the model as well as future decisions.

4.1.1 The intuitive brain

Humans rely much more on intuition and pattern recognition than searching deep lines when playing games, as De Groot uncovered during his studies [49]. The trend for game-playing agents, however, focuses more on the latter: the deeper and longer an α-β or MCTS algorithm can search, the better it performs on average. This is mainly due to our inneficiency in capturing human players' pattern matching capabilities. Even modern approaches based on deep neural networks tend to fall for Binet's theory that the model of cognition for human players evaluating positions relates to visual recognition [64] [7].

Humans tend to prune extremely large portions of the search tree, focusing on a few identified good actions. This leads to relatively shallow, sharp lines that are informed and guided purely through intuition. The difference between an expert and an amateur rarely lies in the depth to which they can simulate future moves rather than their ability to identify good lines intuitively.

We represent this cognitive feature of humans as **the intuitive block**, corresponding to System 1 of the DPTC. This system focuses on intuitive, memory based thinking. As to clarify, this is not akin to *visual memory*, as Alfred Binet theorized prior to De Groot's thesis, but rather an abstract representation of positions based on spatial relationships between pieces. We also tend to not think about the intrinsic value of position but rather the value of actions that can be made in the current state.

We posit that state-action features as described by Dennis Soemers in his thesis [43] and implemented in the Ludii game playing system would be prime candidates for this type of heuristic. They represent abstract spatial relationships between pieces based on the rules of the current game. As such, they fit the criterion of:

- **explainability**, as learned features can be easily extracted from the model.
- **adaptability**, since they can be learnt on the fly using the process of expert iteration. The expert system is here provided by the analytical brain, further explained in 4.1.2.
- being based on **pattern matching** and recognizing previously encountered relations between pieces.

- providing a **policy over legal actions**.

The resulting policy is then first observed to decide whether the agent is in a state of **doubt** or not. Intuitively, if doubt arises in the intuitive brain, it falls back to the secondary analytical block to inform its final decision. A question of importance here is that of deciding what it means for the agent to be in doubt.

When introducing the DPTC model for their human-like conversational agent, He et al. described the state of doubt "by the probability difference for the top-2 values" [12]. Put simply, if the top-2 choices as ranked by the policy are close enough e.g. top-1 has a value of 100 while top-2 has a value of 98, then the agent is in doubt and falls back to the analytical brain to search the full range of possible actions. Our definition of "doubt" will be more closely tied to the results obtained by De Groot that humans search a few good moves. Our idea is to use the policy as a *filter of potentially good actions* by picking the best action until the entropy of the policy is deemed too high i.e. induce a state of doubt by reducing the action space. This is done by computing the entropy of the policy, then picking actions to search by removing them from the action set and normalizing the policy. This process is repeated until the policy is too uncertain in regards to an ϵ parameter.

Formalizing this idea, we compute the initial entropy of the policy distribution as such:

$$H(P_A) = -\sum_{a \in A} p_a log_2 p_a$$

Where P_A is the policy over a set of actions A and p_a is the probability that the agent will choose action $a \in A$.

Our set of actions to be searched can then be computed using the following algorithm (Algorithm 1), where ϵ is a threshold value for which we consider the policy to be uncertain:

This algorithm filters actions that are deemed not worthy of further search by the policy function. If $|A'| = 1$, we can skip the second system altogether since the policy was absolutely certain that this action was the right one. On the other hand, if $|A'| = 0$ then the policy was completely unsure about the right course of action, and we fallback to our slow analytical system entirely to decide what action is best.

4.1.2 The analytical brain

The second block of our model, or **analytical brain**, will be based on the Monte-Carlo Tree Search (MCTS) algorithm to further plan ahead. The choice for this algorithm instead of alternatives like α-β or Proof-Number Search is due to its best-first, unbalanced nature, which has been deemed to be more closely tied to human cognition when planning in games [49] [4].

The original MCTS algorithm (Figure 2.2) works in 4 phases that constitute one iteration:

- the **selection** phase starts from the root node, then recursively selects a child node until a leaf (unexpanded) node is reached. This selection is most often done using a formula such as Upper-Confidence bound applied to Tree (UCT) to balance exploration and exploitation.

Algorithm 1 Pseudo-code of an algorithm to sample a subset of actions to search deeper. It does so by picking out the best action until the subset of non-picked actions reaches a high enough entropy, determined by an ϵ parameter. This cuts the legal actions space in two: a subset of "bad" actions, and a subset of "good" actions, both with high entropy. In the event that all actions are equally good from the start i.e the "good" subset is empty, the whole legal actions space is returned.

$P_A \leftarrow$ policy over$A$
$A' \leftarrow \{\}$
$H \leftarrow H(P_A)$
if $2^{|A|} - \epsilon \leq H$ or $|A| = 0$ **then**
    return A
end if
repeat
    $a^* \leftarrow \arg\max p_a$
    $A' \leftarrow A' \cup \{a^*\}$
    $A \leftarrow A \setminus \{a^*\}$
    $P_A \leftarrow \{\frac{p_a}{1-p_{a^*}}, a \in A\}$
    $H \leftarrow H(P_A)$
until $\frac{2^{|A|}}{H} \leq \epsilon$ or $|A| = 0$
return A'

- the selected node is then **expanded**, aka one of its children is added to the tree.
- a **simulation** or **rollout** is conducted starting from the new leaf node. This amounts to sampling a trial starting from this node to obtain a result.
- the sampled result is then **backpropagated** up the tree, updating the values for nodes visited during the selection phase.

There are a few modifications made to this base algorithm for it to better fit the human psyche:

- the **simulation** phase is discarded, as it makes no sense for a human player to "randomly sample" a possible trial from a position. The expanded leaf node will instead have the value of its best possible child according to the policy (intuitive brain).
- children of a state will not encompass its **entire action space**, but rather the filtered actions from the intuitive brain. This better fits the results showing that humans only look at a subset of actions they deem "good" from experience, discarding the rest of the action space.
- the search tree **cannot grow unbounded**. Even when given ample time to think, humans keep a cognitive limit to how much they are able to remember about their search. We focus on refining lines that we have deemed exploitable after analysis. A hard limit on the number of nodes stored is thus placed.

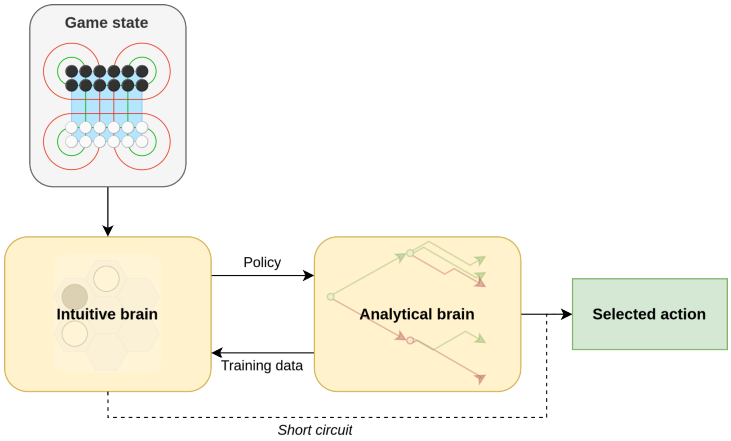

Figure 4.2: Co-dependency between the two systems of DPTC is illustrated as a back-and-forth, the intuitive brain guiding the analytical MCTS algorithm during its iterations by reducing the search space drastically, while the results of this deeper analysis are learnt from to sharpen the intuition of the model.

- it is possible, though not backed up by previous results, to incur **noise** during back-propagation and/or selection.

This scheme puts the spotlight on the intuitive brain, as it highly biases the search algorithm. It entirely decides of the values of leaf nodes when expanded and dictates the action space to be searched.

The hard-limit that is put on the number of nodes that can be stored in the game tree exists to avoid the unrealistic scenario of a human searching to unimaginable depths, which is likely to occur due to the reduced search space. We propose to use a fixed size table mapping states to nodes, with a replacement strategy in the event of a collision. Such replacements strategies could favor nodes with:

- **high exploitation values** i.e. nodes that are known to be good.
- **high amount of visits** as they are likely important to the search.

Many other such replacement schemes exist and will be tested for experimental results. In the same vein, the actual size limit will have to be determined experimentally, as no results from various fields of cognitive science are able to provide an upper bound on the matter.

4.1.3 Learning and interaction between the two brains

These two blocks should be co-dependant (Figure 4.2), as is the case in DPTC. As already stated, the analytical brain is greatly biased by the intuitive one due to the reduction of search space and influence on the values of leaf nodes.

While the analytical brain is purely based off of the rules of the game, and can thus function without any preparation, the intuitive brain is based solely on **recognition of learnt patterns**. As such, it requires a learning phase, which can be separated in two paradigms:

- the agent should learn **during the game**, reusing results from its analysis to adjust its playing style. If an action that was intuitively good turns out to lead to a losing position after further analysis, the agent should be able to learn from that result.

- after a definitive result has been reached i.e. at the **end of a trial**, the agent should be able to *reflect* on this outcome and adjust its beliefs accordingly.

Online learning
During a trial, the agent should be able to learn online from the insightful results of the MCTS algorithm. Once the search has finished, two scenarios can occur: either the analytical brain **agrees with the policy**, or it produces a **different policy**. In any case, the results of the analytical brain **should be used to bias intuition**.

This is akin to an **expert iteration** system, already used to learn state-action features from game play in the Ludii game playing system. In such systems, and *apprentice* (usually used to produce a policy) learns from the results of an *expert* (Monte-Carlo Tree Search, α-β, and other search algorithms) in order to infer better heuristics to guide the expert. In our model, the expert would be represented by the analytical brain while the intuitive brain acts as the apprentice.

Offline learning
When starting to learn a game, the intuitive brain is completely blank and unable to give insight to direct the analysis. In this case, it is in a constant state of doubt regarding the policy, and relies entirely on the MCTS algorithm to analyze the game. This is akin to a child trying out a game for the first time: it has no preconcieved ideas about which actions are good in the long term, and requires deeper analysis of the game in order to make any sort of plans.

These plans are obviously subpar, or even random, due to the fact that all states feel the same. Planning ahead without any intuition on the long-term value of actions is thus futile at this point. Only at the end of the game can the intuitive brain begin to analyze what happened and infer some first preconceptions about the value of patterns. A step of learning has to happen **in between games** in order for the intuitive brain to learn correctly. This approach is more akin to **reinforcement learning**. After a playout has been done, we give the agent time to reflect on the outcome by assigning game-theoric values from the game's result to all actions that were taken.

The aim is twofold:

- we avoid a scenario where the agent's learning curve is slow at the start of training due to non-existent heuristics and the removal of the simulation phase of the MCTS algorithm.

- we allow the agent to infer *long-term rewards* rather than forcing it to rely only on the relatively short-sighted analytical brain.

In order to account for that, the ϵ parameter controlling whether the model is in a state of doubt regarding actions could evolve as the model learns to represent its confidence in its intuitive decisions. A model with no experience would use a high value, being

more doubtful about its insights. As it gathers more experience and validates its intuition through analysis more and more accurately this ϵ value would reduce.

TODO: insert figure here about this process. CAPTION: **Left:** the intuitive brain gathers experience **online** by comparing its output policy with the analytical brain's policy after a searching deeper into the game tree, similar to expert iteration. The ϵ parameter controlling doubt is adjusted based on the difference between the policies (gets lower if the policies are close, increases otherwise). **Right:** the phase of *reflection* after a trial ends is more akin to traditional reinforcement learning. Chosen actions are marked with the game theoric value of the trial and the intuitive brain adjusts its knowledge based on this game theoric value.

4.2 Human-like rewards for tabletop games

Besides the planning algorithms used to mimick human though processes, an important aspect is the modelling of *human affect* in games. This is reflected in finding an appropriate **reward function**.

Our first model will use Inverse Reinforcement Learning (IRL), which we presented formally in 3.2.5, to study how humans perceive rewards in games. For example, two-player zero-sum games often consider reward functions to be −1 for the losing player, 1 for the winner, and 0 for both in case of a draw. It would not be far-fetched however to infer that so called "sore loosers" would value losing and winning in an unbalanced way, as would a novice playing against an expert. The former could value losing states with −10 while winning states are still valued 1 and draws hold a reward of 0. For the latter, maybe winning states would be valued extremely highly (30), draws are still thought of as a win (1) since the opponent was supposed to dominate us, and losses are valued 0, since they were just expected.

Many online plateforms for tabletop games make their databases of recorded trials available, some holding millions of games that can be used to infer a reward function.

4.2.1 Clustering player behavior

As stated multiple times throughout this report, human motivations are *varied*. Some players are aggressive, others are tactical, some hate loosing, others try to play tricky positions hoping to make their opponent fall for a trap.

IRL problems would benefit from learning rewards from *clusters of players* rather than the entire playerbase at once. Clustering players by their behavior would help the model learn multiple sharp reward functions rather than one vague "average of human behavior" way of valuing game states.

4.3 Evaluation scheme

Our evaluation scheme to assess the human-likeness of our model is two-fold:

- **move matching** will be used to compare its decisions to that of humans in the same environment or state.
- **Turing tests** are organized for qualitative feedback on the humanness of the model, as well as more intricate attributes relating to its play style.

This evaluation will also be held on other models to serve as comparison. Notably, we'll conduct the same experiments on a **pure MCTS-UCT model** in Ludii, which also uses state-action features, as well as **other well-known game playing agents**.

4.3.1 Move matching

As described in chapter 3, move matching is a statistical test assessing how well an artificial model is able to predict human moves given a state [4].

In this evaluation scheme, skill levels must be taken into account. One could argue that a game played between two skilled human players is different than one played between two novices. As such, states and their according action distribution must be classified by both game and skill level when available. This greatly reduces the amount of games that move matching can be conducted on as we need both a large enough dataset and access to some form of skill rating.

Multiple online board game platforms offer such data for popular games. One example is the Lichess database, which features a wide range of games of various skill levels. Similar databases for games such as Go also exist.

A noteworthy element is that move matching only cares about **the actually chosen action**. The ranking of other actions is never taken into account. Our evaluation scheme would first create statistics on chosen moves based on games between humans. The chosen move of an agent would then determine its statistical success based on the probability of this move being chosen by a human player. For example, if in the starting position of the game of chess, 60% of human players chose to play e4, and the agent also chooses e4, then it has 60% accuracy in the move matching test.

4.3.2 Turing tests

These tests will serve as the basis for more nuanced qualitative feedback on the humanness of our model. It will deviate from the original and well known method of ranking observees by instead asking observers to rate the human-likeness of each opponent individually. On top of this general observation, multiple scales for play style related attributes will be added in order to better understand "how the model plays".

These tests can be conducted directly on the Ludii online platform, which offers to play implemented games against AI opponents. A form would pop-up before the game starts offering the visitor to participate in the study. Participation would obfuscate information about the available AI opponents as to avoid prior knowledge, as well as offer a form to fill information about the perceived humanness of the opponent after the game. Additionally, information about the game such as moves played, states reached, and evaluation information from the artificial agent would be recorded.

Usually, these kinds of tests include a human participant in order to form a statistical baseline of the perception of observers. While our methodology allows for a large number of participants, it lacks this important aspect. As such, more formal studies should be conducted by gathering participants and randomly assigning them to either play against each other or AI opponents. This primary dataset would form a more trusted source of information, albeit on a smaller population.

4.4 Malleability of the model

There are multiple parameters that can be adjusted depending on the feedback obtained through our evaluation scheme:

- the ϵ parameter deciding on whether the intuitive brain is in **doubt**.
- the maximum size of the remembered game tree.
- the allowed complexity of state-action features.

This gives us some leverage to test various sets of parameters in order to adjust our current model. However, would the model prove to not feel human-like after said adjustments have been exhausted, it would have to be redesigned.

These parameters also give us a way to adjust the model's perceived strength in an organic way. Online platforms featuring AI opponents often limit their strength based on **depth-limit** i.e. limiting the maximum search depth the program is allowed to reach, or though **random blunders** i.e. assigning a probability of picking an action the program has deemed sub-optimal.

While this indeed reduces the playing strength of these programs, it does so in an *artificial way*. The resulting agents often play *"weird"* actions that don't make much more sense from a human perspective, or resort to straight up nonsensical play. A good example of the latter is the lowest level chess bots available on many online platforms.

With our model, reducing the playing strength of an agent can come from:

- reducing the size of the remembered game tree, making it less capable of analyzing the position deeper while not hindering its ability to play intuitively good moves.
- reducing the complexity of learnt state-action features, providing a ceiling to how much the model can hone its intuition.

The latter matches what has been observed by De Groot or Chase et al. that lower ranked players often don't differ much in their ability to analyze a position rather than in their ability to intuitively identify good moves [49] [51].

5

Conclusion

Current state-of-the-art for human-like agents in multiple fields of computer science and robotics showcase promising results in bridging the gap in behavior between AI models and humans, cementing our hopes of creating a similarily performing agent for General Game Playing. Notably, they bring forward a standard definition of the problem, as well as various methods of evaluation, both of which were found to be complex questions spanning multiple subfields of cognitive science.

A human-like model for GGP derived from these methods and approaches in Section 4 will be implemented and evaluated over multiple levels of granularity. First, we aim to implement a simple game-specific agent using this model for the game of Renju. Due to their random sampling model, standard Monte-Carlo Tree Search (MCTS) approaches tend struggle with ”trap states”—situations with only one winning or losing move—which are inherent to the game. In contrast, humans excel at recognizing important patterns and have a strong focus on them to guide their planning, a behavior which we aim to replicate in this restricted framework. This game-specific model will be evaluated using both methods described in this report, Turing tests and move matching, in order to gather various forms of feedback to further improve the model.

Our model, improved through collected feedback, will later be generalized for General Game Playing in the Ludii game playing system.

Bibliography

URLs in this thesis have been archived on Archive.org. Their link target in digital editions refers to this timestamped version.

References

[1] Jacques Pitrat. Realization of a general game-playing program. In *IFIP Congress (2)*, pages 1570–1574, 1968.

[2] Yngvi Bj"ornsson and Stephan Schiffel. General game playing. In *Handbook of Digital Games and Entertainment Technologies*, pages 1–23, Singapore, 2016. Springer Singapore.

[3] Walter Crist, Alex de Voogt, and Anne-Elizabeth Dunn-Vaturi. Facilitating interaction: Board games as social lubricants in the ancient near east. *Oxford Journal of Archaeology*, 35:179–196, 05 2016.

[4] Reid McIlroy-Young, Siddhartha Sen, Jon Kleinberg, and Ashton Anderson. Aligning superhuman ai with human behavior: Chess as a model system. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery amp; Data Mining*, KDD '20. ACM, August 2020.

[5] J. Schaeffer. *One Jump Ahead: Challenging Human Supremacy in Checkers*. Copernicus Series. Springer, 1997.

[6] Murray Campbell, A.Joseph Hoane, and Feng hsiung Hsu. Deep blue. *Artificial Intelligence*, 134(1):57–83, 2002.

[7] David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484–489, Jan 2016.

[8] Barney Pell. Metagame: A new challenge for games and learning. 1992.

[9] Michael Genesereth, Nathaniel Love, and Barney Pell. General game playing: Overview of the aaai competition. *AI magazine*, 26(2):62–62, 2005.

[10] Andrzej Ehrenfeucht. An application of games to the completeness problem for formalized theories. *Fund. Math*, 49(129-141):13, 1961.

[11] Patrick Bryant, Gabriele Pozzati, Wensi Zhu, Aditi Shenoy, Petras Kundrotas, and Arne Elofsson. Predicting the structure of large protein complexes using alphafold and monte carlo tree search. *Nature Communications*, 13(1):6028, Oct 2022.

[12] Tao He, Lizi Liao, Yixin Cao, Yuanxing Liu, Ming Liu, Zerui Chen, and Bing Qin. Planning like human: A dual-process framework for dialogue planning. 2024.

[13] Stephanie Milani, Arthur Juliani, Ida Momennejad, Raluca Georgescu, Jaroslaw Rzepecki, Alison Shaw, Gavin Costello, Fei Fang, Sam Devlin, and Katja Hofmann. Navigates like me: Understanding how people evaluate human-like ai in video games. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*, CHI '23. ACM, April 2023.

[14] Micah Carroll, Rohin Shah, Mark K Ho, Tom Griffiths, Sanjit Seshia, Pieter Abbeel, and Anca Dragan. On the utility of learning about humans for human-ai coordination. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d 'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[15] Annemarie Turnwald and Dirk Wollherr. Human-like motion planning based on game theoretic decision making. *International Journal of Social Robotics*, 11(1):151–170, Jan 2019.

[16] Rachid Alami, Aurélie Clodic, Vincent Montreuil, Emrah Akin Sisbot, and Raja Chatila. Task planning for human-robot interaction. In *Proceedings of the 2005 Joint Conference on Smart Objects and Ambient Intelligence: Innovative Context-Aware Services: Usages and Technologies*, sOc-EUSAI '05, page 81–85, New York, NY, USA, 2005. Association for Computing Machinery.

[17] Aaron Tucker, Adam Gleave, and Stuart Russell. Inverse reinforcement learning for video games. *CoRR*, abs/1810.10593, 2018.

[18] Bernard Gorman and Mark Humphrys. Towards integrated imitation of strategic planning and motion modeling in interactive computer games. *Comput. Entertain.*, 4(4):10–es, oct 2006.

[19] Ben George Weber, Michael Mateas, and A. Jhala. Building human-level ai for real-time strategy games. In *AAAI Fall Symposium: Advances in Cognitive Systems*, 2011.

[20] Jacek Mańdziuk. Towards cognitively plausible game playing systems. *IEEE Computational Intelligence Magazine*, 6(2):38–51, May 2011.

[21] Jacek Mańdziuk. Human-like intuitive playing in board games. In Tingwen Huang, Zhigang Zeng, Chuandong Li, and Chi Sing Leung, editors, *Neural Information Processing*, pages 282–289, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[22] Cameron Browne. Modern techniques for ancient games. 2020.

[23] Éric Piette. Computational techniques for tabletop games heritage, Oct 2023.

[24] Éric Piette, Dennis J. N. J. Soemers, Matthew Stephenson, Chiara F. Sironi, Mark H. M. Winands, and Cameron Browne. Ludii – the ludemic general game system. 2020.

[25] Cameron Browne, Dennis J. N. J. Soemers, Éric Piette, Matthew Stephenson, Michael Conrad, Walter Crist, Thierry Depaulis, Eddie Duggan, Fred Horn, Steven Kelk, Simon M. Lucas, João Pedro Neto, David Parlett, Abdallah Saffidine, Ulrich Schädler, Jorge Nuno Silva, Alex de Voogt, and Mark H. M. Winands. Foundations of digital archæoludology. 2019.

[26] RICHARD BELLMAN. A markovian decision process. *Journal of Mathematics and Mechanics*, 6(5):679–684, 1957.

[27] L. S. Shapley. Stochastic games. *Proceedings of the National Academy of Sciences*, 39(10):1095–1100, 1953.

[28] Sergiu Hart. Chapter 2 games in extensive and strategic forms. volume 1 of *Handbook of Game Theory with Economic Applications*, pages 19–40. Elsevier, 1992.

[29] Cameron B. Browne, Edward Powley, Daniel Whitehouse, Simon M. Lucas, Peter I. Cowling, Philipp Rohlfshagen, Stephen Tavener, Diego Perez, Spyridon Samothrakis, and Simon Colton. A survey of monte carlo tree search methods. *IEEE Transactions on Computational Intelligence and AI in Games*, 4(1):1–43, 2012.

[30] Leslie Pack Kaelbling, Michael L. Littman, and Andrew W. Moore. Reinforcement learning: a survey. *J. Artif. Int. Res.*, 4(1):237–285, may 1996.

[31] Thomas McGrath, Andrei Kapishnikov, Nenad Tomašev, Adam Pearce, Martin Wattenberg, Demis Hassabis, Been Kim, Ulrich Paquet, and Vladimir Kramnik. Acquisition of chess knowledge in alphazero. *Proceedings of the National Academy of Sciences*, 119(47):e2206625119, 2022.

[32] Stuart Russell. Learning agents for uncertain environments (extended abstract). In *Proceedings of the Eleventh Annual Conference on Computational Learning Theory*, COLT' 98, page 101–103, New York, NY, USA, 1998. Association for Computing Machinery.

[33] Donald E. Knuth and Ronald W. Moore. An analysis of alpha-beta pruning. *Artificial Intelligence*, 6(4):293–326, 1975.

[34] Levente Kocsis and Csaba Szepesvári. Bandit based monte-carlo planning. In Johannes Fürnkranz, Tobias Scheffer, and Myra Spiliopoulou, editors, *Machine Learning: ECML 2006*, pages 282–293, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.

[35] Tristan Cazenave. Generalized rapid action value estimation. In *Proceedings of the 24th International Conference on Artificial Intelligence*, IJCAI'15, page 754–760. AAAI Press, 2015.

[36] Sylvain Gelly and David Silver. Monte-carlo tree search and rapid action value estimation in computer go. *Artificial Intelligence*, 175(11):1856–1875, 2011.

[37] R.S. Sutton and A.G. Barto. *Reinforcement Learning: An Introduction*. Adaptive Computation and Machine Learning series. MIT Press, 1998.

[38] C. Browne. *Connection Games: Variations on a Theme*. Ak Peters Series. Taylor & Francis, 2005.

[39] Cameron Bolitho Browne. *Automatic generation and evaluation of recombination games*. PhD thesis, Queensland University of Technology, 2008.

[40] Cameron Browne. *A Class Grammar for General Games*, pages 167–182. Lecture Notes in Computer Science. Springer, Cham, Switzerland, 2016. DBLP's bibliographic metadata records provided through http://dblp.org/search/publ/api are distributed under a Creative Commons CC0 1.0 Universal Public Domain Dedication. Although the bibliographic metadata records are provided consistent with CC0 1.0 Dedication, the content described by the metadata records is not. Content may be subject to copyright, rights of privacy, rights of publicity and other restrictions.

[41] Cameron Browne, Dennis Soemers, Eric Piette, Matthew Stephenson, and Walter Crist III. *Ludii Language Reference*. December 2020.

[42] Dennis J. N. J. Soemers, Éric Piette, Matthew Stephenson, and Cameron Browne. The ludii game description language is universal, 2024.

[43] Dennis J.N.J. Soemers. *Learning state-action features for general game playing*. PhD thesis, Maastricht University, 2023.

[44] Dennis J. N. J. Soemers, Éric Piette, and Cameron Browne. Biasing MCTS with features for general games. *CoRR*, abs/1903.08942, 2019.

[45] Dennis J.N.J. Soemers, Spyridon Samothrakis, Éric Piette, and Matthew Stephenson. Extracting tactics learned from self-play in general games. *Information Sciences*, 624:277–298, 2023.

[46] Thomas Anthony, Zheng Tian, and David Barber. Thinking fast and slow with deep learning and tree search. *CoRR*, abs/1705.08439, 2017.

[47] Rodrigo Canaan, Christoph Salge, Julian Togelius, and Andy Nealen. Leveling the playing field – fairness in ai versus human game benchmarks. 2019.

[48] Robert Levinson and Richard Snyder. Adaptive pattern-oriented chess. In Lawrence A. Birnbaum and Gregg C. Collins, editors, *Machine Learning Proceedings 1991*, pages 85–89. Morgan Kaufmann, San Francisco (CA), 1991.

[49] Adriaan D De Groot and Adrianus Dingeman De Groot. *Thought and choice in chess*, volume 4. Walter de Gruyter, 1978.

[50] What is the average number of legal moves per turn? 2018.

[51] William G. Chase and Herbert A. Simon. Perception in chess. *Cognitive Psychology*, 4(1):55–81, 1973.

[52] Robert M. French. Subcognition and the limits of the turing test. *Mind*, 99(393):53–65, 1990.

[53] Rohan Choudhury, Gokul Swamy, Dylan Hadfield-Menell, and Anca D. Dragan. On the utility of model learning in HRI. *CoRR*, abs/1901.01291, 2019.

[54] Alison Gopnik and Henry M. Wellman. *The theory theory.*, pages 257–293. Mapping the mind: Domain specificity in cognition and culture. Cambridge University Press, New York, NY, US, 1994.

[55] Chris L. Baker, Rebecca Saxe, and Joshua B. Tenenbaum. Action understanding as inverse planning. *Cognition*, 113(3):329–349, 2009. Reinforcement learning and higher cognition.

[56] Beate Sodian, Barbara Schoeppner, and Ulrike Metz. Do infants apply the principle of rational action to human agents? *Infant Behavior & Development*, 27(1):31–41, 2004.

[57] György Gergely and Gergely Csibra. Teleological reasoning in infancy: the naïve theory of rational action. *Trends in Cognitive Sciences*, 7(7):287–292, 2003.

[58] *Theories of Theories of Mind*. Cambridge University Press, 1996.

[59] Claude E Shannon. Xxii. programming a computer for playing chess. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 41(314):256–275, 1950.

[60] Daniel Kahneman. A perspective on judgment and choice: mapping bounded rationality. *Am Psychol*, 58(9):697–720, September 2003.

[61] Saurabh Arora and Prashant Doshi. A survey of inverse reinforcement learning: Challenges, methods and progress. *Artificial Intelligence*, 297:103500, 2021.

[62] Aristide CY Tossou and Christos Dimitrakakis. Probabilistic inverse reinforcement learning in unknown environments. *arXiv preprint arXiv:1307.3785*, 2013.

[63] Tomer D. Ullman, Chris L. Baker, Owen Macindoe, Owain Evans, Noah D. Goodman, and Joshua B. Tenenbaum. Help or hinder: Bayesian models of social goal inference. In *Proceedings of the 22nd International Conference on Neural Information Processing Systems*, NIPS'09, page 1874–1882, Red Hook, NY, USA, 2009. Curran Associates Inc.

[64] Alfred Binet. Psychologie des grands calculateurs et joueurs d'échecs (book review). *The Monist*, 5:142, 1894.

UNIVERSITÉ CATHOLIQUE DE LOUVAIN
École polytechnique de Louvain
Rue Archimède, 1 bte L6.11.01, 1348 Louvain-la-Neuve, Belgique | **www.uclouvain.be/epl**