Machine Learning in a Social Deduction Game

Francisco Aristi Reina
London School of Economics
London, United Kingdom
F.Aristi-Reina@lse.ac.uk

Steffen Issleib
London School of Economics
London, United Kingdom
S.Issleib@lse.ac.uk

Bernhard von Stengel
London School of Economics
London, United Kingdom
B.von-Stengel@lse.ac.uk

ABSTRACT

We let various AI agents play "Temple of Terror", a known social deduction game between two teams. Success is measured by winning or losing, which should lead to benchmarks for learning good strategies, and allow the application of game-theoretic concepts about team games. Players have hidden roles, declare in rounds their cards but may lie, and select the next player to uncover a card. This results in quick play and actions counting more than words. The agents studied are: An ESCHER-based triple of neural networks to learn behavior strategies with counterfactual regret, and prompt-engineered LLMs with different reasoning capabilities.

CCS CONCEPTS

• **Computing methodologies** → **Multi-agent systems**.

ACM Reference Format:
Francisco Aristi Reina, Steffen Issleib, and Bernhard von Stengel. 2026. Machine Learning in a Social Deduction Game. In *Proc. of the Adaptive and Learning Agents Workshop (ALA 2026), Paphos, Cyprus, https://alaworkshop2026.github.io/, May 25–26, 2026*, IFAAMAS, 16 pages.

1 INTRODUCTION

Our general research question is how multi-agent learning can be *validated* for success, using a suitable game-theoretic scenario. AI agents have been very good at competitive two-player games like Chess, Go, Stratego, and multi-player Poker [4]. Their success is evidenced in beating world-class human players, or in machine versus machine play.

In an economic game, success is harder to identify. For example, trade has win-win aspects where other players can be partners as well as competitors. Nash equilibrium is an interdependent concept of success, but may be far from socially optimal. Socially superior cooperation that emerges as the result of some learning dynamics may also be viewed positively (as in repeated Prisoner's Dilemma games) – or, at yet another level, negatively for consumers if it represents algorithmic collusion among pricing agents [5].

In this paper, we study games that have clear competitive elements and therefore measures of being successful, but also cooperative aspects because they involve teams.

The game we consider is a version of an existing *social deduction game* (see Section 1.2). Players belong to one of two opposing teams in a randomly assigned *role* at the beginning of the game that is only known to them privately. In the course of the game, they pursue actions to move their own team to victory. However, in initial phases of the game it may be useful to deceive other players about one's role in order to gain trust that one later exploits.

Proc. of the Adaptive and Learning Agents Workshop (ALA 2026), Aydeniz, Delgrange, Mohammedalamen, Yang (eds.), May 25–26, 2026, Paphos, Cyprus, https://alaworkshop2026.github.io/. 2026.

1.1 Related work

Games have provided important benchmarks for validating the capabilities of AI agents. In addition, these algorithms have been used to approximate game-theoretic solutions like Nash equilibria that are computationally intractable for larger games.

Learning algorithms to play competitively perfect-information zero-sum games like Go [25], Chess, and Backgammon have been trained using self-play reinforcement learning. Subsequently, research works showed how to achieve high competitive play in two-player zero-sum imperfect-information games like Leduc Poker, with game theoretic algorithms such as counterfactual regret minimization [33], and later on larger games such as Stratego [21], where state of the art capability was achieved using reinforcement learning and other no-regret methods. For all these zero-sum games with two players, learning algorithms have been proved to converge or be close to the max-min equilibrium.

Additionally, different research focused on competitive play involving multiplayer games like Poker[4], large games like Starcraft [30], and games with natural language, for example Diplomacy [13]. These achieved state of the art capability with the Outcome Sampling Monte Carlo Couterfactual Regret Minimization algorithm [16], PSRO, and supervised learning combined with self-play reinforcement learning, respectively. This class of multiplayer games is not evaluated by comparing a policy to an exact Nash equilibrium solution. Rather, experimental results against professional players and the emergence of patterns of play are the appropriate evaluation criteria.

Further on, AI agents have achieved state of the art play in coordinated team games like football, for example TEAM PSRO [19], where the appropriate game-theoretic solution and convergence was the concept of correlated team-maxmin equilibrium [7].

Social deduction games are team games where some agents need to learn the roles of other players to coordinate and win. AI agents that can play competitively one-sided imperfect information social deduction team games [6] included LLM solutions and MARL. The considered game-theoretic solution for evaluation is the team-maxmin equilibrium, an independent maxmin strategy from each member of the team that obtains the maximum payoff in equilibrium against a single adversary [32], and, with a communication phase, the related solution concept of communication team-maxmin equilibrium [7].

Providing an algorithmic competitive play for imperfect information with unknown roles for all players and two teams with communication remains an open problem in the Machine Learning and Game Theory literature. Typically, no theoretical Nash equilibrium solution to compare against has been found in general, and therefore no convergence analysis given, nor is this provided in the present paper.

1.2 Game rules

Social deduction games are parlour games where players have hidden *roles* that are uncovered in the course of play. At the beginning of the game, each player is randomly assigned their role as belonging to one of two opposing teams. Usually one of the teams is smaller, and its members profit from deceiving the other players about their role to create trust that they later exploit.

In the archetypal social deduction game *Mafia* or Werewolf [9], play alternates between the "mafiosi" who "at nighttime" decide to kill one of the "citizens", followed by a "daytime" discussion between all surviving players to execute a suspected mafioso (who may however be a citizen), until only one team successfully remains. The game has been adapted in the popular reality television show "The Traitors" in Britain and the US [3].

The game *Resistance* (and similarly *Avalon* [6, 29]) avoids the elimination of players by letting all players discuss and agree on "missions" of sets of players that can "succeed" or "fail". A single member of the "bad" team in a mission can cause the mission to fail (but can also choose not to in order to hide their role). The game is designed so that often the fifth and final mission is decisive, which needs all members of the "good" team in it to succeed.

In addition to strategic thinking, Mafia and Resistance involve talk, persuasion, deception, and acting skills. A faster and less talk-based card game is *Temple of Terror (ToT)* (our translation of the German game title "Tempel des Schreckens" [20]). In this paper, we study the game-theoretic aspects of ToT, and how AI agents can learn to play it well.

We call the two teams of ToT *attackers* and *defenders* (in the game's narrative: male "gold hunters" versus female "amazons" who defend their "temple"). The attackers are the larger team and aim to uncover all *gold* cards during the game, whereas the defenders aim to uncover all *fire* cards (which are "traps").

At the start of play, roles are assigned randomly and privately to each player. For 6 players, there are 4 attackers and 2 defenders, and the game has 30 cards: 20 *empty* cards, 8 gold cards, and 2 fire cards. The game is played in 5 rounds of successively fewer cards that are dealt to each player. In the first round, each player gets 5 cards, lays them face-down, and *declares* (not necessarily honestly) what they are: for example "2 gold, 1 fire, 2 empty cards". After these declarations at the beginning of each round, one designated starting player chooses any other player and uncovers randomly one of their cards. That player then chooses the next player, until 6 cards (the number of players) have been uncovered, after which the not yet uncovered cards are re-shuffled and dealt for the next round (that is, in round 2 each player gets 4 cards, then 3 cards, and so on). The last player in a round becomes the starting player in the next round. Uncovered gold cards accrue to the attackers, fire cards (of which there are only two) to the defenders. Whichever cards are all uncovered first determines the winning team.

In human play, players can reason and argue about the game at any time. However, this is "cheap talk" – the card declarations and whom players choose based on these declarations are more informative, and can also reveal if someone lied (e.g., declared only gold and empty cards but had a fire card, as a defender might do).

The simplicity of the actions – card declarations and choice of next player to uncover one of his or her cards – is the reason that we choose this game for our study. The game is also easily simplified to fewer players and fewer cards and rounds. We chose 4 rounds with initially 4 cards per player, one gold card per attacker, and two fire cards (a single fire card would mean that defenders win by chance alone), with the other cards empty. We considered 3, 4, 5, or 6 players with one third of them defenders (for 4 players, and in the original game rules also for 3 players, the number of defenders is randomly 1 or 2 for extra uncertainty).

1.3 Plausible strategies

A plausible good strategy for an attacker is initially to declare his cards honestly, and to choose another player with many declared gold cards. Deception seems more appropriate for defenders, who may initially aim to masquerade as attackers. One defender strategy could also be to aim to uncover one fire card quickly, and thereby quickly reveal her role, so that her team-mate (who aims at hiding her role) can choose her later at the right moment to uncover the second fire card.

Strategies should therefore depend on the game phase, which changes once a fire card has been uncovered. Initially, an attacker who has a fire card may declare this honestly, also to lure a defender to out herself by choosing him in the hope of uncovering the fire card. However, if this is the second fire card that would lose the game if uncovered, the attacker may prefer to declare only empty cards (even if he has gold cards) to avoid being chosen.

As in Mafia and Resistance, the original ToT rules state that the members of the smaller team (mafiosi, bad guys, defenders) identify each other once their roles have been assigned. From a game-theoretic viewpoint, that team could then in principle act as a single player, whose opponents (the larger team) can coordinate less and only use individually randomized strategies, for example. This is the framework of team-maxmin games in [32], as we discuss in Section 5. However, we do not let the defenders identify each other, which simplifies playing the game by using only public signals. (For human players, this also removes the need for defenders to act as if they don't know the other players' roles, because indeed they don't.)

1.4 Random play analysis

There is a simple benchmark for winning the game, which is rather unsophisticated for the attackers but of some merit (and favoring) the defenders: Completely random play of choosing the next player to uncover a card, ignoring any card declarations. Then only the number of cards matters, not the number of players: *In random play with N gold cards and 2 fire cards, the probability that the attackers win* is $2/(N+2)$. The reason is simple: Cards are uncovered in random order, where all that matters is the relative order of gold cards versus fire cards. Empty cards can be ignored as they just extend the duration of the game without altering the win/loss condition. Hence, the attackers win if and only if the last non-empty card in that order is a fire card, because then all gold cards are uncovered first. The probability for this event is $2/(N+2)$ (also confirmed by Monte Carlo simulations).

The lower number of fire cards means a lower probability of being uncovered last, which favors the defenders. While the random strategy is not particularly interesting, it provides a useful benchmark.

2 LEARNING WITH COUNTERFACTUAL REGRET

Our aim is to train AI agents to learn how to play the ToT game well. The game has a standard description as an extensive game with imperfect information [31], with the following components. A game tree has decision nodes with outgoing edges that denote a player's possible moves, chance nodes for random moves that determine a player's role or deal the cards, and terminal nodes with outcomes for all players, here that all attackers win and all defenders lose or vice versa. Every node u has a unique path from the root of the tree to u with corresponding moves, also called a *history*, or a *trajectory* (a full run of play) if u is a terminal node.

The decision nodes are partitioned into *information sets*. Each information set belongs to one player, who is not informed about where he or she is in the set, and has by definition the same move choices for each node in that set. The game has *perfect recall*, that is, the information sets reflect that players remember their own moves and previous knowledge. Perfect recall implies that it suffices for players to use *behavior strategies*, which are local randomizations over their moves at each of their information sets [15].

Counter-factual regret (CFR) [33] records a counterfactual regret for each move at an information set, rather than regrets for strategies for the entire game. This allows for efficient regret-based learning in game trees of reasonable size. Counterfactual means that the information set is not necessarily reached due to earlier actions, but rather describes a "what if?" scenario.

The ToT game is rather large, so that regret-based explorations of the entire game tree (in particular in rarely reached parts) are too slow. Outcome Sampling Monte Carlo Counterfactual Regret (OS-MCCFR) is a sampling-based algorithm that leads to much faster convergence [16], but with an estimation variance problem for very large games.

In this section, we describe a learning approach based on CFR minimization using the algorithm ESCHER (Eschewing importance Sampling by Computing a History value function to Estimate Regret) due to [18]. ESCHER has two main improvements relative to traditional OS-MCCFR, namely better learning scalability in time and memory by using approximated functions, and a reduction in the estimated variance by removing the importance-sampling component. In addition, in two-player zero-sum games the average policy is shown to converge to the Nash equilibrium.

A *cumulative reward neural network* with parameters ϕ_i represents the counterfactual regret $R_i(I_i, a \mid \phi_i)$ for the legal move a at the information set I_i of player i. The acting player i (in a one-hot encoding), information set (via its history), and action are represented as suitable neural network inputs.

Let σ be a behavior strategy. Then $Q^\sigma(I_i, a \mid I_i \prec z, \theta)$ denotes a *Q-value function neural network* with parameters θ. It represents the outcome value of a trajectory to a terminal node z that traverses the information set I_i.

Finally, $\bar{\sigma}(. \mid I_i, \rho)$ denotes the *average policy neural network* with parameters ρ over legal actions for each information set I_i. We specify the *sampling policy* by special moves $b_i(I_i)$ at I_i if it is player i's turn (see below), otherwise by the fixed moves of the other players. That is, $\bar{\sigma}$ encodes an average of behavior strategies σ, with suitable inputs encoding the player, their private information (role and cards), and public information about the game history.

The training alternates between networks. First, a learning player samples complete game plays using the fixed sampling policy, then the immediate regret is computed for the learning player using equation (5) in [18], data is collected, and the cumulative regret neural network is trained for the learning player. After a loop of learning for all players, the Q-value function neural network is trained. This procedure runs many episodes until the learning stabilizes and, in the last step, the policy neural network is trained. This is equivalent to training the average policy after some number of episodes, to see the learning evolution.

Algorithm 1: ToT-ESCHER with N players and two Teams A and D

Input: Number of outer iterations T, trajectories per update P
Output: Average policy network $\bar{\sigma}_\phi$
Method:
Initialize history Q-value function Q_θ with buffer.
Initialize policy $\bar{\sigma}_\phi$ with buffer.
Initialize cumulative regret R_{ρ_i} for every team i with buffer.
for $t = 1, \ldots, T$ **do**
  Re-initialize regret networks R_{ρ_i} for every player i.
  for *each player* $i \in N$ **do**
    Generate a set P of trajectories using the fixed sampling policy.
    for *each trajectory* τ *in* P **do**
      Estimate immediate counterfactual regret for each action of player i along trajectory τ using (5) in [18] and store the data in regret and Q-value buffers. For the rest of the players collect data for the average policy buffer.
    Train regret network R_i on cumulative regret buffer.
  Train Q-network Q_θ on the information state action buffer.
Train average policy network $\bar{\sigma}_\phi$ on average policy buffer.
return $\bar{\sigma}_\phi$

Algorithm 1 defines our ToT-ESCHER implementation with two teams and five players. We define seven fixed protocols of communication that the agents can choose from in each declaration phase of the game: truthful, hide golds, hide fires, hide one gold, hide one fire, say nothing, and credible (not obviously lying) random.

We provide a definition for the fixed sampling policy $b_i(I_i)$ that differs from the original ESCHER algorithm. We use a fixed sampling policy to induce a behavior for the attackers that maintains a positive probability for all actions. The possible policies determine which player to choose for uncovering a card. First, attackers choose 90% of the time players that only declared gold cards, with the rest of the time as random exploration. If no player declared only gold cards, they avoid players that have lied before. Otherwise, players are chosen randomly. Defenders choose players randomly with equal probability. Biasing the fixed sampling policy for the attackers achieves faster convergence, because it eliminates the exploration of losing strategies. More importantly, this bias doesn't compromise learning different policies for attackers; it is only a control on how the tree is explored. Finally, each information set is abstracted with the following state information: public information in the table (declared messages and opened cards by players),

private role, private cards, number of play in the declaration phase, score, and current round. This is an abstraction of the history of play. It doesn't record every declaration or player selection and therefore omits in which order the decisions were made. The algorithm induces attackers to always follow a truthful protocol and select gold card holders. The strategy of how to hide cards and lie is left to the defenders.

Our experimental results show that this achieves good competitive play. In the future, to have more robust policies that can play against heterogeneous players or humans, the state can be expanded to approximate histories of the current state with a sequential neural network such as LSTM, Transformer architectures, or an LLM as described in Section 3. Importantly, because of the coordination nature of the game, augmenting the state to include histories of play might still not increase competitive play, and some policies may need to be predefined. This can be seen in Section 5 where simple learning dynamics from independent players fail to play the best possible team-maxmin strategy.

Appendix C gives more details on the ESCHER algorithm.

There are no theoretical guarantees that ToT-ESCHER converges to a Nash Equilibrium. In Section 4 we discuss its learning performance and results when it plays against other algorithmic agents.

3 LLM-BASED AGENTS

Using Transformers outside of language understanding to solve complex problems via abstraction has delivered impressive results in several fields, including the ARC challenge [2, 8]. In multi-agent games with imperfect information, the ability of transformer models to abstract information based on their "language understanding" provides a promising path to dimensionality reduction that we explore.

We study three LLM-based agents. The first two are sequential reasoning agents that develop an analysis of the game state before making a decision. Agent 1 (SeMA) conducts sequential reasoning across six distinct steps, focusing on particular game-theoretic aspects to narrow down its logic. Agent 2 (SeOA) conducts this sequential reasoning in a single step to speed up inference time. We deployed SeOA as an automated agent in a web app that we wrote for human play, used during a live event at our university to showcase our research (see Appendix B). Finally, Agent 3 (GaTTA) applies unbounded, inference-time reasoning to derive complex strategies without strict token restrictions. We use Llama-Maverick and DeepSeek as the underlying models for SeMA and SeOA, and DeepSeek-R1 with thinking mode for GaTTA.

Across all experiments we always assign one agent family to an entire team of agents (attackers or defenders). Each agent in one team is an independent copy of that class of agents.

3.1 Game state representation

Representing the game state in multi-agent social deduction games is computationally challenging for traditional game theory or reinforcement learning strategies. Including the entire game history quickly creates a very large state space. Manually reducing the state space to a subset or summary of the history is "feature engineering" that imposes a strong expert prior on the learning agent and reduces the generality of the approach.

Transformer-based agents bypass this bottleneck. They can be given the entire game state history as input in natural language (or a language-based representation). This allows the LLM to leverage its transformer architecture and internal representation of language concepts to apply relevant abstractions. To investigate the inherent abstraction capabilities of LLMs, we constructed our three agents with significantly varying approaches to abstraction and reasoning.

3.2 Sequential Multi-step Agent (SeMA)

The Sequential Multi-step Agent (SeMA) is a Large Language Model agent, who is asked to make the decision for one individual player. To generate each action, SeMA calls its assigned LLM model 6 times.

Each time the model is told the player's name and role (attacker or defender) and is passed in each prompt the rules of the game and the history of the game with the card declarations and chosen players until the present state of the game.

SeMA conducts sequential analysis steps to gather relevant information and conclusions before taking an action. The following are the sequential analysis steps:

(1) **Information comprehension**: Analyze the current game state, including the player's own role and cards, as well as the current remaining cards and announcements made in this round.
(2) **Announcement analysis**: Historical announcement analysis of all players by cross-referencing player claims against revealed cards to identify inconsistencies, lies, or truthful patterns over time.
(3) **Behavior analysis**: Evaluate player choices, such as targeting choices and nomination patterns, to infer other players' hidden roles (attacker or defender) and strategies.
(4) **Probability analysis of cards**: Estimate the likelihood that specific players hold gold or fire cards based on the remaining deck composition and public information.
(5) **Probability analysis of roles**: Estimate likelihood of each player being an attacker or defender based on all players' behavior and announcement history.
(6) **Final Decision**: In the final step, the agent looks at all previous information gathered and focuses on taking the best action given the game state and the previous sequential reasoning steps.

Given the nature of the sequential reasoning, it is not possible to parallelize the reasoning step. Hence, the total inference time for a large capable LLM such as DeepSeek with the architecture described here is about 30 seconds to 1 minute, depending on the length of the game history. While for autoregressive LLM models "thinking" is proportional to the output tokens generated, there is some overhead in the startup of an API call. Hence the sequential agent was deemed too slow to mimic human response times for a live tournament with humans (see Section 4.6).

The idea of this model is that it learns via a sequence of abstractions. Here, learning was done via optimising the sequence of information steps and the prompts given to the model to abstract the relevant information.

The design of SeMA currently relies on experts to induce a reasoning structure via specific prompts and agentic subflows.

In more advanced settings, this reasoning structure could be learned autonomously by a meta-agent leveraging frameworks like Sutton's OaK architecture [27, 28]. An OaK-inspired agent would utilize continual experiential learning [26] to autonomously construct its own latent features, formulate its own reasoning subtasks, and develop strategic options directly from its interactions within the game environment.

3.3 Sequential One-step Agent (SeOA)

The Sequential One-step Agent SeOA is basically a simplified version of SeMA, where instead of sequential six calls to the language model a single call is made to speed up inference time. While the agent gains in decision speed, it loses the strength of the concept of sequential abstraction which relies on step-by-step focused small decisions to come to its final conclusion. While the SeOA agent was performing at a level that was not worse than human play, agent SeMA outperforms agent SeOA in internal tournaments.

3.4 Game-Theory-Thinking Agent (GaTTA)

The Game-Theory-Thinking Agent (GaTTA) calls DeepSeek-R1 to leverage inference-time thinking for complex reasoning. Inference-time thinking is a search over reasoning paths at a particular decision point for the agent. The model generates a chain of thought where it explores an approach, evaluates whether it is on the right track, backtracks when it hits a bad option or dead end, and tries alternative strategies.

DeepSeek-R1 was trained via Group Relative Policy Optimization (GRPO) to generate an internal reasoning trace prior to its final output. It is reported to exhibit emergent reasoning behaviors such as self-verification, backtracking, and dynamic strategy adaptation [10–12, 17].

In the GaTTA framework, the LLM is provided the rules of the game and the complete game state. We guide GaTTA's native reasoning phase by providing *thinking process guidelines*. These guidelines do not dictate how the model thinks, but rather constrain what strategic concepts it should apply during its internal search. GaTTA is instructed to apply the following frameworks during its reasoning phase:

(1) **Logic Systems:** Use of formal logic (*Modus Ponens*, *Modus Tollens*) and proof by contradiction.
(2) **Systematic Work:** Creation of mental tables or grids to track possibilities.
(3) **Game Theory:** Application of cooperating game strategies relevant to their specific role.
(4) **Verification:** A distinct step to double-check constraints before finalizing the output.
(5) **In essence:** The prompt asks the LLM to roleplay a possibly deceptive player who must calculate the optimal lie (or truth) to tell the other players, based on a mathematical analysis of the current claims and the probability of other players' roles.

GaTTA learns only within one game. It applies its own inference-time thinking to learn about the nature of the players based on their behavior. This seems akin to constructing a detailed mental model of the game state and history to deduce hidden roles and identify inconsistencies. This allows the agent to make high-level strategic decisions, such as using deceptive announcements (bluffing) to manipulate opponent targeting, while attempting to pass the turn to perceived allies to maintain control of the game flow.

The inference-time thinking also introduces weaknesses, in particular an apparent "analysis paralysis" where the agent overly complicates simple decisions or cycles between conflicting deductions. GaTTA's reasoning trace scales automatically with perceived complexity, hence GaTTA can suffer from long inference times of more than 2 minutes.

3.5 Trained Transformer Agent (TraiTA)

In order to train these models, we propose a three-phase post-training pipeline to adapt a pre-trained language model to play ToT. We opted for the Qwen3 family, specifically focusing on the 0.6B parameter model to balance reasoning capabilities with computationally tractable reinforcement learning.

In the first phase, our methodology generates a warm-start dataset by serving a larger teacher model (e.g., Qwen3-30B-A3B) locally via vLLM. This teacher plays as a single agent against five ToT-ESCHER opponents across tens of thousands of complete games, capturing full game transcripts from the language model player's perspective. In the second phase, we fine-tune Qwen3-0.6B-Instruct on these transcripts using supervised learning (via TRL) until the model demonstrates basic strategic competence and produces valid game moves.

The third phase employs Group Relative Policy Optimization (GRPO) [24] in a mixed-opponent setting. In each training episode, the learning agent is randomly assigned a role alongside five opponents drawn from a pool of different ESCHER agents and earlier checkpoints of the TraiTA agent. The agent receives the complete structured game history as input, in the same way SeMA and GaTTA receive the game history as input. Policy updates are driven by a reward function that penalizes illegal moves, while rewarding winners in the game.

We plan, as work in progress, that this reinforcement learning phase will process hundreds of thousands of simulated games across several days on a single A100 GPU. Continuous checkpointing and evaluation against a fixed tournament of ESCHER agents are being implemented to track learning progress. We aim to analyse whether a localized, small-parameter model can learn to abstract more effectively strategic complexities of social deduction in ToT than a pre-trained large language model can based on its general language understanding.

4 EXPERIMENTAL RESULTS

This section provides learning results for ToT-ESCHER and LLM based agents. There are three different test to benchmark the algorithms: against the best responder, against different algorithms, and in play with humans at an open event that showcased this research at our university.

4.1 ToT-ESCHER

ToT-ESCHER showed higher results when attackers restrict their policy to be always truthful and defenders are free to learn across protocols. We tried multiple experiments to see the self-emergence of truthful messaging or better competitive play, but without success, for two likely reasons. First, the model needs a more descriptive state space that includes the history of play. Second, because of the

coordination nature of the game, the learning procedure cannot find strategies that will improve every player from a team. An example of the latter can be seen in the simple game (1) in Section 5, where replicator dynamics fails to converge to the team-maxmin equilibrium.

The table below presents the win rates with 95% confidence interval of ToT-ESCHER against different types of agents: random, naive agents, which are common heuristics used in human play and specifically described in Appendix A, and approximated best responders computed with the proximal policy optimisation (PPO) algorithm. The PPO algorithm maintains the same non-recurrent state space but includes more historical information about the environment, in particular, who has lied, and the history of the announcement phase. This state with extra information was tested in ToT-ESCHER without a change in the evaluation results. The ToT-ESCHER algorithm was trained for 300 iterations. ToT-ESCHER achieved its highest win rate against random opponents and naive players for defenders and less so for attackers. The PPO algorithm exposes that ToT-ESCHER architecture is still exploitable. Further tests were conducted for the approximated best response against ToT-ESCHER (first level) with a new trained best response (second level, shown in parentheses in the table) using the PPO algorithm. The first level achieved 50% exploitability for defenders and 27% for attackers but with lower results against random and naive players.

Team	Random	Naive Players	Best Responders
Defenders	93.8% (92.7%, 94.9%)	57% (54.8%, 59.2%)	31.4% (29.4%, 33.4%)
Attackers	74.65% (72.7%, 76.6%)	39.2% (37.0% , 41.3%)	25.7% (23.8%, 27.6%)

The results suggest that a change in state abstraction to incorporate history in ToT-ESCHER could help decrease exploitability against the worst-case adversary. An alternative architecture is to run PSRO using ToT-ESCHER as a first instance. For attackers, there has to be a more robust mechanism to teach how to coordinate better.

4.2 LLM agents SeOA-DeepSeek vs. SeOA-Llama-4 Maverick

An initial test was conducted with Sequential One-step Agent (SeOA) on different Large Language Models: DeepSeek SeOA and SeOA Llama-4 Maverick. We chose a 6-player game (similar to the open day of playing with humans, see Appendix B) to assess LLM agents that play against each other.

Metric	Matchup A	Matchup B
Attacker Agent	SeOA DeepSeek	SeOA Llama-4-Maverick
Defender Agent	SeOA Llama-4-Maverick	SeOA DeepSeek
Attacker Win %	75%	20%

In total, SeOA-DeepSeek has a 75% win rate as an attacker against SeOA-Llama-4 Maverick. This is a better performance than the 25% playing random. In contrast, SeOA-Llama-4 Maverick underperforms random play. This suggests that the selection of the type and size of the underlying Large Language Model matters significantly in sequential play.

4.3 LLM agents SeOA-DeepSeek vs. GaTTA-DeepSeek-R1

The following table shows the performancce comparison between the GaTTA-DeepSeek agent and the standard SeOA-DeepSeek agent.

Metric	Matchup A	Matchup B
Attacker Agent	GaTTA DeepSeek-R1	SeOA DeepSeek
Defender Agent	SeOA DeepSeek	GaTTA DeepSeek-R1
Attacker Win %	45%	30%

The sample size is 20. Both agents are defeated as attackers more than 50% of the time, but SeOA-DeepSeek is more exploitable.

4.4 LLM agents SeMA DeepSeek vs. GaTTA DeepSeek-R1

Given the unconstrained, complexity-dependent inference-time thinking for GaTTA, and the multiple number of inference decision calls for SeMA, running these experiments is time-consuming as one game can last longer than one hour. We plan to extend our basic infrastructure to include parallelization.

We currently study if inducing a game-theory-based decision structure on SeMA can lead to superior performance compared to pure inference-time reasoning with relevant guidance. Early results indicate that a simplistic version of SeMA (with the described architecture) has slightly worse performance (average win rate) than a GaTTA agent that relies on its inference-time thinking.

Indeed, more sophisticated architectures for SeMA seem to lead to superior results than GaTTA. There are strong indications that both SeMA and GaTTA may play at average human level or better. The current speed of these decision-making agents (2 min+ for GaTTA and up to 40 seconds for SeMA) is significantly slower than human decision-making in such scenarios. All LLM-based results are based on a limited number of data points. Hence these preliminary results are only indicative.

4.5 LLM agents vs. ESCHER

This section compares ESCHER agents and the different LLM agents, SeMA, SeOA, and GaTTA, as adversaries in a game. The nature of reasoning of ESCHER and LLM based agents is vastly different. ESCHER and TraiTA are trained in self-play (against itself or other agents), while SeMA and GaTTA agents learn by divising reasoning trees at each decision step or via meta-structure learning of an agentic architecture and implied reasoning flow. Hence comparing different versions of these architectures in tournaments is interesting. Given the slow decision speed of LLM agents, analysis takes time. We are experimenting with playing parallel independent games to speed up the analysis. A particular open research question is the trade-off between pre-trained abstraction abilities based on general language concepts vs. specifically learned abstraction via self-training using the same language-based game state when we compare.

Preliminary results suggest that ToT-ESCHER exploits random attackers better than the LLM agents. All agents as attackers outperform random play. In fact, GaTTA seems to perform worst against random play, both as attacker and as defender. It is the only agent where random attackers have a higher win rate (39 percent) than

against random (28 percent). GaTTA might "overthink" random opponents; it performs best against itself and generally better against sophisticated opponents than random players or basic strategies.

Att \ Def	ToT-Esch	Random	SeMA	SeOA	GaTTA
ToT-Esch	45%	75%	x %	x %	x %
Random	6%	28%	22%	20%	39%
SeMA	x %	48%	38%	29%	31%
SeOA	x %	40%	48%	35%	37%
GaTTA	x %	32%	51%	20%	51%

The above is the matrix of win rates of attackers (row players) vs defenders (column players), (x % = to be completed). For all experiments (table entries) involving GaTTA, 50 games have been played. For all other experiments, 100 games have been played.

4.6 Non-expert humans vs. LLM agents

Our university's summer festival 2025 had a Temple Of Terror game session where the public could join, play against each other, and had the option to choose secretly in the web app whether to play or to let a SeOA-DeepSeek agent play for them. For each game, six players took part by logging into a web application while sitting around one table in one room. In total, nine games were played at the event, each involving six participants. The win rates, disaggregated by agent type and role, are as follows.

Type	Role	Count	Win Rate
Human	Attacker	21	52%
Human	Defender	7	43%
Bot	Attacker	15	60%
Bot	Defender	11	45%

The data suggest that the automated agents performed at a level comparable to non-expert human players. While the agents demonstrated a slightly higher win rate, the sample size remains limited. The baseline win rate for attackers under random play in a six-player setting is approximately 25 percent. Both human and agent attackers significantly outperformed this baseline, indicating the ability to cooperate and reason in human-like fashion within this complex game environment.

Following each game, participants were asked to vote about each player whether it was an automated agent or not. The gameplay incorporated both strategic actions and natural language chat, both of which could have been used to distinguish automated bots from human players.

The results indicate that participants struggled to distinguish between human and agent players. On average, only one in five participants correctly identified an agent. While there was a slight tendency to correctly vote for actual agents over humans, human players also received, on average, just under 0.5 votes for being an agent. These findings suggest that the agents were hard to distinguish from human players, both in their strategic actions and their communication. However, the number of samples is too small for statistical significance, and we did not conduct a random control trial. We plan further tests to validate the hypothesis that LLM bots can credibly mimic human players.

5 GAME-THEORETIC ANALYSIS

AI is useful for game theory in that it allows investigating more complex and realistic games. Conversely, game theory provides concepts and benchmarks for evaluating a multi-agent system.

ToT has a large game tree and involves more than two players, which makes a direct game-theoretic analysis difficult. Our work that we have described so far is based on various forms of learning agents with neural networks. Nevertheless, the game raises a number of interesting conceptual questions for game theory that we outline in this section; much of this is work in progress.

5.1 Team games

In a non-cooperative game, a *team* is a set of players with identical payoffs for every outcome. Suppose the game has two teams with opposing, zero-sum payoffs. If the team had a way to randomize in a correlated way, this would just be the situation of a standard two-person zero-sum game.

Consider the following three-player game where each player has the two actions 0 or 1, with strategy triples (p, q, r). Players 1 and 2, who choose p and q, form a team whose payoffs are shown here, with player 3 who chooses r minimizing those payoffs:

p, q	$0,0$	$0,1$	$1,0$	$1,1$
$r = 0$	0	6	$4 - \varepsilon$	0
$r = 1$	0	4	6	0

(1)

Assume first that $\varepsilon = 0$, which is the game (11) in [32]. Played as a two-player game, the game has value 5 where player 3 randomizes with $(\frac{1}{2}, \frac{1}{2})$ and the team randomizes (correlated among the players) with probabilities $(0, \frac{1}{2}, \frac{1}{2}, 0)$ for the shown columns.

Consider now (1) as a three-player game where the players can only randomize independently, with (p, q, r) extended to probabilities for the players' second strategies. Then the team can no longer correlate. The game has three Nash equilibria, which are the pure equilibria $(p, q, r) = (0, 1, 1)$ with payoff 4 for the team, $(1, 0, 0)$ with payoff $4 - \varepsilon$ (even for ε near zero), and $(\frac{1}{2}, \frac{1}{2}, \frac{1}{2})$ with payoff 2.5 to the team. If $\varepsilon \geq 0$, the best possible payoff to the team is 4.

The payoff 4 in (1) is also the *team-maxmin* payoff, which the team can guarantee to get, by playing $(p, q) = (0, 1)$, irrespective of the action of their opponent. In general, the team-maxmin action profile requires mixed strategies, for a guaranteed expected payoff. As shown in [32], in zero-sum games of a team against a single opponent, every team-maxmin action profile can be extended to a Nash equilibrium. That is, the opponent can exploit the team's inability to randomize jointly by adopting an equilibrium strategy where no single team player has an incentive to deviate. The team-maxmin payoff is also the best possible payoff to the team, and thus represents a natural "value" for such games. The corresponding team-maxmin equilibrium is not unique, as when $\varepsilon = 0$ in (1). Generically (here when $\varepsilon \neq 0$), it is unique.

In zero-sum games between two teams that both have more than one player, there is in general no unique team-maxmin equilibrium. The possible payoff gaps are studied in [23].

The team-maxmin equilibrium solution concept is appropriate in a setting where one team can act as a single player. As mentioned

in Section 1.3, the original ToT rules state that the defenders identify each other. In that case this concept could provide a possible performance benchmark.

One question is if a learning dynamics converges to the team-maxmin equilibrium. We have investigated this for the replicator dynamics from evolutionary game theory (see, for example, [22]). However, this is not the case: The replicator dynamics can also locally converge to another Nash equilibrium, as to the equilibrium $(1, 0, 0)$ in (1) that is suboptimal for the team when $\varepsilon > 0$.

5.2 Roleplay team games

As described, it turns out that ToT is not a team game. Assume for simplicity that there are only three players, two attackers and one defender. This seems to match the situation of a team against a single adversary, with the team-maxmin equilibrium of [32] as an appropriate solution concept.

An original application of the team-maxmin framework [32] is to consider a zero-sum game in extensive form, where only one player (the adversary) has perfect recall but the other player has not. Then a behavior strategy for the latter player is not as powerful as for the player with perfect recall, and in general a maxmin behavior strategy guarantees a lower payoff than a mixed strategy that allows correlated moves across information sets. By considering that player as a team of agents, one for each information set, their independent random choices correspond to a behavior strategy in the extensive game, and the team-maxmin game description applies.

However, in ToT the *players* in the game are not the team members. The two teams are the attackers and defenders, and a player assumes one of these roles according to an initial chance move. In a team game, the team membership of a player has to be known in order to define the corresponding payoffs. However, it is possible to apply a suitable transformation that changes the game into one of two opposing teams, which we outline here as follows. The sets of players are indeed two teams of attackers and defenders. Each game participant is assigned one of the two roles as a kind of "actor on behalf" of their role. The attackers or defenders then win according to which of these actors uncovers the winning gold or fire card. The game has imperfect information and, in a sense, imperfect recall, because the assumed roles are not publicly known.

This is a conceptual leap that requires careful consideration with respect to the symmetry of the two teams. It is best understood with some small games first, which we have done with some three-player "matching pennies" games, however in a different way than proposed in [6]. We have not yet applied these insights to ToT.

5.3 Signaling games

ToT offers also a connection to signaling games and to models of language in game theory. A main issue in this context is that the messages sent by players, here their card declarations, are intrinsically meaningless. That is, any Nash equilibrium where players declare their cards in certain numbers as "gold" or "fire" can equally replaced by one with the two words exchanged and thus having their exact opposite meaning (assuming that there are no limits on declared card numbers). This is standard assumption about cheap-talk games where signals are arbitrary [14]. What is the appropriate game-theoretic concept here? Costly signals, where it should be easier to tell the truth than to lie? In game theory, this issue seems to have only been considered in very stylized single-round sender-receiver games [1]. Social deduction games such as ToT may provide an interesting conceptual challenge for game theory in this context.

6 CONCLUSIONS

Temple of Terror (ToT) is a social deduction game with simple rules and a small set of possible messages, namely card declarations and actions for choosing the next player to act. It is a zero-sum game between two teams and thus amenable to tournament play that reveals good performance, and the potential of generating large amounts of training data for learning agents.

For a game-theoretic analysis and identifying and proving a Nash equilibrium, the game is too large. However, it gives rise to conceptually interesting questions for game theory (outlined in Section 5): Can the framework of team games [23, 32] be applied properly by letting the players be the roles (of attacker or defender), with game participants acting on their behalf? And what are the assumptions about symmetry and perfect recall in this context? Secondly, can the literature about signaling games [14] be adapted such that truth-telling and lying are naturally associated with messages, in contrast to considering language as cheap talk?

In this paper we have designed machine-learning agents in two ways to learn ToT. The first is based on the ESCHER framework that employs counterfactual regret with a directed exploration of the game tree with a representative generation of game play trajectories to reduce the noise in training data. The corresponding components are represented by separate neural networks. This approach can be used to generate large amounts of training data. It is largely model-free, except for the provision of some plausible strategies for which ESCHER computed a behavior of fairly even randomization probabilities. As *future improvements* we envisage using the transformer architecture for processing the players' sequence of actions to approximate the histories for the neural networks prediction.

Our second approach has been to design various LLMs driven by prompts that describe rules of the game, the history of play, and the public and private information in the game. The model is then asked to make suitable moves at each turn during play. Depending on the size of the used model, its "reasoning capability" reveals more or less sensible choices for the players.

Decision speed of the LLM agents varies vastly. The GaTTA agent that uses the largest model exhibits unpredictable, often very slow decision speed. This makes GaTTA not viable for generating large amounts of inference data. A single GaTTA decision often takes more than 2 minutes. This is even unsuitable for use in play with humans. Our web app (see Appendix B) can be used to collect data on how humans play, but so far not enough data has been collected for a rigorously controlled statistical trial. We have played the various agents against each other (see Section 4). Sequential reasoning that uses individual LLM calls for different analysis steps before making a decision (SeMA) seems not to outperform SeOA that conducts the same analysis in one LLM call. It may be worth investigating if random play confuses GaTTA and its "thinking" that possibly tries too hard to assign sophisticated reasoning to random behavior.

There is clearly room for improvement for better benchmarks and substantially larger number of games played.

ACKNOWLEDGMENTS

The authors thank the LSE Data Science Institute for the computational resources to run the study. In addition, we thank Gabriele Farina, Matteo Bettini, Rahul Savani, Theodore Turocy, Sherif Eltarabishy, and the reviewers for their critical contributions to this work. Steffen Issleib was supported by the LSE Research Investment Fund project "Transformers for Improved Strategic Learning".

REFERENCES

[1] Ivan Balbuzanov. 2019. Lies and consequences: The effect of lie detection on communication outcomes. *Int. J. Game Theory* 48, 4 (2019), 1203–1240.

[2] Guillermo Barbadillo. 2024. Solution Summary for ARC24. *ARC Prize 2024 – 2nd Place Solution Write-up* (2024). https://ironbar.github.io/arc24/05_Solution_Summary/ Accessed: 2025-05-01.

[3] BBC. 2022. The Traitors: How the Show Works. https://web.archive.org/web/20221212171149/https://www.bbc.co.uk/programmes/articles/2xRtJpWRbKcwPdXs7bZnxxH/the-traitors-how-the-show-works. Accessed: 26 February 2026.

[4] Noam Brown and Tuomas Sandholm. 2019. Superhuman AI for multiplayer poker. *Science* 365, 6456 (2019), 885–890. https://doi.org/10.1126/science.aay2400 arXiv:https://www.science.org/doi/pdf/10.1126/science.aay2400

[5] Emilio Calvano, Giacomo Calzolari, Vincenzo Denicolo, and Sergio Pastorello. 2020. Artificial Intelligence, Algorithmic Pricing, and Collusion. *Am. Econ. Rev.* 110, 10 (2020), 3267–3297.

[6] Luca Carminati, Brian Hu Zhang, Gabriele Farina, Nicola Gatti, and Tuomas Sandholm. 2024. Hidden-role games: Equilibrium concepts and computation. In *Proceedings of the 25th ACM Conference on Economics and Computation (EC '24)*. Association for Computing Machinery, New York, NY, USA.

[7] Andrea Celli and Nicola Gatti. 2018. Computational Results for Extensive-Form Adversarial Team Games. *Proceedings of the AAAI Conference on Artificial Intelligence* 32, 1 (Apr. 2018). https://doi.org/10.1609/aaai.v32i1.11462

[8] François Chollet. 2024. OpenAI o3 Breakthrough High Score on ARC-AGI-Pub. *ARC Prize Blog* (2024). https://arcprize.org/blog/oai-o3-pub-breakthrough Published 20 December 2024. Accessed: 2025-05-01.

[9] Dimma Davidoff. 1987. The Original Mafia Rules. https://gusandco.net/wp-content/uploads/2021/04/The-Original-Mafia-Rules-1.pdf. Accessed: 26 February 2026.

[10] DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z.F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, Bing Xue, Bingxuan Wang, Bochao Wu, Bei Feng, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, Damai Dai, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, et al. 2025. DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning. *Nature* 645, 8081 (2025), 633–638. https://doi.org/10.1038/s41586-025-09422-z arXiv:2501.12948 [cs.CL]

[11] DeepSeek-AI, Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, Damai Dai, Daya Guo, Dejian Yang, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, et al. 2024. DeepSeek-V3 Technical Report. *arXiv preprint* (2024). https://doi.org/10.48550/arXiv.2412.19437 arXiv:2412.19437 [cs.CL]

[12] DeepSeek-AI, Aixin Liu, Aoxue Mei, Bangcai Lin, Bing Xue, Bingxuan Wang, Bingzheng Xu, Bochao Wu, Bowei Zhang, Chaofan Lin, Chen Dong, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenhao Xu, Chong Ruan, Damai Dai, Daya Guo, Dejian Yang, Deli Chen, Erhang Li, Fangqi Zhou, Fangyun Lin, Fucong Dai, Guangbo Hao, Guanting Chen, Guowei Li, et al. 2025. DeepSeek-V3.2: Pushing the Frontier of Open Large Language Models. *arXiv preprint* (2025). https://doi.org/10.48550/arXiv.2512.02556 arXiv:2512.02556 [cs.CL]

[13] Meta Fundamental AI Research Diplomacy Team (FAIR), Anton Bakhtin, Noam Brown, Emily Dinan, Gabriele Farina, Colin Flaherty, Daniel Fried, Andrew Goff, Jonathan Gray, Hengyuan Hu, Athul Paul Jacob, Mojtaba Komeili, Karthik Konath, Minae Kwon, Adam Lerer, Mike Lewis, Alexander H. Miller, Sasha Mitts, Adithya Renduchintala, Stephen Roller, Dirk Rowe, Weiyan Shi, Joe Spisak, Alexander Wei, David Wu, Hugh Zhang, and Markus Zijlstra. 2022. Human-level play in the game of Diplomacy by combining language models with strategic reasoning. *Science* 378, 6624 (2022), 1067–1074.

[14] David M. Kreps and Joel Sobel. 1994. Signalling. In *Handbook of Game Theory with Economic Applications*, Robert J. Aumann and Sergiu Hart (Eds.). Vol. 2. North-Holland, Amsterdam, 849–867.

[15] Harold W. Kuhn. 1953. Extensive games and the problem of information. In *Contributions to the Theory of Games, Vol. II*, Harold William Kuhn and Albert William Tucker (Eds.). Annals of Mathematics Studies, Vol. 28. Princeton University Press, Princeton, NJ, 193–216.

[16] Marc Lanctot, Kevin Waugh, Martin Zinkevich, and Michael Bowling. 2009. Monte Carlo Sampling for Regret Minimization in Extensive Games. In *Advances in Neural Information Processing Systems*, Y. Bengio, D. Schuurmans, J. Lafferty, C. Williams, and A. Culotta (Eds.), Vol. 22. Curran Associates, Inc. https://proceedings.neurips.cc/paper_files/paper/2009/file/00411460f7c92d2124a67ea0f4cb5f85-Paper.pdf

[17] Sara Vera Marjanović, Arkil Patel, Vaibhav Adlakha, Milad Aghajohari, Parishad BehnamGhader, Mehar Bhatia, Aditi Khandelwal, Austin Kraft, Benno Krojer, Xing Han Lù, Nicholas Meade, Dongchan Shin, Amirhossein Kazemnejad, Gaurav Kamath, Marius Mosbach, Karolina Stańczak, and Siva Reddy. 2025. DeepSeek-R1 Thoughtology: Let's think about LLM Reasoning. *arXiv preprint* (2025). https://doi.org/10.48550/arXiv.2504.07128 arXiv:2504.07128 [cs.CL]

[18] Stephen Marcus McAleer, Gabriele Farina, Marc Lanctot, and Tuomas Sandholm. 2023. ESCHER: Eschewing Importance Sampling in Games by Computing a History Value Function to Estimate Regret. *The Eleventh International Conference on Learning Representations* (2023). https://openreview.net/forum?id=35QyoZv8cKO

[19] Stephen Marcus McAleer, Gabriele Farina, Gaoyue Zhou, Mingzhi Wang, Yaodong Yang, and Tuomas Sandholm. 2023. Team-PSRO for Learning Approximate TMECor in Large Team Games via Cooperative Reinforcement Learning. In *Thirty-seventh Conference on Neural Information Processing Systems*. https://openreview.net/forum?id=lCThtrJxoH

[20] Richard Ng. 2021. Tempel des Schreckens – board game rules in English. https://github.com/richardcrng/tempel-des-schreckens/blob/main/RULES.md. Accessed: 26 February 2026.

[21] Julien Perolat, Bart De Vylder, Daniel Hennes, Eugene Tarassov, Florian Strub, Vincent de Boer, Paul Muller, Jerome T. Connor, Neil Burch, Thomas Anthony, Stephen McAleer, Romuald Elie, Sarah H. Cen, Zhe Wang, Audrunas Gruslys, Aleksandra Malysheva, Mina Khan, Sherjil Ozair, Finbarr Timbers, Toby Pohlen, Tom Eccles, Mark Rowland, Marc Lanctot, Jean-Baptiste Lespiau, Bilal Piot, Shayegan Omidshafiei, Edward Lockhart, Laurent Sifre, Nathalie Beauguerlange, Remi Munos, David Silver, Satinder Singh, Demis Hassabis, and Karl Tuyls. 2022. Mastering the game of Stratego with model-free multiagent reinforcement learning. *Science* 378, 6623 (2022), 990–996. https://doi.org/10.1126/science.add4679 arXiv:https://www.science.org/doi/pdf/10.1126/science.add4679

[22] William H. Sandholm. 2010. *Population Games and Evolutionary Dynamics*. MIT Press, Cambridge, MA.

[23] Leonard J. Schulman and Umesh V. Vazirani. 2019. The duality gap for two-team zero-sum games. *Games Econ. Behav.* 115 (2019), 336–345.

[24] Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y.K. Li, Y. Wu, and Daya Guo. 2024. DeepSeekMath: Pushing the Limits of Mathematical Reasoning in Open Language Models. *arXiv preprint* (2024). https://doi.org/10.48550/arXiv.2402.03300 arXiv:2402.03300 [cs.CL]

[25] 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. 2016. Mastering the game of Go with deep neural networks and tree search. *Nature* 529, 7587 (2016), 484–489. https://doi.org/10.1038/nature16961

[26] David Silver and Richard S. Sutton. 2025. Welcome to the Era of Experience. *To appear in Designing and Intelligence, G. Konidaris, Ed., MIT Press* (2025). https://storage.googleapis.com/deepmind-media/Era-of-Experience%20/The%20Era%20of%20Experience%20Paper.pdf Preprint, April 2025.

[27] Richard S. Sutton, Michael Bowling, and Patrick M. Pilarski. 2022. The Alberta Plan for AI Research. *arXiv preprint* (2022). https://doi.org/10.48550/arXiv.2208.11173 arXiv:2208.11173 [cs.AI]

[28] Richard S. Sutton, Marlos C. Machado, G. Zacharias Holland, David Szepesvari, Finbarr Timbers, Brian Tanner, and Adam White. 2023. Reward-Respecting Subtasks for Model-Based Reinforcement Learning. *Artificial Intelligence* 324 (2023), 104001. https://doi.org/10.1016/j.artint.2023.104001 arXiv:2202.03466 [cs.AI]

[29] UltraBoardGames. 2020. Avalon Game Rules. https://www.ultraboardgames.com/avalon/game-rules.php. Accessed: 26 February 2026.

[30] Oriol Vinyals, Igor Babuschkin, Wojciech M. Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H. Choi, Richard Powell, Timo Ewalds, Petko Georgiev, Junhyuk Oh, Dan Horgan, Manuel Kroiss, Ivo Danihelka, Aja Huang, Laurent Sifre, Trevor Cai, John P. Agapiou, Max Jaderberg, Alexander S. Vezhnevets, Rémi Leblond, Tobias Pohlen, Valentin Dalibard, David Budden, Yury Sulsky, James Molloy, Tom L. Paine, Caglar Gulcehre, Ziyu Wang, Tobias Pfaff, Yuhuai Wu, Roman Ring, Dani Yogatama, Dario Wünsch, Katrina McKinney, Oliver Smith, Tom Schaul, Timothy Lillicrap, Koray Kavukcuoglu, Demis Hassabis, Chris Apps, and David Silver. 2019. Grandmaster level in StarCraft II using multi-agent reinforcement learning. *Nature* 575, 7782 (2019), 350–354. https://doi.org/10.1038/s41586-019-1724-z

[31] Bernhard von Stengel. 2022. *Game Theory Basics*. Cambridge University Press, Cambridge, UK.

[32] Bernhard von Stengel and Daphne Koller. 1997. Team-maxmin equilibria. *Games Econ. Behav.* 21, 1-2 (1997), 309–321.

[33] Martin Zinkevich, Michael Johanson, Michael Bowling, and Carmelo Piccione. 2007. Regret minimization in games with incomplete information. In *Proceedings of the 21st International Conference on Neural Information Processing Systems* (Vancouver, British Columbia, Canada) *(NIPS'07)*. Curran Associates Inc., Red Hook, NY, USA, 1729–1736.

A NAIVE PLAYERS

This appendix describes two different heuristics of play used by humans and denoted by the name Naive Attackers and Naive Defenders.

Naive Attacker plays an honest, information driven strategy. In the message phase, it always tells the truth, reporting its exact fire and gold card counts. During the action phase, it actively tries to detect liars by cross-referencing public messages with revealed table cards: any agent whose claimed count falls below what is physically visible has been caught lying. It also performs a private detection step using knowledge of its own hand to flag impossible claims by others. With liars filtered out, it preferentially targets agents claiming zero fire and positive gold, maximizing its chance of revealing gold cards.

Naive Defender plays a deceptive strategy. In the message phase, it chooses between two modes each round: a deterrence mode (default, 65% probability) where it inflates its fire claim and reports zero gold and a deceitful mode (35% probability, only when fire is in hand) where it disguises some fire cards as gold, confusing gold-seeking attackers into revealing fire instead. In the action phase, it plays as a fire-hunter: it first targets agents who already have fire visible on the table, then those claiming the highest remaining fire in their messages, and finally falls back to agents with the fewest claimed gold cards.

The naive attacker's truthfulness is directly exploited by the naive defender's deceitful strategy: since the attacker trusts messages and hunts for gold, a defender broadcasting fake gold becomes a dangerous trap. Conversely, the attacker's liar-detection mechanism partially counters the deterrence mode. This creates a common adversarial heuristics deployed by humans: the defender's deception quality is bounded by detectability, while the attacker's effectiveness depends on how reliably it can distinguish truth from lies.

B UNIVERSITY EVENT WEB APPLICATION

The backend is written in Flask/Python. The frontend code uses Node.js and npm for compiling.

We use a Hetzner server to host both the web app and another Hetzner server to host the agent. The agent is called from the web app server via FastAPI from the agent's server.

The app can be found at URL http://138.199.197.147/ as a setup for human players.

C TOT-ESCHER ALGORITHM INFORMATION

The ToT-ESCHER algorithm is configured with the following training hyperparameters. Each outer iteration samples $P = 1{,}000$ trajectories per learning player via outcome-sampling Monte Carlo CFR. The replay buffers store up to 10,000,000 transitions, and training draws mini-batches of $|\mathcal{B}| = 2{,}500$ samples for the regret and utility networks and $|\mathcal{B}\sigma| = 2{,}000$ samples for the policy network. The regret network is trained for 300 gradient steps per iteration, the utility network for 300 steps, and the policy network for 500 steps every 3 outer iterations. The Adam optimiser is used throughout, with learning rates $\alpha_{\text{regret}} = 10^{-3}$, $\alpha_{\text{utility}} = 3\times10^{-4}$ (with ℓ_2 weight decay $\lambda = 10^{-4}$), and $\alpha_\sigma = 3\times10^{-4}$. Gradients of the utility network are clipped to a maximum ℓ_2 norm of 1.0.

Three feed-forward neural networks are employed, each sharing a common depth of four layers and a hidden dimension of 256 units. All networks use ReLU activations on hidden layers and orthogonal weight initialisation, with gain $\sqrt{2}$ on hidden layers and gain 0.01 on the output layer, which stabilises early training. The *Regret Network* $\hat{R}_\rho$ maps an observation $o \in \mathbb{R}^{|O|}$ to a masked vector of cumulative counterfactual regrets over the action space $\mathcal{A}$, with illegal actions zeroed out by an element-wise Boolean mask. The *Utility Network* $\hat{Q}_\theta$ takes as input the concatenation of an observation and a one-hot action encoding and outputs a scalar estimate of the expected utility; its training objective is mean-squared error against sampled Monte Carlo returns. The *Policy Network* $\hat{\sigma}_\phi$ maps an observation to a probability distribution over $\mathcal{A}$: after applying the action mask (setting logits of illegal actions to -10^9), a softmax is applied to produce the average strategy approximation, trained by cross-entropy against the observed empirical action frequencies of opponent nodes.

Each agent's observation vector has dimension $|O| = 39 + |\mathcal{M}|$, where $|\mathcal{M}| = 7$ is the number of communication protocols, giving $|O| = 46$. The observation is augmented with a one-hot encoding of the agent's chosen communication protocol before being fed to any network. The joint action space has size $|\mathcal{A}| = (N-1)+|\mathcal{M}| = 11$ for $N = 5$ players, covering both the card-selection actions (choosing one of the $N-1$ other agents' cards to reveal) and the $|\mathcal{M}|$ messaging actions. Message-phase actions are offset by $N - 1 = 4$ in the index space to share a unified action representation across both game phases.

D PROMPTS FOR LLM AGENTS

We give the prompt templates for SeOA and GaTTA. SeMA has identical prompts to SeMA, except that the individual prompt steps are executed sequentially with separate LLM calls instead of one LLM call, and the output format is adjusted respectively for each individual call.

D.1 Prompt Templates – SeOA

```
General Task: Complete different subtasks in
    order to then use that information to make
     a final decision.

## Task: Information Comprehension

You are player {player_name}, your role is {
    role}.

Please analyze and summarize the current game
    state, making sure you are aware of your
    own information, focusing on what is known
     with certainty versus what is uncertain.

Identify key facts about the game state that
    will be important for strategic decision-
    making.
```

```
Also repeat clearly the last announcements of
    each player, making it clear that you are
    talking about the last announcement that
    refers to their current cards in the
    current round.

Do not repeat rules, only output precise facts
     about the game state, nothing else.

## Task: Announcement Analysis

For each announcement round, analyze each
    player's announcements vs the cards that
    then have been revealed in that round.

For example if a player would announce 3 gold
    cards and 0 trap cards in that round, but
    then a trap card would be revealed as part
     of the card reveals in that round, then
    the player would have been caught lying.
    That would be inconsistent with his claims
    .

Be aware that announcements are only relevant
    till the next card reshuffle (before the
    new announcement round). So in one round a
     player might announce 0 gold and 0 trap
    cards, and in the card reveal phase in
    that round he would only reveal empty
    cards. Then in the next round he might
    announce that he has 2 gold cards and no
    trap cards and in that round he would
    reveal in fact a gold card. Then he would
    be consistent with his announcements in
    the sense that his announcements were in
    line with his cards for each respective
    round.

For each player:
 - Track how their announcement behaviour has
     been and if in any round their
     announcement has been not true or
     consistent with the cards revealed in
     that round.
 - Note if the total number of gold/trap cards
      they claim matches the game totals,
     taking into consideration your own cards
     in that round.
 - Identify any logical inconsistencies (e.g.,
      claiming more gold than exists).
 - Check if their announcements in each round
     have aligned with cards that have been
     revealed from them in that round.

Focus on finding patterns that might reveal
    whether players are being truthful or
    deceptive.
```

```
Do not repeat rules, only output precise facts
     about each player's announcement patterns
     across redistribution rounds to identify
    inconsistencies or strategies.

## Task: Behavior Analysis

Analyze each player's behavior patterns to
    determine their likely role (attacker or
    defender):

Attackers typically:
 - Target players claiming to have gold and
     truthfully announce their gold cards
 - Avoid targeting players claiming to have
     traps
 - Possibly avoid targeting players that are
     likely to be defenders

Defenders typically:
 - Try to attract to be chosen by attackers
     when they have trap cards by lying about
     their number of trap cards
 - Defenders may overstate their number of
     gold cards to be chosen/ targeted by
     attackers so that they get the next turn
     to choose a player
 - Generally may avoid targeting players that
     only have gold cards
 - Target players who announce to have trap
     cards or a mix of cards

For each player:
 1. Examine who they choose to target and why
 2. Look for patterns across rounds in their
     targeting behavior with regards to the
     announcements of the players they chose
     in that round
 3. Assess whether their behavior is
     consistent with their claimed cards
 4. Determine if their actions align more with
      attacker or defender goals

Provide an assessment of each player's likely
    role based on their behavior. Do not
    repeat rules, only output precise facts
    about each player's likely role based on
    their behavior, nothing else.

## Task: Card Probability Analysis - Cards

Calculate the probability of each player
    having specific cards in their hand in the
     current round:

For each player:
 1. Consider their announcements history and
     how truthful they seem based on the
     previous rounds
 2. Account for cards already revealed from
     their hand in this current round
```

```
 3. Factor in the total remaining gold and
     trap cards in the game, making sure to
     include your own current cards in the
     analysis, so for example if 3 gold and 2
     traps are remaining officially, and you
     have 1 gold and 1 trap, then there are
     only 2 gold and 1 trap that the other
     players could have. Make sure to include
     this fact into your probability analysis
     of other players cards.
 4. Assign percentage probabilities for them
     having gold or trap cards
 5. Include your assessment of their roles, in
      general an attacker is more likely to be
      honest, so if you are confident a player
      is an attacker, then you can assign a
     higher probability that their
     announcement about what cards they have
     in that round are true.

Provide a numerical probability assessment for
     each player:
 - P(has gold): X%
 - P(has trap): Y%

Rank players from most to least likely to have
     gold cards, and separately for trap cards
    . Do not repeat rules, only output the
    rank of players from most to least likely
    to have gold cards, and separately for
    trap cards.

## Task: Card Probability Analysis - Roles

Calculate the probability of each player being
     an attacker. (Probability of being a
    defender is 1 - probability of being an
    attacker):

The rules for number of attackers vs defenders
     are:
There can be 1 or 2 defenders, for 3 or 4
    player games, for all other games there
    are always 2 defenders. The remaining
    players are attackers.
 - For 3 players: 1 defender vs 2 attackers or
      2 defenders vs 1 attacker.
 - For 4 players: 1 defender vs 3 attackers or
      2 defenders vs  2 attackers.
 - For 5 players: always 2 defenders vs 3
     attackers.
 - For 6 players: always 2 defenders vs 4
     attackers.

Take into account your behaviour analysis
    result. Take this into account when
    assessing the probability of the other
    players being an attacker or defender, and
     take into account your own role. So if
    you are a defender, then at most one other
     player can be a defender.
```

```
For each player:
 1. Consider their target player choices and
     what those target players claimed to have
      as cards in that particular round.
 2. Consider their own revealed cards vs their
      claimed cards in each round.
 3. Consider also their target player choices
     with respect to what likely roles these
     target players have, players prefer to
     target players of their own role in
     general.

Provide a numerical probability assessment for
     each player:
 - P(is attacker): X%

Rank players from most to least likely to
    being attacker. Do no repeat rules, only
    output the rank of players from most to
    leas likely to being an attacker.
Remember that you are player {player_name},
    your role is {role}.
        Only the last announcement round is
            about the current cards that the
            players hold now.
        The previous announcement rounds are
            only useful to establish the trust
            -worthyness of a player's
            announcement.

        If you're an attacker:
        - In general you have several
            objectives
        - Targeting players that are most
            likely to have gold cards and no
            trap cards
        - Ideally you target the player with
            the best gold vs empty cards ratio
            , avoid any players that are
            likely to have trap cards
        - try to avoid to target players that
            are defenders, as by targeting
            them you give them the option
            afterwards to target players
            themselves
          This holds true in particular when
              only 1 trap is unrevealed as a
              defender can end the game and
              win with one move then
        - try to establish trustworthy other
            attackers that have gold cards and
             try to target them repeatedly to
            reveal their gold cards in that
            round
```

```
        - Be aware: Never go for a player you
            believe to have traps, a revealed
            trap, no matter by who it got
            chosen to be revealed, will help
            defenders win. So never try to
            reveal or find a trap, you just
            help defenders. Only ever go for
            players that you think are most
            likely to have gold, and are not
            defenders.

        If you're a defender:
        - Initially target players with gold
            or empty cards to avoid early
            suspicion of being a defender
        - Later in the game in rounds 3 or 4
            try to target players that are
            likely to have trap cards when you
             then can end the game (when only
            one trap card is left)
        - Be more aggressive in searching for
            trap cards yourself and in
            misleading by lying about your own
             trap cards and pretending you
            have gold cards when only 1 trap
            card is remaining and hence you
            can win the game when you find the
             next trap card or when some
            player chooses you when you hold a
             trap card.
        - Try not to open gold cards,
            especially when only few golds
            cards are remaining
        - Consider misdirection strategies if
            it could be advantageous to make
            others believe you are not a
            defender

        Just return the respective player name
             as value to the target_player key
            . If the player name is just a
            numeric ID (0, 1, 2, etc.) return
            the target player as just their
            numeric ID, not as 'Player X' or
            similar name additions."
        Provide your final decision as JSON
            and provide only the JSON and
            nothing else, in this JSON format:
        Output your results in the following
            JSON schema:

"type": "json_object",
"value": {
    "properties": {
    "information_comprehension_result":{"type
        ": "string"},
    "announcement_analysis_result":{"type": "
        string"},
    "behaviour_analysis_result":{"type": "
        string"},
    "probability_analysis_cards_result":{"type
        ": "string"},
    "probability_analysis_roles_result":{"type
        ": "string"},
    "primary_reasoning": {"type": "string"},
    "chat_message": {"type": "string"},
    "target_player": {"type": "string"},
    "confidence_level": {"type":"integer","
        minimum":1, "maximum":3 }
    },
"required": ['information_comprehension_result
    ','announcement_analysis_result','
    behaviour_analysis_result','
    probability_analysis_cards_result','
    probability_analysis_roles_result','
    chat_message','target_player', '
    confidence_level','primary_reasoning'],
}
```

For announcement, the decision part above is replaced with this prompt :

```
Remember that you are player {player_name},
    your role is {role}. Your task is to make
    the new announcement about how many gold
    cards and how many trap cards you have now
     in your current private cards.

- Avoid making non-sensical announcements that
     make no sense given the remaining
    outstanding number of gold and trap cards.

If you're an attacker:
- In general attackers benefit from being
    honest and revealing correct information
    to the other players.
- Since most players are attackers be honest
    unless the current game state gives a good
     reason not to be honest.
- Think how much you want to signal your gold
    cards so that other attackers can find
    them. It may be useful to be truthful if
    you have gold cards so other attackers can
     find them.
- Initially reveal trap cards honestly so
    other attackers can learn to trust you,
    however when only 1 trap card is left
    avoid revealing trap cards when it is a
    likely defenders turn or when a defender
    can become the turn player.
- If you have trap cards you need to balance
    honest communication to other attackers vs
     revealing your trap card to defenders.
- If the game can be finished by the next
    player (either only 1 trap or 1 gold cards
     remaining unrevealed) take into
    consideration whose turn it will be to
    start after the announcements. If it is a
    likely defender do not announce honestly
    if you have a trap card, if it is an
    attacker its better to announce truthfully
     if you have a trap card so the likely
    attacker can avoid targeting you.

If you're a defender:
```

```
- In early phases of the game try to avoid to
    reveal too easily that you are a defender,
     avoid lying too much.
- If you have no trap card it may be useful to
     be honest.
- If you have many gold cards it may be
    advantageous to announce not all gold
    cards.
- Later in the game if you have a trap it may
    be useful to lie by pretending to have
    gold instead of the trap so that attackers
     might target you and the trap card gets
    revealed.
- Take into account if you have so far
    revealed that you are a defender (via
    lying or your card choices).
- Consider misdirection strategies if it could
     be advantageous to make others believe
    you are not a defender.
- If only 1 trap card is left in the game it
    may be useful to lie or bluff. If you have
     the trap card and the player whose turn
    it is is likely a defender it may be
    useful to honestly announce the trap card.
- If you have not yet taken actions so that
    others likely know that you are a defender
    , it may be good to pretend to have gold
    instead of a trap to mislead attackers
    into targeting you.

Provide your final announcement decisions as
    JSON and provide only the JSON and nothing
     else, in this JSON format, where the
    value for the 'gold_cards' and 'trap_cards
    ' keys should hold your final announcement
     decision cards, and consistent with your
    'primary reasoning' reasons, not
    necessarily the actual cards you have.
```

D.2 Prompt Templates – GaTTA

```
You are an intelligent agent playing a social
    deduction card game.
You are an agent specialized in finding the
    best action in social
deduction games. When you are asked to take an
     action as
participating agent in a multi player game,
    you must come up with
the next action step-by-step using systematic
    analysis and clear
reasoning.

Response Format

You must always structure your response
    exactly as follows:

<think>
[Your detailed reasoning process here - work
    through the problem
step by step, showing all logical steps,
    eliminations, deductions,
and verification]
</think>

<answer>
[Your final answer decision - just the action
    taken]
</answer>

Critical: Your response will be evaluated by
    extracting only the
content between the <answer> tags. You must
    follow this format
precisely or you will be penalized.

## Game Rules

- N Players are either attackers or defenders
    (roles are secret)
- If N > 4 there are exactly 2 defenders. If N
     <= 4 there can be
  2 or 1 defender.
- So for N > 4 there are exactly N - 2
    attackers.
- When the game starts each player gets
    randomly assigned a role
  (attacker or defender) and gets 4 cards
      distributed.
- Cards are either gold, trap, or empty.
- There are N gold cards and 2 trap cards in
    the game. The rest
  of the cards are empty cards.
- Attackers win when all N gold cards have
    been revealed.
- Defenders win when all 2 trap cards have
    been revealed.
- At the beginning of each round, players
    consecutively make an
  announcement about the number of gold cards
      and the number of
  trap cards they have. They can be honest or
      lie, depending on
  their strategy.
- Then in the second part of each round, the
    player whose turn it
  is chooses another player. One card of this
      targeted player gets
  then revealed at random, any yet unrevealed
      card of that player
  in that round is equally likely to be
      revealed by chance.
- Cards get revealed at random, however
    players influence which
  cards may get revealed by choosing a player
      of whom one of his
  cards gets revealed at random. Note that
      players do not actively
  reveal cards, cards are revealed at random
      from the hand of
  unrevealed cards of the targeted player.
      Players however target
```

```
  a player that has a high probability of
      having a certain card
  that they want to be revealed.
- Now it is the players turn whose card has
    been revealed and he
  can choose the next player, and one of their
      cards gets revealed.
- The fact that only players that have been
    chosen / targeted get
  the chance to choose the next player is
      strategically important.
- It may be strategically advantageous to only
    target a player that
  you are confident has the same role as you,
      in order not to give
  the opponents the chance to target or choose
      players.
- After N players have made a choice of
    another player and hence N
  cards have been revealed at random from the
      targeted players, a
  round ends and all not revealed cards get
      collected, re-shuffled
  and redistributed evenly (redistribution
      phase). So players'
  cards are changing completely from
      redistribution round to
  redistribution round.
- Players announce anew their cards after each
    redistribution phase
  but do not have to be honest, they can be
      honest or lie about
  what their actual cards are. The
      announcement only is valid for
  the current round until cards get reshuffled
      again, so in the
  sequence of announcements, one announcement
      by a player is only
  relevant until his next announcement as
      cards get reshuffled and
  redistributed then.
- There can be up to 4 rounds where each round
    involves the
  announcement phase and then consecutively
      the phase where players
  target other players of which random cards
      get revealed.

----------------CURRENT GAME STATE
    ---------------------------
{game_state}

Your role is {role}

Thinking Process Guidelines

1. Parse and Organize Information
   Extract all given facts, constraints, and
       rules from the current
   game state. Identify what roles other
       players might have, what
   strategies they might pursue.

2. Choose Your Strategy
   Depending on the role and the game state,
       choose your strategy.
   Use tools of cooperative game theory, think
       about good strategies
   for your own team.

3. Work Systematically
   Create tables, grids, or diagrams when
       helpful. Track
   possibilities and eliminations clearly.
       Apply constraints
   methodically, not randomly. Double-check
       each logical step.

4. Verify Your Solution
   Ensure your answer satisfies all given
       constraints. Check that
   your reasoning contains no logical gaps.
       Consider if there could
   be multiple valid solutions. Test edge
       cases if applicable.

5. Key Logical Principles to Remember
   - If A implies B, and A is true, then B is
       true (Modus Ponens)
   - If A implies B, and B is false, then A is
       false
     (Modus Tollens)
   - Proof by contradiction: Assume the
       opposite and derive a
     contradiction
   - Process of elimination: If all but one
       possibility is ruled
     out, the remaining must be true

Common Game Theory Solving Strategy Types and
    Approaches

   - Syllogisms: Convert to standard logical
       form, check validity
     vs. soundness
   - Mathematical Logic: Look for patterns,
       use algebraic thinking
     when appropriate
   - Probability assessment: Use probability
       and conditional
     probability where appropriate
   - Spatial/Temporal Reasoning: Draw diagrams
       , use relative
     positioning

Remember: Show your work clearly, state
    assumptions explicitly,
always verify your final answer against all
    constraints, and never
skip logical steps. Your thinking should be
    thorough enough that
another person could follow your reasoning
    completely.

Important Notes
```

```
- Do not generate or write any code unless
    explicitly requested
- Focus on pure logical reasoning and
    deduction
- If given example problems, follow their
    format but solve the
  specific problem presented, not the examples
- Be precise with language - distinguish
    between "must be true,"
  "could be true," and "cannot be true"

Remember that you are player {player_name},
    your role is {role}.
Only the last announcement round is about the
    current cards that the players hold now.
The previous announcement rounds are only
    useful to establish the trust-worthyness
    of a player's announcement.
Your task is to chose the next player. Just
    return the chosen player as result between
     the <answer> tags.
If the player name is just a numeric ID (0, 1,
     2, etc.) return the target player as just
     their ID as string, not as 'Player X' or
    similar name additions."
Provide your final player_choice decisions as
    json string, in JSON format with one key '
    target_player' where the respective value
    should be your chosen player ID. This JSON
     should be between the <answer> tags. All
    your thinking and strategic reasoning has
    to be between the <thinking> tags.
Output both your reasoning between the <
    thinking> tags and your answer between the
     <answer> tags.
```

For announcements, the following replaces the player choice in the given prompt template:

```
Remember that you are player {player_name},
    your role is {role}.
Your task is to make the new announcement
    about how many gold cards
and how many trap cards you have now in your
    current private cards.
All your thinking and strategic reasoning has
    to be between the
<thinking> tags.
Provide your final announcement decisions as
    json string, in JSON
format with two keys 'gold_cards' and '
    trap_cards' where the
respective values should be your announced
    gold and trap cards.
This JSON should be between the <answer> tags.
Output both, your reasoning between the <
    thinking> tags and your
answer between the <answer> tags.
```