LLM-Coordination: Evaluating and Analyzing Multi-agent Coordination Abilities in Large Language Models

Saaket Agashe, Yue Fan, Anthony Reyna, Xin Eric Wang
University of California, Santa Cruz
{saagashe, yfan71, ancreyna, xwang366}@ucsc.edu

Abstract

Large Language Models (LLMs) have demonstrated emergent common-sense reasoning and Theory of Mind (ToM) capabilities, making them promising candidates for developing coordination agents. This study introduces the LLM-Coordination Benchmark, a novel benchmark for analyzing LLMs in the context of **Pure Coordination** Settings, where agents must cooperate to maximize gains. Our benchmark evaluates LLMs through two distinct tasks. The first is Agentic Coordination, where LLMs act as proactive participants in four pure coordination games. The second is Coordination Question Answering (CoordQA), which tests LLMs on 198 multiple-choice questions across these games to evaluate three key abilities: Environment Comprehension, ToM Reasoning, and Joint Planning. Results from Agentic Coordination experiments reveal that LLM-Agents excel in multi-agent coordination settings where decision-making primarily relies on environmental variables but face challenges in scenarios requiring active consideration of partners' beliefs and intentions. The CoordQA experiments further highlight significant room for improvement in LLMs' Theory of Mind reasoning and joint planning capabilities. Zero-Shot Coordination (ZSC) experiments in the Agentic Coordination setting demonstrate that LLM agents, unlike RL methods, exhibit robustness to unseen partners. These findings indicate the potential of LLMs as Agents in pure coordination setups and underscore areas for improvement.

1 Introduction

In a wide range of activities, from daily tasks such as cooking to critical operations like rescue efforts, cooperation without mixed intentions is essential. These scenarios are examples of Pure Coordination Games, where all involved parties benefit from choosing strategies that are perfectly aligned, avoiding any conflict of interest. These games require agents to reason about their environment and plan while considering the beliefs and intentions of their partners. Recently, Large Language Models (LLMs) have demonstrated emergent planning abilities in both physical and virtual settings (Raman et al., 2022; Wang et al., 2023a; Wu et al., 2023), impressive reasoning capabilities (Wei et al., 2022), and the hints of a Theory of Mind (Kosinski, 2023) making them promising candidates for developing coordination agents. Previous works have explored the use of LLMs for developing collaborative agents, yet the requisite conditions, strengths, and limitations of LLMs in coordination games remain unclear. In this study, we intend to bridge the gap by performing a comprehensive evaluation and analysis of the multi-agent coordination abilities of LLMs.

Therefore, we introduce the **LLM-Coordination Benchmark** featuring two task settings for pure coordination games: 1. Agentic Coordination and 2. CoordinationQA. In Agentic Coordination, LLMs are scaffolded with components that allow them to act within actual game environments, providing a holistic evaluation of the competencies of LLMs to act as coordination agents. In CoordinationQA, LLMs have to answer a curated set of questions about edge-case scenarios drawn from coordination games where agents need to actively cooperate with their partners. The benchmark includes four collaborative games, providing a comprehensive analysis platform. Unlike studies on multi-LLM frameworks (Hong et al., 2023; Qian et al., 2024; Li et al., 2023a), which focus on orchestrating multiple LLMs to solve tasks, our benchmark assesses the innate ability of individual LLMs to understand and act within pure coordination scenarios where cooperation is essential.

Our experiments in the Agentic Coordination setting reveal that Large Language Models are com-

Findings of the Association for Computational Linguistics: NAACL 2025, pages 8053–8072
April 29 - May 4, 2025 ©2025 Association for Computational Linguistics

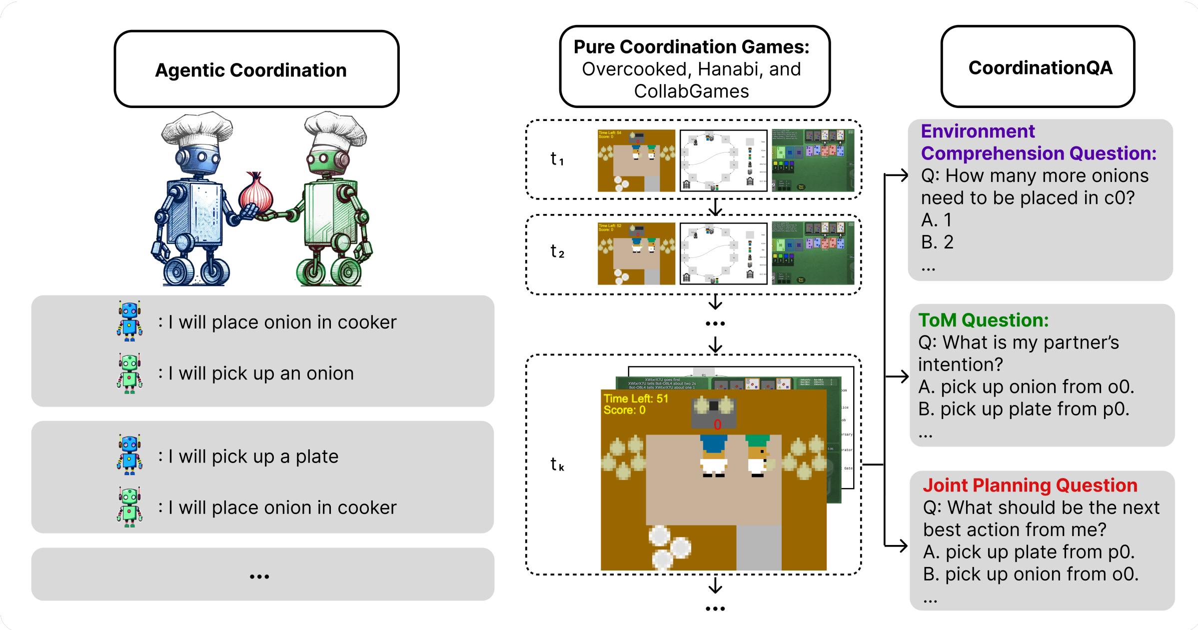

Figure 1: **The LLM Coordination Benchmark** consists of two tasks: *Agentic Coordination* to study the holistic abilities of LLMs in multi-turn pure coordination games, and *Coordination QA* to perform a fine-grained analysis of the Environment Comprehension, Theory of Mind Reasoning, and Joint Planning abilities of LLMs in the context of pure coordination scenarios.

petent at understanding the game objectives, generating coherent reasoning for their next actions, and coordinating with partners across all coordination games. They exhibit these behaviors without any training, fine-tuning, or few-shot examples. A comparative analysis reveals that LLM agents match or outperform RL baselines in games where optimal decision-making can be done by observing environment variables and positions (e.g., Overcooked). However, they struggle in settings where agents need to actively consider their partner's beliefs and intentions (e.g., Hanabi). We also observe that LLM agents are capable of collaborating with new partners, unlike self-play MARL methods (Carroll et al., 2019a; Bard et al., 2020) that fail to adapt to unseen agents.

For a more nuanced analysis of the coordination abilities of LLMs, we create the CoordinationQA Suite. This suite is designed to dissect the capabilities of LLMs in single-turn reasoning within coordination games, focusing on three key areas: Joint Planning, Theory of Mind (ToM), and Environment Comprehension. Joint Planning (JP) evaluates LLMs' planning abilities for optimal coordination, ToM questions probe their understanding of partner agents' intentions and needs, and Environment Comprehension (EC) assesses their inference of environment details, rules and objectives.

First, Our findings on CoordinationQA show a marked performance gap between GPT-4-turbo and other LLMs across three question types. Secondly, LLMs are most proficient in Environment Comprehension, indicating they understand the rules and environment states well. However, they face significant challenges in Theory of Mind Reasoning, with difficulty inferring others' intentions and needs. This issue worsens in Joint Planning, where most LLMs underperform, some even worse than random choices. These results highlight LLMs' limited reliability and effectiveness as coordination partners. Correlation analysis between LLMs' performance on CoordinationQA and their performance on agentic coordination setting further highlights their strengths in environmental reasoning but exposes significant weaknesses in Theory of Mind inference and Joint Planning capabilities.

In summary, our contributions are threefold:

1. We introduce the LLM-Coordination Benchmark for evaluating and analyzing LLMs in Pure Coordination Games, covering multi-turn Agentic Coordination and single-turn Coordination QA tasks.

2. We perform a holistic evaluation of LLM agents in Self-play and Cross-play settings, offering a detailed comparison with RL baselines

and showcasing their potential as Coordination Agents.

3. We investigate Environment Comprehension, Theory of Mind Reasoning, and Joint Planning as essential components of LLMs' overall coordination capabilities, highlighting their critical importance in pure coordination setups.

2 Related Work

Multi-agent Coordination. Pure Coordination games in game theory are scenarios where agents share the payoffs, and cooperation is the optimal strategy. Benchmarks like the Multiparticle Environment (Lowe et al., 2017), Overcooked-AI (Carroll et al., 2019a), and the Hanabi Challenge (Bard et al., 2020) evaluate multi-agent coordination. Carroll et al. (2019a) highlighted the importance of human data for effective Human-AI collaboration. Subsequent Overcooked-AI research focuses on aligning self-play-trained agents with humans using techniques such as self-play with past checkpoints (Strouse et al., 2021), population entropy objectives (Zhao et al., 2023), graph-theoretic objectives (Li et al., 2023c), policy ensembles (Lou et al., 2023), and integrating human biases (Yu et al., 2023). In the Hanabi Challenge, efforts aim to learn grounded policies over arbitrary conventions (Hu et al., 2021b,a). While most solutions enhance RL methods for coordination, we propose that LLMs offer an alternative due to their emergent reasoning and theory-of-mind-like abilities, avoiding arbitrary joint interactions.

Planning and Reasoning with Large Language Models. LLMs have shown remarkable natural language reasoning abilities (OpenAI, 2023; Ouyang et al., 2022; Chiang et al., 2023), achieving state-of-the-art results in verbal reasoning tasks. Augmented with components like memory and tools, LLMs can interact with external environments, solving long-horizon tasks and playing complex games (Wu et al., 2023; Wang et al., 2023a; Liang et al., 2022; Song et al., 2022). Guided by Cognitive Architectures for Language Agents (Sumers et al., 2023) as a design principle for agent design, we experiment with advanced reasoning strategies such as ReAct (Yao et al., 2023), Self-Verification (Weng et al., 2023), and Self-Consistency (Wang et al., 2023b) to enhance LLM reasoning. These strategies establish strong baseline performance for our Language Agent implementations.

Multi-agent LLMs. Recent studies have explored LLMs in multi-agent cooperation settings. Zhang et al. (2023b) developed a modular agent framework for spatial rearrangement tasks. Zhang et al. (2023a) introduced an architecture enabling LLMs to play Overcooked-AI. Shi et al. (2023) demonstrated positive zero-shot coordination in Avalon using code-driven reasoning. Li et al. (2023d) showed emergent collaborative abilities of LLMs in simulations, while Li et al. (2023b) investigated theory-of-mind inference using explicit belief representations. Xu et al. (2024) perform an analysis of LLMs in communication games Xu et al. (2023) analyzed LLM cognitive abilities through games, highlighting benefits of probabilistic modeling. In contrast, our research rigorously evaluates LLM agents' coordination abilities in established pure coordination games, where coordination is essential. Our setting only includes scenarios where agents must fully cooperate with each other with no competitive incentives. We also conduct a fine-grained, component-level analysis to understand the intricacies of LLMs' coordination capabilities.

3 LLM-Coordination Benchmark

3.1 Multi-turn Agentic Coordination

In the Multi-turn Agentic Coordination task, LLMs participate in end-to-end pure coordination games as agents, where the best strategy for all participating agents is to cooperate. We only consider pure coordination scenarios with no competitive incentives. LLMs under test are plugged into coordination frameworks with memory and the ability to act in complete games. These **LLM agents** can then be partnered with any policies or agents to complete the games.

Our LLM-Coordination benchmark includes 4 pure coordination games: Hanabi Challenge (Bard et al., 2020), Overcooked-AI (Carroll et al., 2019a), and Collab Capture and Collab Escape (inspired by the Pursuit-Evasion problem). These games were carefully selected for their ability to isolate and highlight specific coordination challenges, providing controlled environments that allow the analysis of pure coordination scenarios without too much emphasis on other reasoning challenges. While LLMs are versatile and capable of addressing a wide range of tasks, these well-studied settings offer established benchmarks, clear metrics, and reproducible scenarios that are particularly suited for examining coordination abilities of participating agents.

Hanabi Challenge. In Hanabi (Bard et al., 2020), players aim to assemble five sequences of cards in ascending order (1 through 5), each sequence dedicated to a different color: purple, red, blue, yellow, and green. A unique aspect of the game is that the players can only view their partner's cards, not their own. This requires players to work collaboratively, utilizing reveal tokens to provide hints about the cards in their partner's hand. These hints can be about either the color or the rank of the cards. For instance, using a single reveal token, a player can indicate all cards of a certain rank in their partner's hand. Once a player has an idea about which card they have, they can choose to play the card on the stack. If the card is correct, they get a point. Otherwise, players lose a collective life token. The loss of all 3 life tokens leads to the end of the game. Hanabi serves as an exemplary Pure Coordination game, necessitating player cooperation to achieve optimal outcomes. Success in Hanabi hinges on the ability to understand partners' perspectives, navigate decisions based on incomplete information, and engage in implicit communication, making it an excellent testing ground for coordination among agents.

Overcooked-AI. In the Overcooked-AI environment (Carroll et al., 2019a), two agents—Alice (Blue) and Bob (Green)—collaborate to cook and deliver onion soups. This environment includes a variety of layouts, each with its own arrangement and quantity of onion dispensers, plate dispensers, cookers, delivery zones, and countertops. To prepare a dish, agents are required to insert three onions into a cooker, initiating a cooking process that lasts 20 time steps. Upon completion, the soup must be plated and delivered to complete the task. Each layout presents unique challenges, emphasizing the need for agents to comprehend their surroundings, locate necessary resources, and synchronize their actions with their teammate for effective collaboration.

Collab Capture. Collab Capture involves two agents trying to capture an adversary in a maze of interconnected rooms. The rooms are connected by doors, which can be controlled through access buttons that can be found in other rooms. The agents' task is to capture the adversary in the least amount of time using effective strategies. To evaluate different coordination strategies between agents, we design four scenarios by controlling the states of the doors (open or closed) within the layout shown in figure 5. These scenarios highlight various coordination challenges, such as trapping the adversary through precise positioning, enabling a teammate by prioritizing door control over direct pursuit, and strategically restricting the adversary's movement to facilitate capture. (see Appendix E.1 for more details.)

Collab Escape. Collab Escape involves two agents trying to escape an adversary in a maze of interconnected rooms. They need to fix two generators (similar to the game Dead-by-Daylight (Dea, 2016)) located in different rooms to open an exit portal. The adversary tries to catch the agents, and the win condition is any one agent escaping. To evaluate coordination strategies in Collab Escape, we develop two scenarios by varying the initial proximity of agents to the adversary and generators in the layout shown in figure 6. Depending on their proximity to the adversary/generators, players need to apply strategies such as luring the adversary away from the partner, choosing to continue fixing the generators while sacrificing for the partner's safety and manipulating the movement of the adversary (see Appendix E.2 for more details.)

3.2 Single-turn Coordination QA

The agentic coordination task paints a holistic picture of the abilities of LLMs as agents. To dive deeper into the specific strengths and weaknesses of LLMs, we develop the CoordinationQA Suite. Inspired by the idea of **Unit Testing** for evaluating AI agents (Knott et al., 2021), we manually sampled edge cases from all 4 pure coordination games mentioned in Section 3.1. All of these edge cases necessitate agents to actively understand their current state, think about their partner's intentions, and come up with the best plans for coordination. We then create a set of three types of questions for each scenario in our CoordinationQA Suite.

- **Environment Comprehension (EC)** questions require LLMs to make indirect inferences about some aspect of their environment (See Appendix D.1). The questions cover details of the layouts, implications of current observations, and counts of artifacts.
- **Theory of Mind Reasoning (ToM)** questions challenge the LLMs to predict the intentions of their partners and probe about the requirements of their partners (See Appendix D.2).

- **Joint Planning (JP)** questions provide agents with the state/observation and ask them to predict the best next action for effective coordination. This question is essentially the same question that LLMs need to repeatedly solve when they act as agents (See Appendix D.3).

All the questions were manually developed and labeled. We filtered out questions and scenarios that showed any ambiguity, leaving only questions that had clear, optimal solutions. We generated a total of N=66 scenarios (25 from Overcooked, 28 from Hanabi, and 13 from the two Collab Games) and created 3 questions per scenario, resulting in 198 unique questions. The right side of Figure 1 demonstrates the sampling process for the three types of questions with an example from the game Overcooked. The selected scenario shows the Blue agent about to place their third onion in the cooker, and the green agent needs to figure out what to do next. See Appendix D for examples of questions and the templates used to formulate these questions.

4 Experimental Setup

4.1 Agentic Coordination

We perform two types of experiments in agentic coordination: Self-Play and Cross-Play. In self-play settings, the participating agents are of the same type. In Cross-Play experiments, we pair agents with unseen partners, and they need to adapt their behavior to the actions of these new partners.

4.1.1 LLM Agents

To allow LLMs to play multi-turn games, we scaffold them with an agentic framework based on Cognitive Architectures for Language Agents Sumers et al. (2023). The framework includes three parts: Memory, Reasoning, and Grounding.

Memory includes (1) Long-Term Memory for storing the Game Description, including the game's rules, conventions, objectives, and action space, (2) Working memory, which consists of a textual description of the current observation, and (3) Episodic Memory which is a list of previous actions selected by the agent.

Reasoning is where the Large Language Model (LLM) is plugged into the framework. It takes the textual description from the Memory as input and generates the next action based on the context. The LLM reasons about the current state and then selects an action from the list of available actions in natural language.

<u>Self-Verification:</u> For the coordination game Hanabi, there is a low margin for error as any misplays lead to the loss of life tokens, and the loss of all three life tokens subsequently results at the end of the game. We thus supplement the reasoning process in Hanabi with Answer-Verification (Weng et al., 2023), where the LLM is re-prompted to confirm that the action it generated is appropriate and does not lead to fatal errors.

<u>ToM-Reasoning:</u> We also demonstrate the positive impact of a Theory of Mind Reasoning step prior to generating the next action for Hanabi and CollabEscape, which benefit from this intermediate step. In the ToM reasoning step, the LLM generates an interpretation of their partner's actions or current position before generating the next action to explicitly capture the belief inference process. We do not test with additional ToM reasoning on Overcooked due to significant latency and cost constraints, with marginal benefits.

Finally, the **Grounding** process translates the natural language action generated by the reasoning module into game-compatible action(s). The exact implementation of the grounding module depends on the game in question; for example, in Overcooked-AI, the grounding module needs to convert high-level actions like "pick up onion from o0." into sequences of lower-level actions. On the other hand, in games like Hanabi, the Grounding needs to match actions like "Reveal Bob's Red Color Cards" to their lower-level representations. The Grounding process is also responsible for filtering out infeasible actions based on the context of the game (See Appendices A, B for more details.)

There are no prompt or setup differences for LLM Agents based on Cross-play or Self-play. We use the LLMs gpt-4-0125-preview, GPT-3.5-turbo-0125, Mixtral 8x7B, and GPT-4o for agentic evaluation studies.

4.1.2 MARL Agents

Self-play MARL Baselines: For Overcooked we use Proximal Policy Optimization (Schulman et al., 2017) and Population-Based Training (Jaderberg et al., 2017) as baselines for comparison. These baselines were established by Carroll et al. (2019a).

For the Hanabi challenge, we use Bayesian Action Decoder (BAD) (Bard et al., 2020), Simplified Action Decoder (SAD) (Hu and Foerster, 2021), and Off-Belief Learning (Hu et al., 2021a) as MARL baselines for Hanabi. All three baselines

Agent	CR	AA	Ring	FC	CC
Self-Play (PPO)	198.8 ± 4.06	167.2 ± 3.63	$\mathbf{190.8 \pm 4.25}$	151.9 ± 3.28	122.3 ± 3.80
PBT	$\mathbf{216.9 \pm 1.31}$	190.1 ± 8.64	173.8 ± 18.27	169.5 ± 10.09	140.1 ± 13.86
GPT-3.5-turbo	33.3 ± 10.88	46.6 ± 10.88	40.0 ± 0.00	66.6 ± 14.40	53.3 ± 5.44
Mixtral8x7B	46.6 ± 14.40	200.0 ± 9.42	113.3 ± 5.44	46.6 ± 14.40	100.0 ± 9.42
GPT-4o	160 ± 0.00	166.66 ± 5.44	66.66 ± 21.77	120.0 ± 9.42	$\mathbf{160.0 \pm 0.00}$
GPT-4-turbo	173.3 ± 6.67	$\mathbf{260.0 \pm 11.55}$	140.0 ± 0.00	$\mathbf{180.0 \pm 11.55}$	$\mathbf{160.0 \pm 0.00}$

Table 1: Performance comparison across Multi-Agent Reinforcement Learning (MARL) and LLM-agent methods. Scores indicate the best performance in each category. The GPT-4-turbo Agent demonstrates superior coordination in 3 out of 5 scenarios, underscoring advanced reasoning capabilities in coordination tasks. The five layouts are CR: Cramped Room, AA: Asymmetric Advantages, Ring: Coordination Ring, FC: Forced Coordination, and CC: Counter Circuit. For visualization and details of these layouts, see appendix A

	Collab Escape		**Collab Capture**	
Agent	Capture Rate	Avg. Turns	Escape Rate	Avg. Turns
GPT-4-turbo	0.83	4.60	1.00	3.5
– (w/out ToM Reasoning)	0.50	2.00	1.00	4.75
GPT-4o	0.67	4.00	1.00	7.17
GPT-3.5-turbo	0.33	2.50	0.67	8.38
Mixtral-8x7b	0.50	7.67	0.92	7.55
Greedy Baseline	0.00	N.A.	0.50	6.00

Table 2: Comparison of different LLM Agents on CollabCapture and CollabEscape. In CollabEscape, two agents work together to escape from an adversary. In CollabCapture, two agents coordinate to capture an adversary. The reported results are run across 3 trials each for various layout configurations (Detailed in Appendix E). The table also demonstrates the impact of the explicit ToM reasoning step in both game setups.

achieve near-perfect performance in Self-play.

Cross-play MARL Baselines: For Overcooked, we use a Behavior Cloning model trained on human data (Carroll et al., 2019a) and a Proximal Policy Optimization (PPO) agent trained with the Human Behavior Cloning agent (Carroll et al., 2019a) as baselines for comparison. We also report Hidden-Utility Self-play (HSP) (Yu et al., 2023) as a baseline. We use human proxies based on behavior cloning as unseen partners.

For Hanabi, we use the Simplified Action Decoder (SAD), which is trained through self-play as a baseline. We pair our agents with Off-Belief Learning (Hu et al., 2021a), which was trained to generate grounded policies and adapt to unseen partner agents.

Metrics. We measure the total score achieved by agents in Overcooked, where each delivery provides 20 points to both agents. In the case of Hanabi, the metric is the total number of cards that have been correctly arranged by the players. For CollabEscape and CollabCapture, we report the success rate of escape or capture across multiple trials and the average turns to capture or escape.

4.2 CoordinationQA

We assess the performance of 5 Families of Large Language Models (LLMs) (Jiang et al., 2023, 2024; Touvron et al., 2023; Chiang et al., 2023; OpenAI, 2023) across three dimensions: Environment Comprehension (EC), Theory of Mind Reasoning (ToM), and Joint Planning (JP). For each category, LLMs respond to multiple-choice questions (MCQs), with their responses evaluated against ground-truth answers through fuzzy string matching. To account for the variability in LLM responses, we conduct three trials per model. All models being tested are shown the same prompts. We also report a Random baseline.

5 Discussion

Zero-shot LLM Agents match or surpass trained RL methods in Environment-focused Coordination Problems. We observed that LLM agents (w. GPT-4-turbo) outperform or match the overall performance of RL methods across all layouts of Overcooked-AI. Table 1 presents the numerical scores attained by different agents when paired with a partner agent of the same type. This

Agent	Score
Bayesian Action Decoder	23.92 ± 0.01
Simplified Action Decoder	24.01 ± 0.01
Off-Belief Learning	24.10 ± 0.01
GPT-4-turbo	$\mathbf{13.33} \pm 0.88$
- (w.o ToM Reasoning)	10.33 ± 0.88
- (w.o ToM Reasoning & Verif.)	4.33 ± 0.88
GPT-4o	8.33 ± 1.20
GPT-3.5-turbo	1.33 ± 0.72
Mixtral-8x7b	0.33 ± 0.27

Table 3: Agentic performance comparison on Hanabi Challenge. RL methods are very strong and obtain near-perfect scores. The best GPT-4-turbo-based LLM Agent is much weaker compared to RL baselines. Removing the ToM reasoning and Verification steps from the LLM agent leads to further performance degradation

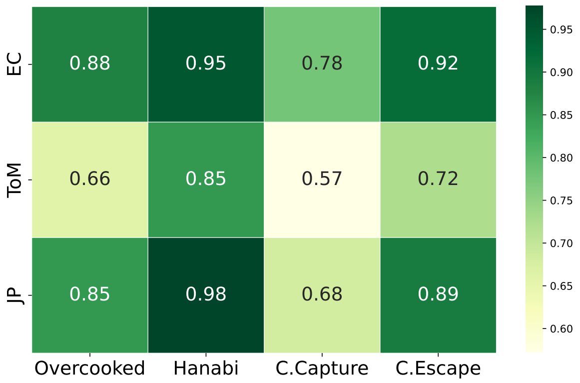

Figure 2: Correlation of LLM Agent performance in Agentic Coordination setup on all four games *vs.* performance on the CoordinationQA benchmark.

implies that LLM agents match RL agents that have been explicitly trained through Self-play without any game-specific training or fine-tuning. However, it is important to note that LLM agents are significantly slower and larger than RL models, making them unsuitable for real-time use at present. We also see positive results on the CollabCapture and CollabEscape games, with most LLMs being able to complete both challenges (see Table 2).

LLM agents struggle at effective planning when advanced Theory of Mind reasoning is required. In Hanabi Challenge, LLM agents seem to struggle compared to RL methods (see Table 3). GPT-4-turbo performs reasonably well, while other LLMs can barely complete the games. We attribute this failure to two factors. First, there is little room for errors in Hanabi. Any misplay leads to the loss of a life token. Second, Hanabi requires more complex Theory of Mind Reasoning compared to the Overcooked-AI environment. Each action requires agents to actively consider their partner's beliefs, intentions, and how they would react to implicit communication. In contrast, Overcooked is fully observable, and its action space consists of actions like *pick up an onion from onion_dispenser_0* and *place onion in cooker_0*. Under most scenarios and layouts, LLMs only need to consider the next best steps based on the state of the environment.

We use correlation study to provide additional validation to these findings. We calculate the Pearson Correlation Coefficient (r) of the performance of the four LLMs (GPT-4-turbo, GPT-4o, GPT-3.5-turbo, and Mixtral 8x7b) on Agentic Coordination setup (Average score per game) *vs.* the score on CoordinationQA task. Figure 2 shows a high correlation between Environment Comprehension capabilities and Success at Overcooked, but a more moderate correlation between Theory of Mind Reasoning capabilities and Success at Overcooked. Conversely, in Hanabi, high success is strongly correlated with both Environment Comprehension and Theory of Mind Reasoning Abilities. In CollabEscape we see a higher correlation with ToM reasoning abilities compared to CollabCapture. This correlation study also connects and establishes a positive alignment between LLM-agent performance on the multi-turn agentic task and the single-turn CoordinationQA.

Auxiliary reasoning strategies like Verification and ToM reasoning help LLMs reason for coordination. Adding an Answer Verification step significantly reduces fatal mistakes (wrong card plays) caused by LLM hallucinations. Without the support of the Verification step, LLM agents bomb (lose all three lives) before the end of the game in every trial. The ToM reasoning step separates the tasks of interpreting partner clues and generating actions, allowing the LLM to better synthesize available information for action planning. Table 3 shows the impact of ablating the verification and ToM reasoning steps from the LLM Agent. The ToM reasoning step is also useful in the CollabEscape (see table 2) game, as players need to actively consider what their partner needs and act sacrificially if needed. In CollabCapture, it shows a relatively low benefit since agents can observe the positions of all agents as well as doors on the map and infer the correct action based on this envi-

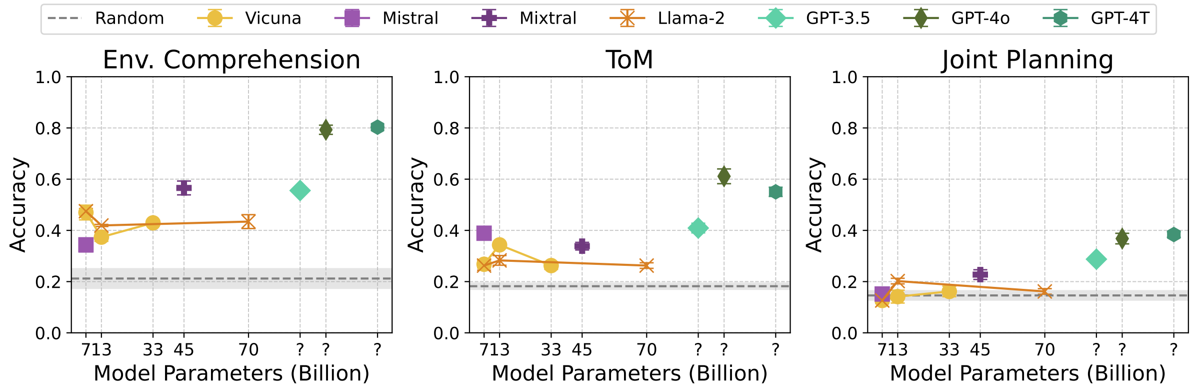

Figure 3: The performance of different LLMs on *CoordinationQA*, which provides a fine-grained analysis of LLMs' Environment Comprehension, Theory of Mind Reasoning, and Joint Planning abilities within pure coordination scenarios.

Method	CR	AA	Ring	FC	CC
BC	103.5 | 110.0	136.5 | 137.5	59.0 | 70.0	20.5 | 31.0	38.0 | 44.0
PPO_{BC}	156.4 | **163.9**	72.6 | 178.8	126.4 | 129.8	58.9 | 76.9	69.5 | 57.6
HSP	-	**300.3** | **217.1**	160.0 | **160.6**	-	107.4 | **106.6**
GPT-4-turbo[1]	**160.0** | 160.0	180.0 | 200.0	**160.0** | 140.0	**120.0** | **80.0**	**140.0** | 100.0

Table 4: Zero-shot coordination results of AI-Human Proxy Gameplay. We compare Behavior Cloning (BC), PPO_BC, HSP (Yu et al., 2023), and GPT-4-turbo agent. The LLM agent outperforms the PPO and BC methods and matches the HSP (Yu et al., 2023) baseline in most cases, demonstrating robustness to unseen partner agents. Since the two agents in Overcooked-AI might be tasked with different roles based on their starting locations, we show results playing from either side separated by |.

ronmental context.

Comparative Results of LLMs in Environment Comprehension, ToM Reasoning, and Joint Planning. In Figure 3, we see that most LLMs achieve their best results on the Environment Comprehension question. The best performing LLM GPT-4-turbo gets more than 80% Environment Comprehension Questions correct. The overall performance across LLMs drops on the more challenging Theory of Mind reasoning questions. Both GPT-4-turbo and GPT-4o do well on the Theory of Mind reasoning questions. The overall accuracy of LLMs on Joint Planning questions is still significantly weak, with even the best LLM scoring less than 40%, indicating a large room for improvement in LLMs' ability to perform coordination reasoning. Another cause for concern is that open-source LLMs perform abysmally at Joint Planning, with some models performing worse than random.

LLM Agents are robust to unseen partners. We use Overcooked-AI and the Hanabi challenge as testbeds to evaluate the performance of LLM agents when paired with unseen agents. This task is popularly known as *Zero Shot Coordination*. In Overcooked-AI, we pair our LLM agents as well as baselines with proxy-human agents. These proxy human agents are behavior cloning agents trained using human data by Carroll et al. (2019b). As shown in Table 4, we discover that LLM agents outperform both Behavior Cloning as well as PPO agents trained with human data. Apart from the Asymmetric Advantages layout, they also match or outperform the Hidden Utility Self-play (HSP) baseline, which is designed to excel at ZSC.

In Hanabi, we pair our agents with Off-Belief Learning (OBL) agents (Hu et al., 2021a). OBL is a MARL strategy that generates grounded clues and actions and is the state-of-the-art method for cross-play in Hanabi. OBL agents provide observation-grounded clues and collaborate well with humans. Therefore, we use them as unseen partners in our experiments. Table 5 shows that the GPT-4-turbo agent scores an average of 15 points with the OBL-1 agent compared to their self-play scores of 13.66, indicating no degradation in performance

[1]For GPT-4-turbo, we run a single trial from either position due to cost and time constraints.

Method	Self-Play	Cross-Play w/ OBL-1	Cross-Play w/ OBL-4
SAD	$\mathbf{23.66} \pm 0.54$	11.33 ± 4.00	8.00 ± 0.47
GPT-4-turbo	13.66 ± 0.27	$\mathbf{15.00} \pm 2.94$	$\mathbf{12.00} \pm 0.94$

Table 5: Cross-Play results of RL agent (SAD) and LLM agent (GPT-4-turbo). All agents play three games with different seeds (same seeds across agents). SAD performs really well at self-play but suffers significant performance degradation with new partners OBL-1 and OBL-4. LLM Agents coordinate well with the new, unseen partners.

with a new partner. The baseline RL method, Simplified Action Decoder (SAD) (Hu and Foerster, 2021), fails critically when paired with unseen OBL agents, even though it excels at self-play (22.00 points) due to self-play training.

6 Conclusion

In this study, we evaluated and analyzed the current large language models in the context of pure coordination games. We introduced the LLM-Coordination benchmark with its two tasks: 1. Agentic Coordination and 2. CoordinationQA. These settings allowed us to conduct holistic comparative studies of LLMs as agents and dive deeper into the fine-grained aspects of LLMs as coordination reasoners. We juxtaposed LLM agents with existing Multi-agent Reinforcement Learning agents, discussing the conditions in which LLMs thrive and fail. Finally, we discussed the Theory of Mind Reasoning, Environment Comprehension, and Joint Planning as prerequisites for coordination and evaluated existing LLMs on these components.

7 Acknowledgements

This research project has benefitted from the Microsoft Accelerate Foundation Models Research (AFMR) grant program.

8 Limitations

Latency and Compute Requirements: As highlighted in Section 5, effective reasoning for coordination is achievable primarily with larger LLMs like GPT-4-turbo. However, these models are associated with significant latency and require substantial computational resources, making them less suitable for real-time applications where rapid decision-making is crucial.

Initial Prompt Configuration: Achieving optimal reasoning performance necessitates careful manual configuration of the initial prompts that describe the game (Procedural Memory). While this prompt could be extracted from game manuals or existing resources, it still needs to be formatted and designed with the LLM agent in mind. Furthermore, the results for individual games could be improved by letting the LLM generate more text and engineering the prompt. However, we leave these optimizations to future works focused on performance improvement rather than benchmarking.

Manual Curation of Edge Cases: The CoordinationQA suite involves manually curating unambiguous edge cases in coordination games to construct the dataset. This can hinder the ability to scale the benchmark to accommodate new scenarios. Yet, it is important to curate these examples for more reliable studies.

References

2016. Dead by Daylight. https://deadbydaylight.com/en. Video game.

Nolan Bard, Jakob N. Foerster, Sarath Chandar, Neil Burch, Marc Lanctot, H. Francis Song, Emilio Parisotto, Vincent Dumoulin, Subhodeep Moitra, Edward Hughes, Iain Dunning, Shibl Mourad, Hugo Larochelle, Marc G. Bellemare, and Michael Bowling. 2020. The hanabi challenge: A new frontier for ai research. Artificial Intelligence, 280:103216.

Micah Carroll, Rohin Shah, Mark K. Ho, Thomas L. Griffiths, Sanjit A. Seshia, Pieter Abbeel, and Anca Dragan. 2019a. On the Utility of Learning about Humans for Human-AI Coordination. Curran Associates Inc., Red Hook, NY, USA.

Micah Carroll, Rohin Shah, Mark K. Ho, Thomas L. Griffiths, Sanjit A. Seshia, Pieter Abbeel, and Anca Dragan. 2019b. overcooked_ai. https://github.com/HumanCompatibleAI/overcooked_ai/tree/master.

Wei-Lin Chiang, Zhuohan Li, Zi Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E. Gonzalez, Ion Stoica, and Eric P. Xing. 2023. Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality.

Sirui Hong, Mingchen Zhuge, Jonathan Chen, Xiawu Zheng, Yuheng Cheng, Ceyao Zhang, Jinlin Wang, Zili Wang, Steven Ka Shing Yau, Zijuan Lin, Liyang Zhou, Chenyu Ran, Lingfeng Xiao, Chenglin Wu, and Jürgen Schmidhuber. 2023. Metagpt: Meta programming for a multi-agent collaborative framework. Preprint, arXiv:2308.00352.

Hengyuan Hu and Jakob N Foerster. 2021. Simplified action decoder for deep multi-agent reinforcement learning. Preprint, arXiv:1912.02288.

Hengyuan Hu, Adam Lerer, Brandon Cui, David Wu, Luis Pineda, Noam Brown, and Jakob Foerster. 2021a. Off-belief learning. Preprint, arXiv:2103.04000.

Hengyuan Hu, Adam Lerer, Alex Peysakhovich, and Jakob Foerster. 2021b. "other-play" for zero-shot coordination. Preprint, arXiv:2003.02979.

Max Jaderberg, Valentin Dalibard, Simon Osindero, Wojciech M. Czarnecki, Jeff Donahue, Ali Razavi, Oriol Vinyals, Tim Green, Iain Dunning, Karen Simonyan, Chrisantha Fernando, and Koray Kavukcuoglu. 2017. Population based training of neural networks. Preprint, arXiv:1711.09846.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2023. Mistral 7b. Preprint, arXiv:2310.06825.

Albert Q. Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, Gianna Lengyel, Guillaume Bour, Guillaume Lample, Lélio Renard Lavaud, Lucile Saulnier, Marie-Anne Lachaux, Pierre Stock, Sandeep Subramanian, Sophia Yang, Szymon Antoniak, Teven Le Scao, Théophile Gervet, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2024. Mixtral of experts. Preprint, arXiv:2401.04088.

Paul Knott, Micah Carroll, Sam Devlin, Kamil Ciosek, Katja Hofmann, A. D. Dragan, and Rohin Shah. 2021. Evaluating the robustness of collaborative agents. Preprint, arXiv:2101.05507.

Michal Kosinski. 2023. Theory of mind might have spontaneously emerged in large language models. Preprint, arXiv:2302.02083.

Guohao Li, Hasan Abed Al Kader Hammoud, Hani Itani, Dmitrii Khizbullin, and Bernard Ghanem. 2023a. Camel: Communicative agents for "mind" exploration of large language model society. Preprint, arXiv:2303.17760.

Huao Li, Yu Chong, Simon Stepputtis, Joseph Campbell, Dana Hughes, Charles Lewis, and Katia Sycara. 2023b. Theory of mind for multi-agent collaboration via large language models. In Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing. Association for Computational Linguistics.

Yang Li, Shao Zhang, Jichen Sun, Yali Du, Ying Wen, Xinbing Wang, and Wei Pan. 2023c. Cooperative open-ended learning framework for zero-shot coordination. In International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA, volume 202 of Proceedings of Machine Learning Research, pages 20470–20484. PMLR.

Yuan Li, Yixuan Zhang, and Lichao Sun. 2023d. Metaagents: Simulating interactions of human behaviors for llm-based task-oriented coordination via collaborative generative agents. Preprint, arXiv:2310.06500.

Jacky Liang, Wenlong Huang, Fei Xia, Peng Xu, Karol Hausman, Brian Ichter, Pete Florence, and Andy Zeng. 2022. Code as policies: Language model programs for embodied control. In arXiv preprint arXiv:2209.07753.

Xingzhou Lou, Jiaxian Guo, Junge Zhang, Jun Wang, Kaiqi Huang, and Yali Du. 2023. Pecan: Leveraging policy ensemble for context-aware zero-shot human-ai coordination. In Proceedings of the 2023 International Conference on Autonomous Agents and Multiagent Systems, AAMAS '23, page 679–688, Richland, SC. International Foundation for Autonomous Agents and Multiagent Systems.

Ryan Lowe, Yi Wu, Aviv Tamar, Jean Harb, Pieter Abbeel, and Igor Mordatch. 2017. Multi-agent actor-critic for mixed cooperative-competitive environments. In Proceedings of the 31st International Conference on Neural Information Processing Systems, NIPS'17, page 6382–6393, Red Hook, NY, USA. Curran Associates Inc.

OpenAI. 2023. Gpt-4 technical report. Preprint, arXiv:2303.08774.

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. 2022. Training language models to follow instructions with human feedback. Preprint, arXiv:2203.02155.

Chen Qian, Wei Liu, Hongzhang Liu, Nuo Chen, Yufan Dang, Jiahao Li, Cheng Yang, Weize Chen, Yusheng Su, Xin Cong, Juyuan Xu, Dahai Li, Zhiyuan Liu, and Maosong Sun. 2024. Chatdev: Communicative agents for software development. Preprint, arXiv:2307.07924.

Shreyas Sundara Raman, Vanya Cohen, Eric Rosen, Ifrah Idrees, David Paulius, and Stefanie Tellex. 2022. Planning with large language models via corrective re-prompting. Preprint, arXiv:2211.09935.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. Preprint, arXiv:1707.06347.

Zijing Shi, Meng Fang, Shunfeng Zheng, Shilong Deng, Ling Chen, and Yali Du. 2023. Cooperation on the fly: Exploring language agents for ad hoc teamwork in the avalon game. Preprint, arXiv:2312.17515.

Chan Hee Song, Jiaman Wu, Clayton Washington, Brian M Sadler, Wei-Lun Chao, and Yu Su. 2022. Llm-planner: Few-shot grounded planning for embodied agents with large language models. arXiv preprint arXiv:2212.04088.

DJ Strouse, Kevin McKee, Matt Botvinick, Edward Hughes, and Richard Everett. 2021. Collaborating with humans without human data. In Advances in Neural Information Processing Systems, volume 34, pages 14502–14515. Curran Associates, Inc.

Theodore R. Sumers, Shunyu Yao, Karthik Narasimhan, and Thomas L. Griffiths. 2023. Cognitive architectures for language agents. Preprint, arXiv:2309.02427.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. 2023. Llama 2: Open foundation and fine-tuned chat models. Preprint, arXiv:2307.09288.

Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. 2023a. Voyager: An open-ended embodied agent with large language models. Preprint, arXiv:2305.16291.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2023b. Self-consistency improves chain of thought reasoning in language models. Preprint, arXiv:2203.11171.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. Advances in Neural Information Processing Systems, 35:24824–24837.

Yixuan Weng, Minjun Zhu, Fei Xia, Bin Li, Shizhu He, Shengping Liu, Bin Sun, Kang Liu, and Jun Zhao. 2023. Large language models are better reasoners with self-verification. Preprint, arXiv:2212.09561.

Yue Wu, Shrimai Prabhumoye, So Yeon Min, Yonatan Bisk, Ruslan Salakhutdinov, Amos Azaria, Tom Mitchell, and Yuanzhi Li. 2023. Spring: Gpt-4 outperforms rl algorithms by studying papers and reasoning. Preprint, arXiv:2305.15486.

Lin Xu, Zhiyuan Hu, Daquan Zhou, Hongyu Ren, Zhen Dong, Kurt Keutzer, See Kiong Ng, and Jiashi Feng. 2023. Magic: Investigation of large language model powered multi-agent in cognition, adaptability, rationality and collaboration. Preprint, arXiv:2311.08562.

Yuzhuang Xu, Shuo Wang, Peng Li, Fuwen Luo, Xiaolong Wang, Weidong Liu, and Yang Liu. 2024. Exploring large language models for communication games: An empirical study on werewolf. Preprint, arXiv:2309.04658.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2023. React: Synergizing reasoning and acting in language models. Preprint, arXiv:2210.03629.

Chao Yu, Jiaxuan Gao, Weilin Liu, Botian Xu, Hao Tang, Jiaqi Yang, Yu Wang, and Yi Wu. 2023. Learning zero-shot cooperation with humans, assuming humans are biased. In The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023. OpenReview.net.

Ceyao Zhang, Kaijie Yang, Siyi Hu, Zihao Wang, Guanghe Li, Yihang Sun, Cheng Zhang, Zhaowei Zhang, Anji Liu, Song-Chun Zhu, Xiaojun Chang, Junge Zhang, Feng Yin, Yitao Liang, and Yaodong Yang. 2023a. Proagent: Building proactive cooperative ai with large language models. Preprint, arXiv:2308.11339.

Hongxin Zhang, Weihua Du, Jiaming Shan, Qinhong Zhou, Yilun Du, Joshua B. Tenenbaum, Tianmin Shu, and Chuang Gan. 2023b. Building cooperative embodied agents modularly with large language models. Preprint, arXiv:2307.02485.

Rui Zhao, Jinming Song, Yufeng Yuan, Haifeng Hu, Yang Gao, Yi Wu, Zhongqian Sun, and Wei Yang. 2023. Maximum entropy population-based training for zero-shot human-ai coordination. Proceedings of the AAAI Conference on Artificial Intelligence, 37(5):6145–6153.

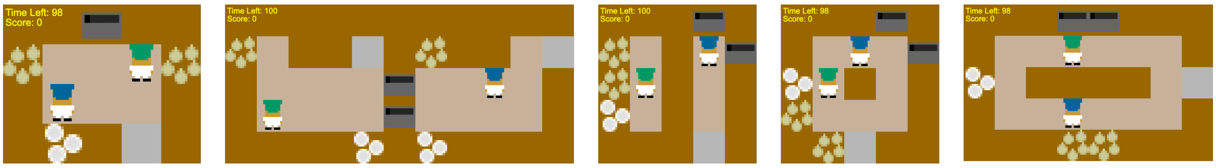

Figure 4: The Overcooked layouts from left to right: Cramped Room (CR), Asymmetric Advantages (AA), Forced Coordination (FC), Coordination Ring (CR), and Counter Circuit (CC).

A Overcooked Implementation Details

In the game Overcooked-AI, two chefs must coordinate their actions to cook and deliver onion soups. Each soup requires three onions as ingredients, which can be found in onion dispensers. The players must add three onions to a cooker to start cooking. Once the three onions are added, the soup starts cooking automatically and requires 20 time steps to complete. Once the soup is cooked, a player needs to pick up a plate, load the soup, and deliver it to the required delivery area. Players must coordinate their actions to effectively deliver soup. Depending on the layouts (see figure 4), players need to adapt to cramped spaces, understand environment layouts for role assignments, pass objects to each other, make way for one another, and find the most effective paths.

A.1 Game and Layout Description

We use a general game description G that explains the rules and objectives of overcooked. Since each layout has a different number of locations, like onion dispensers and cookers, we include a succinct description of each environment L_i, which includes how many instances of particular facilities there are. For environments that include partitions, we mention which partition each of the agents is situated in and what facilities that agents can access. In addition, we also mentioned the shape of the environment.

```
I am {self.player_names[self.player_id]}. I am playing the
game Overcooked with my partner {self.player_names[self.
      other_player_id]}.
{EnvDescriptions[self.layout_name]}
Overcooked has the following rules: {self.rules}.
We have agreed to follow the following conventions:
{self.conventions}. I\'ll provide my action history,
current state, teammate's status, and my possible actions.
Help me select the best action from the list.
Format your response as:
Explanation:<Brief explanation for my next action>.
Action: <action>.
Only select one action. Do not say anything else. Got it?
```

A.2 State Description

The State is represented in natural language $D(S)$ in the working memory, which can be processed by a Large Language Model (LLM). The state S includes variables that fully represent the necessary details of the layout as well as the players. The information provided in $D(S)$ is equivalent to what would be accessible to a Reinforcement Learning (RL) agent in the form of state representations. The following information is included in $D(S)$:

Objects Held by Each Player The state description $D(S)$ begins by detailing the inventories I_{α_1} and I_{α_2} of Alice and Bob, respectively. Each inventory I_{α_i}(where $i \in \{1, 2\}$) can contain one of the following items: {"onion", "plate", "cooked soup"}. This inventory information is translated into natural language and incorporated into $D(S)$ in the format: "I am holding I_{α_1}. Bob is holding I_{α_2}." Such information is vital for inferring the likely subsequent actions of the partner agent.

Location of the Agent Controlled by LLM: Given the limitations of Large Language Models (LLMs) in interpreting grid-based spatial information, we opt to provide processed location data to the LLM. For each agent P_i (where $i \in \{1, 2\}$), and for each location of interest denoted as loc, we calculate the distance $d_{(P_i,\text{loc})}$ as the number of steps required to reach loc from P_i using the shortest available path. The state description $D(S)$ then includes this processed location information in the format: "loc is $d_{(P_i,\text{loc})}$ units away." Here, loc can represent various points of interest such as onion dispensers, plate dispensers, cookers, delivery areas, kitchen counters, or shared counters. If a location is either inaccessible or blocked by another agent, this is explicitly stated in $D(S)$. For example, if a location is blocked by Bob, it would be stated as "loc is blocked by Bob." To distinguish between the location information relevant to each agent, $D(S)$ prefixes the respective sections with "Your location information:" for the agent controlled by the LLM and "Bob's location information:" for the partner agent.

Cooker Information The state description $D(S)$ also incorporates information about the cooker, which is central to the gameplay strategy. Specifically, for each cooker i, $D(S)$ includes the number of onions n_{i} currently in the pot. Additionally, $D(S)$ provides the operational state of the cooker, denoted as CookerState_i, which can be either "Off" or "On". Lastly, the current condition of the soup in the cooker is represented by SoupState_i, which can take one of the following values: "Cooking", "Cooked", or "Not Started". Thus, the information for cooker c_i is formatted as: "c_i has n_{i} onions. c_i is CookerState_i. Soup in c_i is SoupState_i."

Kitchen Counter Information The state description $D(S)$ includes information about kitchen counters, which are primarily used for temporary object storage. Specifically, $D(S)$ identifies the closest empty kitchen counter k_{empty} and the set K_{filled} of all counters currently holding an object.

Shared Counter Information Shared counters serve as specialized kitchen counters for object transfer between agents. For each shared counter i, $D(S)$ includes the status for s_i, as "s_0 is empty" or "s_1 contains onion," to offer a complete environmental overview. Unlike kitchen counters, where only the closest empty counter is mentioned, all empty shared counters are mentioned.

```
<Inventory>: I am holding onion. Bob is holding nothing.

<My Location Information>: o0 is 0 units away. o1 is 1 units
      away. p0 is 3 units away. c0 is 6 units away blocked by
      Bob. c1 is 7 units away. d0 is 4 units away. s0 is 1
      units away. s1 is 0 units away. s2 is 1 units away. s3
      in 2 units away. Closest empty kitchen counter k12 is 1
      units away.

<Bob's Location Information>: o0 is blocked by Alice. o1 is 7
      units away. p0 is 3 units away. c0 is 0 units away. c1
      is 1 units away. d0 is 4 units away. s0 is 1 units away.
       s1 is 0 units away. s2 is 1 units away. s3 in 2 units
      away.

<Environment Details>: c0 contains 1 out of 3 onions. c0 is
      off. soup in c0 is not cooking. c1 contains 0 out of 3
      onions. c1 is off. soup in c1 is not cooking.

Available Actions: [place onion in c0, place onion in c1.,
      place onion on s0., place onion on s1., place onion on
      s2, place onion on s3., place onion on k12., wait., move
       away.]
```

B Hanabi Implementation Details

Hanabi is a card game in which all players are on the same team. The deck is made up of cards numbered 1 through 5, further divided into five different colors. Players are working together to create these numbered sequences (1 through 5) for each of the five colors. Each card played provides 1 point, and the goal is to obtain all 25 points (5 colors with 5 cards each) by completing the sequence for each color. The three actions a player can take include playing a card, discarding a card, and giving a hint (reveal) to a teammate. The challenge is that players cannot see their own cards. They may, however, see the cards that are in the hands of their teammate(s). This is where hints (reveals) come into play. Players are able to figure out which cards should be played through the reveal mechanism. You can hint (reveal) a teammate about a color or number in their hand, and this will point out to them all cards in their hand that have this color or number. The team starts with 8 reveal tokens but can recover used tokens when discarding a card or upon the completion of a stack. Playing a card carries a risk because the team loses completely if three incorrect cards are played. Further, an incorrectly played card gets sent to the discard pile. Each number of a particular color only has so many copies, so if, for example, the last copy of a green 3 is lost, then the team loses out on not only playing the green 3 but also the green 4 and 5. This is the risk of playing or discarding. With these risks, and since reveal tokens are scarce, players ideally try to make assumptions about implicit information that a reveal may offer, and, ideally, the team converges to particular conventions. Implicit communication, collaboration, and memory are key.

B.1 Game Description

We structure the game description of Hanabi into the overall objective, and the rules of the game.

```
The card game Hanabi has the following rules:
- The game uses a 50-card deck, divided into five colours (red
       (R), green (G), blue (B), yellow (Y), white (W)). Each
      color has cards of ranks 1 to 5. Each color has with
      three 1's, two 2's, two 3's, two 4's, one 5.
- Players have to create stacks of each color. Each color
      stack starts with a Rank 1 card and goes up one by one
      in ascending order up to Rank 5.  (e.g. Red Stack should
       go from R1 -> R2 -> R3 -> R4 -> R5). A card can only be
       played if it is the next in the incremental sequence
      for its color stack.
- Players can only see the other's hand, not their own.
- Players have plausible knowledge of their cards based on
      previously provided hints by the other player
- They can either play a card, give a reveal, or discard a
      card.
- Players can only chose an action from the Available Legal
      Actions.
***Actions:***
1. Reveal (Clue): Spend a reveal token to reveal cards with a
      particular color or rank. Revealing a color reveals all
      cards of that color in partner's hand. Revealing a rank
      reveals all cards with that rank in partner's hand. The
      game starts with 8 reveal tokens. If no token left, no
      more reveals can be given.
2. Discard: Discard a card to regain a reveal token and draw a
       new card.
3. Play a Card: If a card played follows sequence in its color
       stack, it succeeds. Success of rank 5 card in any stack
       gives an additional reveal token. Failure discards the
      card, and loses a life. Playing a card you are unsure
      about is risky as it costs a life and you have only 3
      lives. Before playing a card make sure that it's the
      next card in the sequence for that stack.
***The game ends when:***
- All five stacks are completed. 25 Points.
- Three lives have been lost. 0 Points no matter how many
      cards have been placed in the stack.

- After the last card from the deck is drawn and each player
      has had a final turn. Sum total of the top card ranks of
       each color stack.
I am Alice, playing the card game Hanabi with my partner Bob.

At each time step I will provide you with the relevant
      information of the game. I will also provide you with
      the legal action, help me select the best next action.
      Remember I am playing as Alice. Format your response as
      Explanation: <brief explanation for selecting the move>\
      nAction:<selected move>. Do not say anything else. Got
      it?
```

B.2 State Description

The state description includes the current Stack S, the player's knowledge of their cards K (updated based on clues), the partner agent's cards C, the partner agent's knowledge of their cards K' (updated based on previous clues), each card in the discard pile d_i, the remaining Life Tokens l, and reveal tokens r and the remaining Deck Size D. We also precalculate the next card that goes on each stack since LLMs frequently fail to count which card should go next on each stack.

```
It is currently My (Alice) turn.
Current Stacks:
Red - Red 5, Yellow - Yellow 4, Green - Green 1, White - White
      1, Blue - Blue 3
My cards based on my knowledge:
Card 0 could be: [Red, Yellow, Green, Blue] [1, 2, 3]
Card 1 could be: [Yellow, White, Blue] [1, 2, 3]
Card 2 could be: [Red] [2]
Card 3 could be: [Yellow, White, Blue] [1]
Card 4 could be: [Yellow, White, Blue] [1]
I can see Bob's Cards are:
[Card 0: Green 1]
[Card 1: Green 2]
[Card 2: Green 4]
[Card 3: White 4]
[Card 4: Yellow 1]
Bob's Knowledge about his cards:
Bob believes his Card 0 could be: [Yellow, Green, White, Blue]
      [1, 2, 4]
Bob believes his Card 1 could be: [Green, White] [1, 2, 4]
Bob believes his Card 2 could be: [Yellow, Green] [1, 2, 3, 4]
Bob believes his Card 3 could be: [Yellow, Green, White] [1, 2,
      3, 4]
Bob believes his Card 4 could be: [Yellow, Green] [1, 2, 4]
Remaining Reveal Tokens: 1
Remaining Lives: 1
Deck Size: 3
The discard pile is: [Red 4, Red 3, Red 1, Red 1, Yellow 5,
     Yellow 2, Yellow 4, Green 3, Green 2, Green 4, Green 3,
     Green 1, Green 5, Blue 5, Blue 3, Blue 4, Blue 4, Blue 1,
      White 4, White 3, White 2, White 5, White 3]
My Action History: [Discard Card 4, Play Card 0, Reveal Bob's
     Rank 3 Cards, Discard Card 0, Play Card 4]
The next playable cards for each stack are:
Red Stack is Full.
Only Yellow 5 can be played on Yellow Stack
Only Green 2 can be played on Green Stack
Only White 2 can be played on White Stack
Only Blue 4 can be played on Blue Stack

Available Actions:
A. Reveal Bob's Yellow color cards
B. Reveal Bob's Green color cards
C. Reveal Bob's White color cards
D. Reveal Bob's rank 1 cards
E. Reveal Bob's rank 2 cards
F. Reveal Bob's rank 4 cards
G. Play my Card 0
H. Play my Card 1
I. Play my Card 2
J. Play my Card 3
K. Play my Card 4
L. Discard my Card 0
M. Discard my Card 1
N. Discard my Card 2
O. Discard my Card 3
P. Discard my Card 4
```

C Examples of prompts of LLMs used in the Agent framework

C.1 Hanabi LLM Prompt

```
The card game Hanabi has the following rules:
    {self.rules}
I am {self.player_names[self.player_id]}, playing the card
game Hanabi with {self.player_names[1 - self.player_id]}.
At each time step I will provide you with the relevant
information of the game. I will also provide you with the
legal action, help me select the best next action.
Remember I am playing as {self.player_names[self.player_id]}.
Format your response as
Explanation: <brief explanation for selecting the move>
Action:<selected move>.
Do not say anything else. Got it?
```

C.2 Prompt for Theory of Mind Reasoning step

```
The card game Hanabi has the following rules:
{self.rules}
I am {self.player_names[self.player_id]}, playing the card
      game Hanabi with {self.player_names[1-self.player_id]}.
You are a Theory of Mind inference agent for our game. You
      will be provided with my partner's selected action and
      my latest state information after my partner took their
      action. You will provide me with two things: 1.  An
      explanation for my partner's previous action along with
      their intention and implicit communication. 2. What is
      the best information for me to give my partner based on
      their knowledge?
Format your response as:
Partner Action Explanation:<1 sentence explanation of partner
      action>
Clue Suggestion:<What information (specify rank or color)
      should I reveal to my partner based on their knowledge>.
```

C.3 Prompt for Answer Verification Step

```
You are an action verification agent for games. I will provide
you with an action and you need to check whether the action
satisfies the criteria:
1. Rule Following: It follows to the rules of the game.
2. Safety: It won't lead to the game ending immediately.
Think about the action, the current state of the stack and the
available lives and reveal tokens.
End you response with "Verification: Okay" if selected action
follows ***both*** criteria and "Verification: Not Okay"
otherwise. Restrict your response to 4-5 sentences.
```

C.4 Ablation on Theory of Mind Reasoning and Verification Steps

Method	Score	Bomb Rate
ToM + GPT-4-turbo + V	$\mathbf{13.33} \pm 0.88$	0.00
GPT-4-turbo + V	10.33 ± 0.88	0.00
GPT-4-turbo	4.33 ± 0.67	1.00

Table 6: Ablation study of LLM agents on Hanabi Challenge (w. GPT-4-turbo). Answer Verification (AV) markedly enhances overall performance by ensuring that actions that make incorrect assumptions are filtered out. The explicit Theory of Mind (ToM) reasoning provides further improvements by directly interpreting partner clues and requirements.

D Generating Questions for CoordinationQA

D.1 Environment Comprehension Questions

The Environment Comprehension (EC) questions are indirect formulations regarding spatial aspects of the layout. In order for an agent to correctly answer an EC question, they must have an understanding of the dynamic details of the current state and the rules of the game and exhibit spatial awareness. As such, when creating the EC questions, we carefully comb through a given scenario in search of salient points to probe an agent's understanding of the given environment. Some examples include:

```
<Inventory>: I am holding nothing. Bob is holding onion.
<My location information:> o0 is 1 units away. o1 is 0 units
      away. p0 is 1 units away. d0 is inaccessible. c0 is
      inaccessible. c1 is inaccessible. s0 is 1 units away. s1
       is 0 units away. s2 is 1 units away.
<Bob's location information>: o0 is inaccessible. o1 is
      inaccessible. p0 is inaccessible. d0 is 2 units away. c0
       is 0 units away. c1 is 0 units away. s0 is 0 units away.
       s1 is 1 units away. s2 is 2 units away.
<Environment Details>: c0 contains 3 out of 3 onions. c0 is on.
       soup in c0 is still cooking. c1 contains 0 out of 3
      onions. c1 is off. soup in c1 is not cooking. s0 is
      empty. s1 contains onion. s2 is empty. Closest empty
      kitchen counter k1 is 1 units away.

How many onions are still needed to fill up c0?
Available Answers:
A. 4 or more
B. 3
C. 2
D. 1
E. 0
```

```
My name is Alice. I am in room 1. Bob is in room 6. I was
      fixing the generator and there is only one more fix
      needed, which could be done before getting caught.
      Currently, we have information that the killer will move
       to the room 1 after this turn. Generator in room 1
      still needs 1 fix. Generator in room 2 is fixed. The
      exit gate is closed.

If I fix generator 1, is Bob in a position to escape?

Available Answers:
A. Yes, he's only one room away from the gate when it opens.
B. No, the killer is blocking his path to the exit gate.
C. No, we stil need to fix generator 2.
```

D.2 Theory of Mind Reasoning Questions

There are two primary question types in Hanabi for ToM Reasoning questions. In the first type, we ask the LLM about what information the partner agent needs, while in the second type, we ask it to make inferences about the partner agent's last action. For all games apart from Hanabi, the ToM questions ask the models to predict the next intended action of the partner agent

D.2.1 Hanabi Question Type-1

```
It is currently My (Alice) turn. Current Stacks: Red - Red 0,
      Yellow - Yellow 0, Green - Green 0, White - White 0,
      Blue - Blue 0
My cards based on my knowledge:
Card 0 could be: [Red, Yellow, Green, White, Blue] [1, 2, 3, 4,
       5]
Card 1 could be: [Red, Yellow, Green, White, Blue] [1, 2, 3, 4,
       5]
Card 2 could be: [Red, Yellow, Green, White, Blue] [1, 2, 3, 4,
       5]
Card 3 could be: [Red, Yellow, Green, White, Blue] [1, 2, 3, 4,
       5]
Card 4 could be: [Red, Yellow, Green, White, Blue] [1, 2, 3, 4,
       5]
I can see Bob's Cards are:
[Card 0: Red 3]
[Card 1: White 1]
[Card 2: Green 3]
[Card 3: White 4]
[Card 4: Blue 4]
Bob's Knowledge about his cards:
Bob believes his Card 0 could be: [Red, Yellow, Green, White,
      Blue] [1, 2, 3, 4, 5]
Bob believes his Card 1 could be: [Red, Yellow, Green, White,
      Blue] [1, 2, 3, 4, 5]
Bob believes his Card 2 could be: [Red, Yellow, Green, White,
      Blue] [1, 2, 3, 4, 5]
Bob believes his Card 3 could be: [Red, Yellow, Green, White,
      Blue] [1, 2, 3, 4, 5]
Bob believes his Card 4 could be: [Red, Yellow, Green, White,
      Blue] [1, 2, 3, 4, 5]
Remaining Reveal Tokens: 8
Remaining Lives: 3
Deck Size: 40
The discard pile is: []
My Action History: []
The next playable cards for each stack are:
Only Red 1 can be played on Red Stack
Only Yellow 1 can be played on Yellow Stack
Only Green 1 can be played on Green Stack
Only White 1 can be played on White Stack
Only Blue 1 can be played on Blue Stack
```

```
What information about his cards should I reveal to my partner
       so that he knows to play a card on his turn?
Available Answers:
A. Reveal Bob's Red color cards.
B. Reveal Bob's White color cards.
C. Reveal Bob's Green color cards.
D. Reveal Bob's Blue color cards.
E. Reveal Bob's rank 1 cards.
F. Reveal Bob's rank 3 cards.
G. Reveal Bob's rank 4 cards.
```

D.2.2 Hanabi Question Type-2

```
It is currently My (Alice) turn. Current Stacks: Red - Red 1,
      Yellow - Yellow 2, Green - Green 1, White - White 4,
      Blue - Blue 3
My cards based on my knowledge:
Card 0 could be: [Red, Yellow, Green, White, Blue] [1, 2, 3, 4,
       5]
Card 1 could be: [Red, Yellow, Green, White, Blue] [1, 2, 3,
      5]
Card 2 could be: [Red, Yellow, Green, White, Blue] [1, 2, 3,
      5]
Card 3 could be: [Red, Yellow, Green, Blue] [3]
Card 4 could be: [White] [5]
I can see Bob's Cards are:
[Card 0: Yellow 1]
[Card 1: Blue 1]
[Card 2: Blue 1]
[Card 3: Red 3]
[Card 4: Green 3]
Bob's Knowledge about his cards:
Bob believes his Card 0 could be: [Red, Yellow, Green, White,
      Blue] [1, 2, 3, 4, 5]
Bob believes his Card 1 could be: [Red, Yellow, Green, White,
      Blue] [1, 2, 3, 4, 5]
Bob believes his Card 2 could be: [Red, Yellow, Green, White,
      Blue] [1, 2, 3, 4, 5]
Bob believes his Card 4 could be: [Red, Yellow, Green, Blue]
      [3]
Remaining Reveal Tokens: 1
Remaining Lives: 2
Deck Size: 25
The discard pile is: [Yellow 4, Blue 2, Blue 3, White 2, White
       3, White 4]
My Action History: [Reveal Bob's Rank 2 Cards, Reveal Bob's
      Rank 5 Cards, Reveal Bob's Rank 2 Cards, Play Card 1,
      Reveal Bob's Rank 1 Cards,  Discard Card 0, Reveal Bob's
       Rank 3, Reveal Bob's Rank 2, Reveal Bob's Rank 2 Cards,
       Reveal Bob's Rank 1 Cards, Discard Card 3, Reveal Bob's
       White Color Cards, Discard Card 1]
The next playable cards for each stack are:
Only Red 2 can be played on Red Stack
Only Yellow 3 can be played on Yellow Stack
Only Green 2 can be played on Green Stack
Only White 5 can be played on White Stack
Only Blue 4 can be played on Blue Stack


What can I infer from my partner's previous action?
Available Answers:
A. I should Play Card 0
B. I should Play Card 1
C. I should Play Card 2
D. I should Play Card 3
E. I should Play Card 4
F. I should Discard Card 0
G. I should Discard Card 1
H. I should Discard Card 2
I. I should Discard Card 3
J. I should Discard Card 4
```

D.2.3 Other Games

```
<Inventory>: I am holding onion. Bob is holding nothing.

<My Location Information>: o0 is 0 units away. o1 is 1 units
      away. p0 is 3 units away. c0 is 6 units away blocked by
      Bob. c1 is 7 units away. d0 is 4 units away. s0 is 1
      units away. s1 is 0 units away. s2 is 1 units away. s3
      in 2 units away. Closest empty kitchen counter k12 is 1
      units away.

<Bob's Location Information>: o0 is blocked by Alice. o1 is 7
      units away. p0 is 3 units away. c0 is 0 units away. c1
      is 1 units away. d0 is 4 units away. s0 is 1 units away.
       s1 is 0 units away. s2 is 1 units away. s3 in 2 units
      away.

<Environment Details>: c0 contains 1 out of 3 onions. c0 is
      off. soup in c0 is not cooking. c1 contains 0 out of 3
      onions. c1 is off. soup in c1 is not cooking.


What action does my partner intend to take?
Available Actions:
A. pick up onion from o0.
B. pick up onion from o1.
C. pick up plate from p0.
D. pick up onion from s0.
E. pick up onion from s1.
F. pick up onion from s2.
G. pick up onion from s3.
H. pick up plate from s0.
I. pick up plate from s1.
J. pick up plate from s2.
K. pick up plate from s3.
L. wait.
M. move away.
```

D.3 Joint Planning Questions

Joint planning questions are effectively the same questions that the LLM solves when they are part of an agentic framework. For each scenario, we ask the LLM to answer the question: "What is the best next action?".

```
I (Alice) am in Room 6. Bob is in Room 1. Thief is in Room 2.
Door between Room 1 and 2 is closed. Door between Room 3 and 4
       is closed.

What action should I take next?
Available Actions:
A. Move to Room 1
B. Move to Room 5
C. Move to Room 9
D. Stay in current Room
```

E CollabCapture and CollabEscape

E.1 CollabCapture

CollabCapture places two agents in a set of interconnected rooms and doors. Their goal is to capture an adversary within the smallest number of moves possible. The adversary moves away from agents in a greedy fashion, but the agents have the ability to close and open doors while the adversary does not. The doors are controlled by a corresponding button that is in another location on the map. (See Figure 5)

CollabCapture contains one layout with four scenarios created by controlling the gate states (open/-closed). This corresponds to cases where players need to: 1. Pincer their opponent through coordination 2. One agent needs to enable the other agent by choosing not to chase the agent but rather open the door to allow the other agent to capture the adversary. 3. One agent needs to disable the adversary by closing a door, allowing the other agent to catch them. CollabCapture is based on the classic task of Pursuit Evasion from the perspective of the pursuers. This is representative of a common-payoff task as the only objective is capturing the adversary with no mixed incentives (Akin to deliveries in Overcooked, where all chefs get a common payoff with no preference for the one making the delivery).

E.2 CollabEscape

CollabEscape also places two agents in a set of interconnected rooms and doors. Their goal in this environment, however, is to escape an adversary that is looking to catch them. The map has two generators that power an exit gate, both of which need to be fixed. Upon fixing both generators, the players must then escape by reaching the exit gate. Only one player needs to reach the exit gate in order for them both to win. If the adversary catches either of them, however, they both lose. (See Figure 6)

In CollabEscape, we have one layout with two scenarios created by varying the starting positions of the two agents. Depending on their proximity to the adversary/generators, players need to apply strategies such as luring the adversary away from the partner, choosing to continue fixing the generators while sacrificing for the partner's safety, and manipulating the movement of the adversary. This is also representative of a common payoff task, as players are commonly rewarded if any one of them escapes, introducing roles of explicit assistance and sacrifice for a higher common payoff.

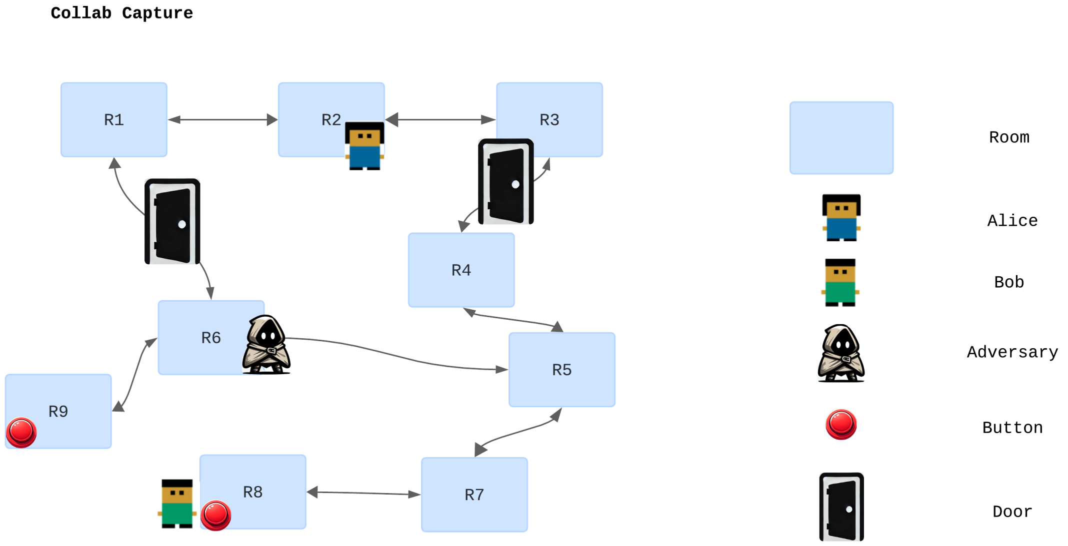

Figure 5: Map layout for CollabCapture

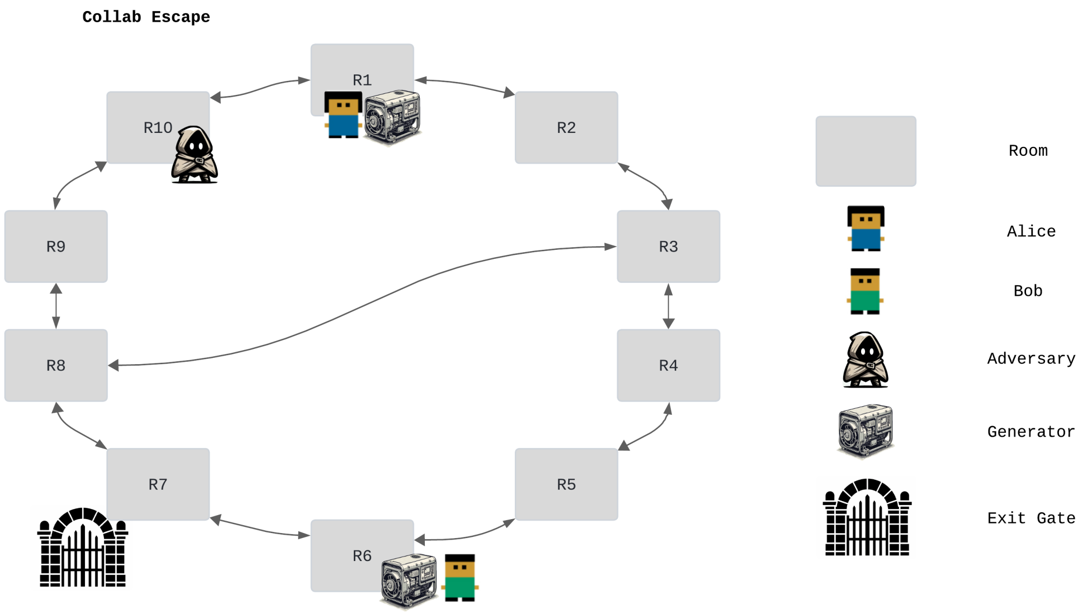

Figure 6: Map layout for CollabEscape