

UC Santa Cruz

UC Santa Cruz Electronic Theses and Dissertations

Title

LLM-Coordination: Developing Coordinating Agents with Large Language Models

Permalink

<https://escholarship.org/uc/item/4rm2z5w5>

Author

Agashe, Saaket

Publication Date

2023

Copyright Information

This work is made available under the terms of a Creative Commons Attribution License, available at <https://creativecommons.org/licenses/by/4.0/>

Peer reviewed|Thesis/dissertation

UNIVERSITY OF CALIFORNIA  
SANTA CRUZ

**LLM-COORDINATION: DEVELOPING COORDINATING  
AGENTS WITH LARGE LANGUAGE MODELS**

A thesis submitted in partial satisfaction of the  
requirements for the degree of

MASTER OF SCIENCE

in

COMPUTER SCIENCE AND ENGINEERING

by

**Saaket Agashe**

December 2023

The Thesis of Saaket Agashe  
is approved:

---

Professor Xin Eric Wang, Chair

---

Professor Yi Zhang

---

Professor Ian Lane

---

Peter Biel  
Vice Provost and Dean of Graduate Studies

Copyright © by

Saaket Agashe

2023

# Table of Contents

<b>List of Figures</b>	<b>v</b>
<b>List of Tables</b>	<b>vii</b>
<b>Abstract</b>	<b>viii</b>
<b>Dedication</b>	<b>x</b>
<b>Acknowledgments</b>	<b>xi</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Document Structure . . . . .	4
<b>2 Background</b>	<b>6</b>
2.1 Autoregressive Large Language Models (LLMs) . . . . .	6
2.2 Large Language Model Agents . . . . .	7
2.3 Multi-agent Coordination . . . . .	8
2.4 Theory of Mind . . . . .	9
<b>3 Related Work</b>	<b>10</b>
3.1 Multi-agent Coordination . . . . .	10
3.2 Planning and Reasoning with Large Language Models . . . . .	11
<b>4 Methodology</b>	<b>13</b>
4.1 LLM-Coordination Framework . . . . .	13
4.1.1 Game and Layout Description . . . . .	15
4.1.2 State Representation . . . . .	15
4.1.3 Feasible Action Generation: . . . . .	17
4.1.4 The Large Language Model . . . . .	18
4.1.5 Action Manager . . . . .	18

<b>5</b>	<b>Evaluation Environments</b>	<b>21</b>
5.1	Collab Capture . . . . .	21
5.2	CollabEscape . . . . .	22
5.3	Overcooked . . . . .	22
5.4	Overcooked-Explicit Assistance . . . . .	23
5.4.1	Gated Delivery . . . . .	24
5.4.2	Locked . . . . .	24
<b>6</b>	<b>Experiments</b>	<b>25</b>
6.1	Theory of Mind Inference and Situated Reasoning . . . . .	25
6.2	Sustained Coordination . . . . .	26
6.3	Robustness to Partners . . . . .	27
6.4	Explicit Assistance . . . . .	27
<b>7</b>	<b>Discussion</b>	<b>28</b>
7.1	Theory of Mind Inference and Situated Reasoning . . . . .	29
7.1.1	Results . . . . .	29
7.2	Sustained Coordination . . . . .	30
7.2.1	Baselines . . . . .	30
7.2.2	Results . . . . .	31
7.3	Robustness to Partners . . . . .	33
7.3.1	Results . . . . .	33
7.4	Explicit Assistance . . . . .	34
7.4.1	Results . . . . .	34
<b>8</b>	<b>Case Studies</b>	<b>37</b>
<b>9</b>	<b>Limitations</b>	<b>41</b>
<b>10</b>	<b>Conclusion</b>	<b>42</b>
	<b>Bibliography</b>	<b>43</b>
<b>A</b>	<b>Prompt Details</b>	<b>53</b>
A.1	Overcooked Game Description Prompts . . . . .	53
A.2	Overcooked Layout Prompts . . . . .	54
<b>B</b>	<b>Details of LLMs used in Experiments</b>	<b>58</b>

# List of Figures

2.1	A summary of LLM Agents and their components [39] . . . . .	7
4.1	Visual summary of the LLM-Co framework. Our framework serves as the backbone for an individual agent, focusing on bringing out its coordination ability. The framework translates abstract game details into an LLM-compatible format and then utilizes the generated LLM output to take actions in the game world. . . . .	13
5.1	The CollabCapture game involves two agents, Alice (Blue) and Bob (Green), chasing a thief across multiple rooms. Some rooms are connected by doors, which can be controlled by buttons in different rooms.	21
5.2	All layouts from the overcooked environment we use for our tests. The two agents Alice (Blue) and Bob (Green) need to collaborate to cook, plate, and deliver onion soups. From Left to Right: Cramped Room, Asymmetric Advantages, Forced Coordination, Coordination Ring, and Counter Circuit. . . . .	22
5.3	Additional Layouts that require agents to explicitly help their partner complete a delivery. These new layouts utilize <b>walls</b> and <b>gates</b> to create situations requiring explicit assistance. . . . .	23
7.1	LLMs performance on the LLM-ToM-Reasoning test set. Partner action intent prediction accuracy shows the Theory Of Mind ability of LLMs under test and the optimal action reasoning accuracy infers the Situated Reasoning effect of LLMs under test. GPT-4 achieves the best performance among tested LLMs. . . . .	28
8.1	The LLM-Co agent (Blue) understands that its partner has already picked up an onion and will probably end up placing it in the cooker to complete the requirement. So, it chooses to "pick up a plate" which would help in the next stage. . . . .	37

8.2	The LLM-Co Agent (Blue) thinks about the partner’s mind states (shown in red) and makes an informed decision (highlighted in green) to move away to make way for the partner agent) . . . . .	38
8.3	The LLM-Co Agent (Blue) realizes that the onion it picked up cannot be used and decides to correct itself by placing it on the kitchen counter to free up its hands. . . . .	39
8.4	The LLM-Co Agent (Blue) recognizes that its partner is stuck behind a closed door an chooses to open the gate g0 to enable them. . . . .	40

# List of Tables

4.1	Complete medium-level action set. . . . .	20
7.1	Comparison of game play between self-play baselines (PPO, and PBT) and LLM-Co Agents. LLM-Co agents outperform RL methods on 4 out of 5 layouts, demonstrating highly effective reasoning under sustained coordination. . . . .	31
7.2	Comparison of AI-Human Proxy Game play. We compare Behavior Cloning Agents, PPO_BC Agents with LLM-Co agents utilizing the GPT-4 LLM. The LLM-Co agents are able to outperform or match the performance of Reinforcement Learning models, indicating that LLM agents are robust to the choice of partner agents. . . . .	33
7.3	Comparison of Gameplay in the Overcooked-Assistance Layouts with and without Helper Directive. The results indicate that the Large Language Model needs to be prompted to be aware of situations where their partner might need assistance in order to be effective in the Overcooked-Assistance layouts. . . . .	34
7.4	Comparison of Gameplay on Overcooked-Assistance Layouts between RL baselines and LLM Agents. The RL baselines being able to effectively solve the deliveries indicates that the environments are solvable through self-play training. The high scores achieved by LLM agents demonstrate that LLM agents are capable of reasoning for providing explicit assistance to their partners. . . . .	35



## Abstract

LLM-Coordination: Developing Coordinating Agents with Large Language  
Models

by

Saaket Agashe

It is essential for intelligent agents to not only excel in isolated situations but also coordinate with partners to achieve common goals. Current Multi-agent Coordination methods rely on Reinforcement Learning techniques to train agents that can work together effectively. On the other hand, agents based on Large Language Models (LLM) have shown promising reasoning and planning capabilities in single-agent tasks, at times outperforming RL-based methods. In this study, we build and assess the effectiveness of LLM agents in various coordination scenarios. We introduce the LLM-Coordination Framework to enable LLMs to complete coordination tasks. We evaluate our method on three game environments and organize the evaluation into five aspects: Theory of Mind, Situated Reasoning, Sustained Coordination, Robustness to Partners, and Explicit Assistance. First, the evaluation of the Theory of Mind and Situated Reasoning reveals the capabilities of LLM to infer the partner’s intention and reason actions accordingly. Then, the evaluation around Sustained Coordination and Robustness to Partners further showcases the ability of LLMs to coordinate with an unknown partner in complex long-horizon tasks, outperforming Reinforcement Learning baselines. Lastly, to test Explicit Assistance, which refers to the ability of an agent to offer help proactively, we introduce

two novel layouts into the Overcooked-AI benchmark, examining if agents can prioritize helping their partners, sacrificing time that could have been spent on their tasks. This research underscores the promising capabilities of LLMs in sophisticated coordination environments and reveals the potential of LLMs in building strong real-world agents for multi-agent coordination.

To My Parents and my Partner.

## Acknowledgments

I would like to express my deepest gratitude to Dr. Xin Eric Wang, my thesis supervisor, for their unwavering support and guidance throughout the course of this research. Your expertise and insights have been invaluable to my work.

I am also immensely grateful to Professor Yi Zhang and Professor Ian Lane, whose expertise and feedback significantly contributed to the refinement of this thesis.

Special thanks to my colleague Yue Fan for his invaluable insights and suggestions regarding this project.

I am indebted to my family and my partner for their endless love, moral support, and encouragement. Your belief in me has been a constant source of strength and motivation.

Lastly, I extend my appreciation to the staff and faculty of the CSE Department at the University of California, Santa Cruz, for providing a conducive environment for learning and research.

# Chapter 1

## Introduction

The need for coordination can arise from mundane tasks like cooking and cleaning to more essential tasks like coordinated search-and-rescue operations. The assimilation of AI tools and agents into the human experience cannot be complete without AI agents being able to coordinate in a similar manner. Current methods for Multi-Agent coordination predominantly utilize Reinforcement Learning approaches that suffer from issues of sample efficiency, lack of robustness to new partners, and lack of interpretability. More Recently, Agents based on Large Language Models have emerged as a new paradigm for developing situated, task-oriented agents [36, 42]. LLMs are capable of solving reasoning problems with minimal downstream training data due to their pretraining and generate free-text explanations, making them promising candidates for developing coordination agents. While the planning and reasoning abilities of Large Language Models for common-sense and logical reasoning tasks have been extensively studied, their emergent cooperative reasoning abilities are underexplored.

In this study, we introduce the LLM-Coordination Framework to facilitate multi-agent coordination using Large Language Models. We will refer to agents using the LLM-Coordination framework as LLM-Co agents. The LLM-Co framework is an Agent framework with the following features

- Programmatically converts real-time state information to textual description that the LLM can parse.
- Generates feasible action space based on the current state.
- Parses LLM response and converts it to a sequence of low-level actions
- Maintains a memory of previous actions for the LLM

We then systematically evaluate the LLMs and the LLM-Co framework on the following critical competencies required for effective coordination: Theory of Mind Inference (understanding others' beliefs and intentions), Situated Reasoning (contextual action analysis), Sustained Coordination (long-term action planning and adjustment), Robustness to Partners (adapting to new collaborators), and Explicit Assistance (actively aiding partners).

We begin by conducting a preliminary study that allows us to determine the baseline abilities of various Large Language Models and their suitability to be used in the LLM-Coordination Framework. We do this by examining LLMs' Theory of Mind Inference and Situated Reasoning capabilities, which are foundational for coordination. These abilities enable models to understand others' intentions and contextualize these

understandings within their environment. We introduce the "LLM-ToM-Reasoning Test Set," comprising diverse scenarios from multiple coordination environments, to assess LLMs' performance in deducing optimal actions based on their partner's intentions and environmental context. Comparing four different LLMs (GPT-4, GPT-3.5-turbo, Vicuna-33B, and Vicuna-13B), we find GPT-4 to be significantly superior, getting an almost perfect score. The other LLMs cannot provide sufficient correct responses to be suitable for following multi-turn coordination.

Next, we evaluate the sustained coordination abilities of LLM-Coordination Agents. We use GPT-4 as the LLM of choice as it is the only candidate that provides acceptable ToM and Situated Reasoning scores. Providing the LLM with Task Description, Turn-by-turn state information, action memory, and a list of feasible actions at each timestep, we explore the sustained coordination reasoning in LLMs on the Overcooked-AI benchmark. We compare the performance of LLM Agents (w. GPT-4) with Reinforcement Learning (RL) based baselines, which are the gold standards for AI-AI gameplay. We then experiment with varying the partners in the Overcooked Environment. We pair LLM-Co agents with proxy human agents to test the agents' robustness to partners. We observe that LLM agents perform better than or equal to the RL baselines in both AI-AI and AI-human proxy gameplay.

Previous works introducing novel methods for developing Collaborative Agents [46, 21, 45] using LLMs do not test their abilities to understand the concept of common payoff and proactively help their partners (Explicit Assistance). We introduce two new layouts in the Overcooked environment, including a gate element that forces agents

to assist their partners in completing deliveries. Through experiments on these new layouts, we discover that LLM agents can determine the right strategy needed to help out their partners. However, they require a "helper prompt" to be attentive to situations where their partner may need such help. We train RL baselines on these new layouts and discover that LLMs with helper directives outperform these baselines.

This research highlights the potential of LLMs in coordination reasoning and contributes a novel perspective to the field of AI-agent collaboration. The LLM-Coordination Framework opens new avenues in the development of AI agents capable of sophisticated, context-aware, and coordinated decision-making, setting a foundation for future explorations in multi-agent AI systems.

## 1.1 Document Structure

- Chapter 2 will cover some prerequisite concepts and terminology about LLMs, LLM Agents, and Multi-agent Coordination that will be used throughout the document.
- Chapter 3 will discuss prominent related work that explores Large Language Models as agents acting in a situated environment and covers the current state of Multi-agent Coordination Methods and Environments.
- Chapter 4 will introduce the LLM-Coordination Framework to enable LLMs to complete coordination games.
- Chapter 5 will cover the environments we have used to conduct experiments, in-



cluding two text-based games, CollabCapture and CollabEscape, the Overcooked-AI benchmark [5] and two newly introduced layouts for Overcooked which we designed for proving explicit assistive abilities of LLM Agents.

- Chapter 6 covers the details of all the experiments for systematically evaluating LLMs and the LLM-Co Framework on the five core competencies for Coordination.
- Chapter 7 consists of detailed results and implications of these results on the suitability of LLMs for developing Coordination Agents.
- Chapter 8 discusses interesting qualitative case studies observed during experiments on the LLM-Co Framework.
- Finally, Chapters 10 and 9 mentioned our conclusions and the current limitations of our method.

# Chapter 2

## Background

### 2.1 Autoregressive Large Language Models (LLMs)

Autoregressive Large Language Models are Transformer models trained on a large amount of textual data. These models are distinguished by their extensive scale, often comprising millions or billions of parameters. LLMs are exposed to internet-scale data during their pretraining. Autoregressive implies that the models are trained to generate the immediate next token ( $w_k$ ), given previous tokens ( $w_{0..(k-1)}$ ) as context (For example GPT [28]). It has been observed that scaling Large Language Models (LLMs) to billions of parameters with more training data results in emergent few-shot reasoning abilities in LLMs [4].

The most recent paradigm shift in Large Language Models came from the introduction of Reinforcement Learning from Human Feedback RLHF [23]. LLMs trained with RLHF are capable of instruction following and verbal reasoning, going beyond mere

language comprehension. The reasoning and planning abilities of LLMs can be further enhanced using the Chain-of-thought reasoning approach where the LLM is prompted to "think step by step" or provide an explanation before outputting the answer [38] or Self-Reflection [43] which requires an agent to reflect on experiences during an episode.

## 2.2 Large Language Model Agents

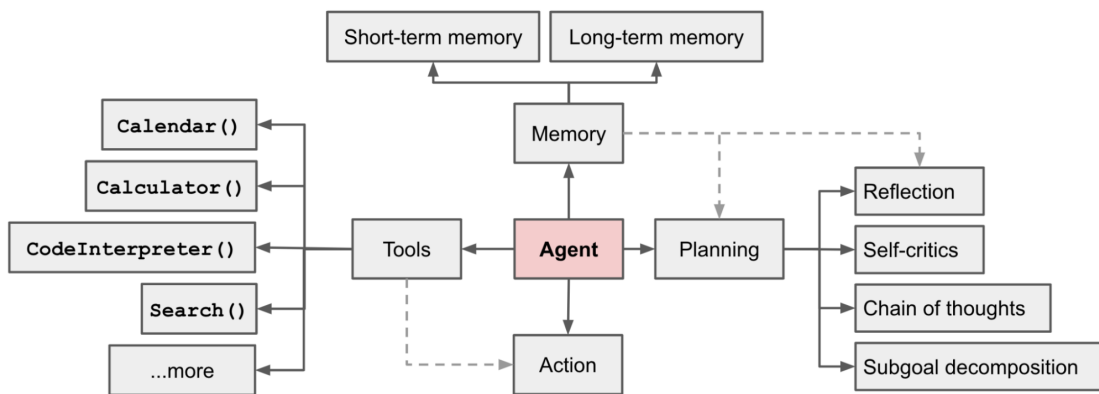


Figure 2.1: A summary of LLM Agents and their components [39]

Large Language Models are only capable of generating textual tokens based on context. Agent Frameworks enable LLMs to interact with the real world. These frameworks typically augment a Large Language Model with Planning, Memory, and Tools. Memory, including long and short-term memory, are summarized textual descriptions of previous actions and interactions, which are provided as context to the LLMs. The LLM agent can choose to use tools for performing specific tasks like arithmetic oper-

ations, path search, code interpretation etc. Tools enable agents to reason based on factually correct information and interact with the real world to gather information. Planning includes strategies like Chain of Thoughts [38] or Self-Reflection [43] that enhance the reasoning abilities of LLMs in context. Figure 2.1 [39] shows an overview of LLM Agents and their modules.

## 2.3 Multi-agent Coordination

A pure coordination game is a setting where all agents get a common payoff as a reward. In such a setting, the best strategy for participating agents is to cooperate with their partners. As humans, we partake in multi-agent coordination in our daily lives, from navigating traffic to saving lives. Multi-agent Reinforcement Learning (MARL) methods are the current gold standard for navigating and solving Multi-agent Coordination problems. These agents are trained using Self-Play, which means an agent is paired up with another agent of the same type, and they learn to complete their objective together through online learning.

Zero-shot coordination [9] is a special case of Multi-agent coordination where participating agents are paired up for the first time during evaluation and haven't been trained together. This ability is essential for agents to coordinate with new, unseen agents in realistic scenarios.

## 2.4 Theory of Mind

Theory of Mind (ToM) is the ability of rational agents to comprehend the fact that other agents have different beliefs from their own. ToM is critical to establishing multi-agent coordination where agents need to frequently reason about their partner’s beliefs and intentions before choosing their own next action. There have been recent studies that show emergent Theory of Mind inference abilities in Large Language Models [15]. We utilize the Theory of Mind inference as a core competency for evaluating Large Language Models in our work.

# Chapter 3

## Related Work

### 3.1 Multi-agent Coordination

There are two types of works prominent in Multi-agent Coordination Problems, the first include test environments and the latter includes developing MARL solutions. Various benchmarks have been used to evaluate Multi-Agent Coordination abilities over the years [20, 1]. In recent years, the Overcooked environment has emerged as a popular testbed for coordination experiments [5, 41]. Our research leverages the Overcooked-AI environment [6]. The foundational work by [5] emphasized the significance of incorporating human data for effective collaboration. Subsequent research has pivoted towards enabling self-play-trained agents to coordinate seamlessly with humans within this environment. These studies employ various techniques, including self-play with past agent checkpoints [33], centralized population entropy objectives [47], open-ended objectives using graph theory [17], policy ensembles with context-aware mechanisms [19], and the

incorporation of human biases as linear hidden rewards [44], to enhance the training and diversity of AI agents in different scenarios. Embodied environments usually set up in household environments have also been recently used to study multi-agent coordination [27, 12, 13, 8].

## 3.2 Planning and Reasoning with Large Language Models

Large Language Models (LLMs) have demonstrated remarkable capabilities of reasoning in natural language [22, 24, 7]. These models have achieved state-of-the-art performance across a spectrum of NLP tasks, showcasing their proficiency at verbal reasoning. Strategies like Chain of thought prompting[38], which generates step-by-step free-text explanations before coming to conclusions have further boosted the reasoning capacities of LLMs. Approaches augmenting an LLM with memory, belief, and tools have shown to be useful in multi-step problem-solving [25, 10, 29]. Isolated LLM agents have shown to be capable of life-long learning and task completion in open-domain survival games, outperforming existing SOTA Reinforcement Learning methods [42, 36]. More recently, such LLM agents have been paired with rule-based low-level planners to execute tasks in embodied environments [18, 32]. [46] demonstrated efficiency increase in collaborative embodied multi-agent setting, and [21] have shown the ability of collaborative manipulator motion planning using LLMs. The abilities of LLM agents as coordination partners are still in the nascent stages. In this work, we perform evaluations to systematically discover how current LLM-based agents reason

and plan while coordinating with partner agents.



# Chapter 4

## Methodology

### 4.1 LLM-Coordination Framework

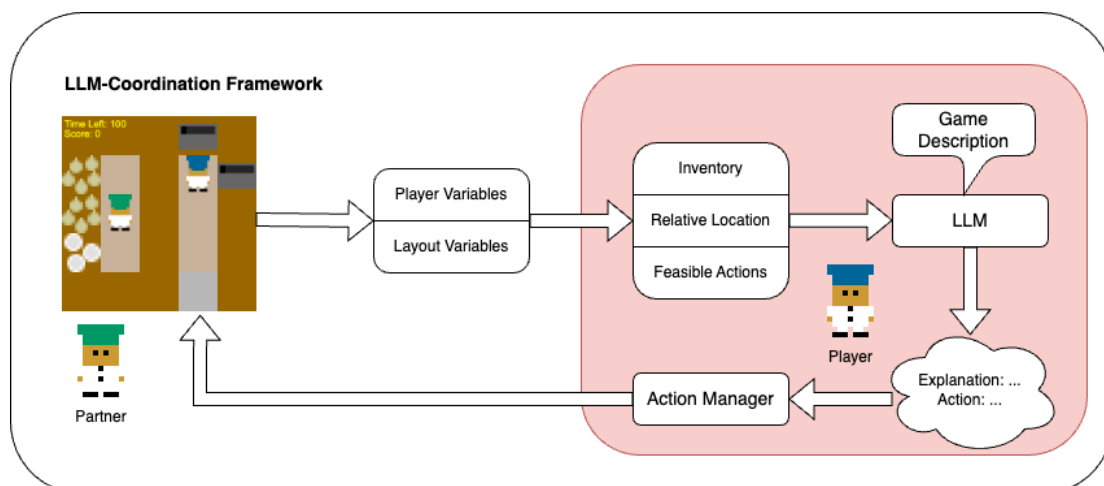


Figure 4.1: Visual summary of the LLM-Co framework. Our framework serves as the backbone for an individual agent, focusing on bringing out its coordination ability. The framework translates abstract game details into an LLM-compatible format and then utilizes the generated LLM output to take actions in the game world.

The LLM-Coordination Framework is responsible for translating coordination games into textual objectives compatible for LLMs. The details of the game along with the rules and the layout of the map are condensed into a short **Game Description** ( $G$ ). Along with the game description, we also provide a set of **Directives** ( $D_i$ ) that guide agent behavior. These descriptions are passed as initial prompts to the Large Language Model.

At each turn, the LLM receives the current **state description** ( $D(S)$ ) that is programmatically obtained from the environment, and the player states  $S$ . Since LLMs struggle with grid-based reasoning and navigation, we provide relative distances from the agent to each location of interest in the state description. Along with player-specific variables, other salient state variables are also included as natural language descriptions. Finally, an agent is provided its partner’s inventory and relative position to allow it to consider their intentions. The state information provided to the LLM is equivalent to what a Reinforcement Learning agent would receive in the form of vectors.

The LLM operates at a **medium-level action space** which is made up of **verb-based actions** like "pick", "place", "move" etc. It is provided with a set of **feasible actions**  $M_f$  to choose from to enable easier reasoning. The feasible action set is decided on the basis of player inventory and accessibility of locations.

The LLM utilizes the information  $\langle G, D_i, S, M_f \rangle$  to assess the situation and generates an action  $m$  from the provided set  $M_f$ . We then use an **Action Manager** to interpret the action based on the verb used and the location mentioned. The Action Manager generates low-level actions needed to execute the medium-level action. In the

following experiments, we will refer to LLM Agents that use the LLM-Coordination Framework as **LLM-Co Agents**. We will now dive deeper into the details of the LLM-Coordination Framework.

#### 4.1.1 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.

#### 4.1.2 State Representation

The State Representation Module programmatically converts the state information into a natural language description  $D(S)$ , 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. We refer to the Blue agent as Alice and the Green agent as Bob. The following information is included in  $D(S)$ :

**Objects Held by Each Player** The state description  $D(S)$  begins by detailing the inventories  $I_{\alpha_1}$  and  $I_{\alpha_2}$  of Alice and Bob, respectively. Each inventory  $I_{\alpha_i}$  (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: "You are holding  $I_{\alpha_1}$ . Bob is holding  $I_{\alpha_2}$ ." 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  $\text{loc}$ , we calculate the distance  $d_{(P_i, \text{loc})}$  as the number of steps required to reach  $\text{loc}$  from  $P_i$  using the shortest available path. The state description  $D(S)$  then includes this processed location information in the format: " $\text{loc}$  is  $d_{(P_i, \text{loc})}$  units away." Here,  $\text{loc}$  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 " $\text{loc}$  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  $\text{CookerState}_i$ , which can be either

”Off” or ”On”. Lastly, the current condition of the soup in the cooker is represented by  $\text{SoupState}_i$ , 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  $\text{CookerState}_i$ . Soup in  $c_i$  is  $\text{SoupState}_i$ .”

**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  $k_{\text{empty}}$  and the set  $K_{\text{filled}}$  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.

### 4.1.3 Feasible Action Generation:

Table 4.1 shows our full high-level action set used for the Overcooked environment. These are directives that can be selected by the LLM based on the provided state information. The action set is based on the planning objectives used by [5] in their coupled planning gameplay. The action set is complete in the sense that an agent can utilize the action set to complete multiple deliveries in the Overcooked AI environment.

The constrained action set generator verifies the feasibility of performing an

action before making it available to the LLM. This is a rule-based program that reduces the set of available actions. For example, It is not possible to pick up an onion or a plate if the agent is already holding an onion. In such a case, we will remove actions like “*pick up onion from o0*”, “*pick up plate from p0*” from the set of available actions, which will be provided to the LLM along with the state description at each turn.

#### 4.1.4 The Large Language Model

The LLM takes the game description, environment description, state description, and the feasible action set as its input, along with a history of the previous actions (5 actions.) It then selects an action from the set of feasible actions and formats its response as *Analysis: <analysis>. Action:<action>*. The LLM is asked to elucidate the current situation, including the environment state, a guess about the other player’s intention (ToM), and the explanation behind their next action in the analysis section.

#### 4.1.5 Action Manager

This module converts the high-level actions chosen by the Large Language Model (LLM) into specific, executable steps. Upon receiving a directive from the LLM, the module uses a Breadth-First Search algorithm to identify the shortest path to the target. The immediate next step along this path is then selected as the agent’s action for that moment.

The module maintains control until it fully executes a complex directive, such as “*place onion in c0*”. This directive would include both the sequence of movements

needed to reach the target location and the final action to complete the task, like placing the onion. Once this directive is completed, control returns to the State Representation Module, which then consults the LLM for the next high-level action.

The Action Manager also handles stalemate situations where both agents contend for the same spot or each other's spots using a combination of querying the LLM for another action and deterministic move-away actions.

This design approach relieves the LLM from the complexities of low-level motion planning, an area where it typically struggles. It also reduces the number of calls to the LLM, saving both time and computational resources.

Table 4.1: Complete medium-level action set.

<b>High-level Actions</b>	<b>Description</b>
pick up onion from oX.	Pick up an onion from onion dispenser number X.
pick up plate from pX.	Pick up a plate from plate dispenser number X.
put onion in cX.	Place the onion into cooker number X.
put soup on plate from cX.	Serve the soup from cooker number X onto a plate.
deliver soup in dX.	Deliver the soup to delivery location number X.
pick up onion from sX.	Pick up an onion from shared counter number X.
pick up plate from sX.	Pick up a plate from shared counter number X.
place onion on sX.	Place the onion on shared counter number X.
place plate on sX.	Place the plate on shared counter number X.
pick up onion from kX.	Pick up an onion from kitchen counter number X.
pick up plate from kX.	Pick up a plate from kitchen counter number X.
place onion on kX.	Place the onion on kitchen counter number X.
place plate on kX.	Place the plate on kitchen counter number X.
open gX.	Open gate gX.
wait.	wait for one time-step.
move away.	Randomly move away from the current location away from the other agent.



# Chapter 5

## Evaluation Environments

### 5.1 Collab Capture

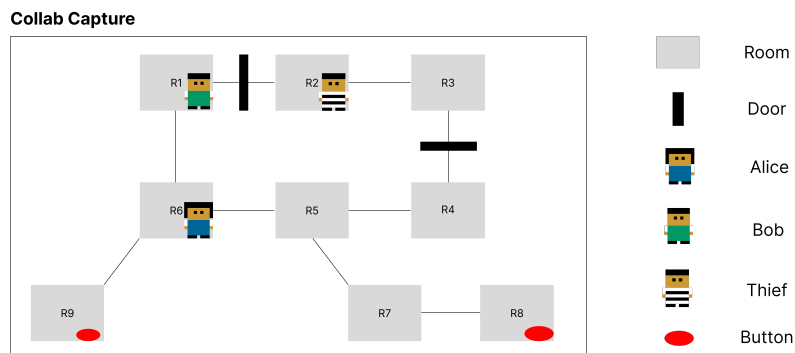


Figure 5.1: The CollabCapture game involves two agents, Alice (Blue) and Bob (Green), chasing a thief across multiple rooms. Some rooms are connected by doors, which can be controlled by buttons in different rooms.

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 different rooms. The agent's task is to capture the adversary in the least amount of time using effective strategies, including

cornering the adversary, disabling the adversary, or enabling their partners.

## 5.2 CollabEscape

Based on the popular Video Game "Dead-by-Daylight", Collaborative Escape involves two agents trying to escape an adversary in a maze of interconnected rooms. They need to fix two generators located in randomly selected rooms to open an exit portal. The adversary tries to catch the agents, and the win condition is any one agent escaping. This game requires strategies like luring the adversary away from the partner, sacrificing for the partner's safety, and manipulating the movement of the adversary.

## 5.3 Overcooked

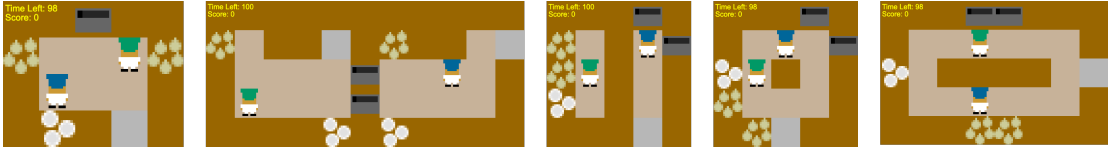


Figure 5.2: All layouts from the overcooked environment we use for our tests. The two agents Alice (Blue) and Bob (Green) need to collaborate to cook, plate, and deliver onion soups. From Left to Right: Cramped Room, Asymmetric Advantages, Forced Coordination, Coordination Ring, and Counter Circuit.

In the Overcooked-AI environment [5], two agents—Alice (Blue) and Bob (Green)—collaborate to cook and deliver onion soups. Different environments feature varying numbers of onion dispensers ( $o$ ), plate dispensers ( $p$ ), cookers ( $c$ ), delivery areas ( $d$ ), and counters ( $k$ ). Agents must load three onions into a cooker to start it, which

takes 20 time steps to cook. Once done, an agent transfers the soup to a plate and delivers it.

## 5.4 Overcooked-Explicit Assistance

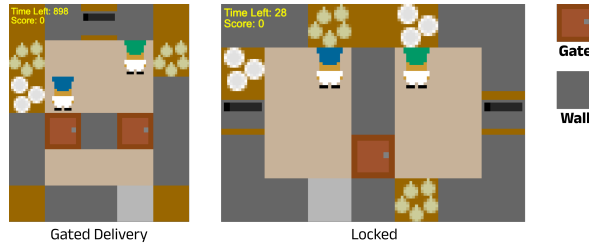


Figure 5.3: Additional Layouts that require agents to explicitly help their partner complete a delivery. These new layouts utilize **walls** and **gates** to create situations requiring explicit assistance.

The layouts in Overcooked-AI [5] are an excellent test for gauging the ability of participating agents to sync their actions with their partners. It requires agents to effectively navigate the layout and time their actions in response to their partners in order to increase efficiency. However, none of these environments elicit the need for agents to explicitly help out their partner sacrificing their own time.

We intend to evaluate LLM agents’ ability to make the choice to actively help their partners. If the LLMs cannot make such a choice implicitly, we intend to see the effect of tuning the directives to bring about such a cooperative intention. To elicit situations that require one agent to drop their own cooking/delivery and help out their partners, we extend the Overcooked environment by introducing 2 new facilities (Gates, Walls) and 2 new layouts. We now introduce the two new layouts:

### 5.4.1 Gated Delivery

Visualized in Figure 5.3, the Gated Delivery layout requires both agents to help out their partners during soup delivery. The two gates, g0 and g1 make the delivery area inaccessible. Gates can be opened by an agent, provided they are not holding anything in their hand. Once opened, a gate remains open for a short time which is enough for an agent to move through it but not enough for an agent to open it in advance before picking up cooked soup for delivery. This necessitates an agent not holding cooked soup in their hand to go and open the gate for the delivery agent. The kitchen counters are replaced by walls to prevent the agents from taking the loophole of placing their soups temporarily on counters to open the gates. In this environment, both agents are equally placed, and they need to be acutely aware of their partner's needs in order to complete even a single delivery.

### 5.4.2 Locked

Visualized in Figure 5.3, features two agents in two partitions of the layout. The agent in the left partition can access onions, plates, a cooker, and a delivery area. The agent in the right partition has their delivery area access restricted. The agent in the left partition has to understand that their partner is locked behind a closed gate, holding a soup. In real-life scenarios, one collaborating partner might find themselves similarly disadvantaged. To be a reliable partner, the advantaged agent needs to understand the situation and make the choice to help out their partner since that provides a better common payoff.

# Chapter 6

## Experiments

### 6.1 Theory of Mind Inference and Situated Reasoning

A coordinating agent is expected to, first and foremost, be mindful of its partner. Having made a guess about what their partner intends to do, an agent needs to take environmental setup into consideration before coming to a rational decision. While recent works have explored the Theory of Mind inference abilities in Large Language Models, this analysis has been decoupled from situated reasoning abilities crucial for coordination. We introduce the first LLM-ToM-Reasoning Test Set to systematically evaluate the ability of Large Language Models to guess their partner’s intentions and follow up with a reasonable explanation of the next step.

The LLM-ToM-Reasoning Benchmark comprises of three text-based game environments: CollabEscape, CollabCapture, and Overcooked. In CollabEscape, two coordinating agents need to repair generators in a connected room environment to open an

escape portal while evading an adversary. In CollabCapture, two agents work together to corner, trap, and capture an adversary in a connected room environment with doors and buttons. Finally, we use a text-based version of Overcooked, where two agents must cook and deliver meals. The LLM-ToM-Reasoning Test Set includes a set of 18 situations extracted from these three imaginary scenarios, designed to evaluate ToM inference and Situated Reasoning abilities of Large Language Models. It is important to note that we only use the objectives and structures of these games to generate situations that elicit coordination reasoning.

We evaluate the ToM inference and Situated Reasoning abilities of four different LLMs, namely Vicuna13B, Vicuna33B, GPT-3.5-turbo (ChatGPT), and GPT-4. While the GPT-4 model has been extensively evaluated for its reasoning and planning abilities in single-agent and multi-agent scenarios, this is the first work extending multi-agent coordination evaluation to open-source LLMs.

## 6.2 Sustained Coordination

While the LLM-ToM-Reasoning Benchmark evaluates single-turn coordination reasoning in LLMs, ideal agents must be capable of sustained coordination over multiple turns. The Overcooked-AI environment [5] has been used by several MARL methods as the go-to evaluation suite for multi-agent coordination and Zero-Shot Coordination. We utilize the five layouts Cramped Room, Asymmetric Advantages, Forced Coordination, Coordination Ring, and Counter Circuit, which feature a two-agent coordination

gameplay for our evaluation.

We provide the LLM with structured inputs consisting of an initial Game Description, a set of Directives to follow, summarized state information, action history and a list of feasible medium-level actions at every turn. The LLM produces a medium-level action and an optional analysis. An action manager interprets this medium-level action, breaking it down into executable low-level actions based on the verb and location provided by the LLM.

### 6.3 Robustness to Partners

It is essential for coordination agents to generalize to different partners. To test the Robustness of LLM-Co Agents to the choice of partners, we pair them with Human-proxy agents. These proxy agents utilize a Behavior Cloning model trained on human-human trajectories collected by [5]. This setting of playing with proxy-human agents has been used by preceding works [5, 33, 17, 47, 19, 44] as a standard test for Zero-Shot Coordination (ZSC).

### 6.4 Explicit Assistance

We use our newly developed layouts for Overcooked (Locked and Gated Delivery) as test-beds to study the ability of LLMs to recognize and provide explicit assistance to their partners during coordination scenarios. The test settings are similar to overcooked spanning 400 timesteps.

# Chapter 7

## Discussion

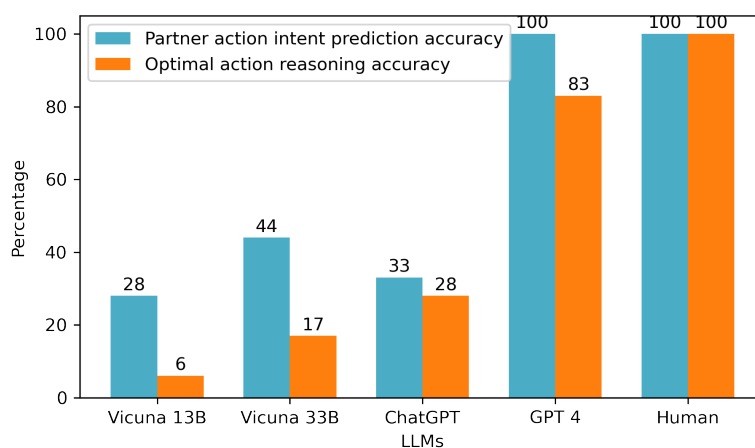


Figure 7.1: LLMs performance on the LLM-ToM-Reasoning test set. Partner action intent prediction accuracy shows the Theory Of Mind ability of LLMs under test and the optimal action reasoning accuracy infers the Situated Reasoning effect of LLMs under test. GPT-4 achieves the best performance among tested LLMs.



## 7.1 Theory of Mind Inference and Situated Reasoning

To evaluate ToM inference and situated reasoning in LLMs, we propose the LLM-ToM-Reasoning test set, which is a suite of 18 scenarios posed with questions among all three games: Collaborative Capture, Collaborative Escape, and Overcooked. The test set only includes scenarios hand-picked to represent pivotal situations that require the agent under-test to first take their partner’s possible next actions into active consideration, reason about the current state, and adjust their actions that ”indirectly” lead to the best possible outcome. The agent includes both the ToM inference as well as its own action prediction in the output.

### 7.1.1 Results

**Comparative analysis of LLMs on the LLM-ToM-Reasoning test set** We use the collected LLM-TOM-Reasoning test set to scrutinize the LLMs in the Theory Of Mind (ToM) and Situated Reasoning aspects, which respectively refer to the ability to understand the beliefs and intentions of other entities and the ability to contextualize this understanding within the environmental dynamics to formulate appropriate responses. LLMs under-test are required to solve the questions in the LLM-TOM-Reasoning test set, where the output answers to questions are manually compared against the ground truth. First, we calculate the accuracy of the LLM predictions concerning their partner’s actions to indicate the ToM ability. Then, the accuracy of predictions for the next appropriate action and analysis of the current scenario shows

the situated reasoning effectiveness. Figure 7.1 shows that GPT-4 outperforms the other LLMs, GPT-3.5-turbo, Vicuna-33B, and Vicuna-13B, with only a marginal difference to ideal responses, indicating a strong potential in understanding and implementing continuous coordination tasks. This makes GPT-4 suitable for developing Coordination Agents out of the box, with minimal engineering.

## 7.2 Sustained Coordination

Sustained coordination refers to the ability of agents to continuously collaborate and adapt their actions over extended periods. Robustness to Partners is about an agent’s ability to adjust and adapt to interacting with new or unseen partners. We use GPT-4 as the LLM of choice to test these aspects. The choice of LLM is dictated by the fact that only GPT-4 is able to display satisfactory reasoning that will be required consistently for Sustained Coordination. We evaluate the LLM-Co agent on 400 timesteps of gameplay in the Overcooked-AI environment. The evaluation metric used in Overcooked [5] is the sparse reward obtained when one whole delivery is completed by the agents. Each delivery wins the agents 20 points.

### 7.2.1 Baselines

We use Self Play with Proximal Policy Optimization (PPO) and Population-Based Training with PPO as the baselines for comparing AI-AI gameplay.

For benchmarking AI-Human Proxy gameplay, we pair our agents with Behavior Cloning model trained on actual human trajectories collected by [5]. As a baseline

Agent Type	Layouts				
	Cramped Rm.	Asymm. Adv.	Coord. Ring	Forced Coord.	Counter Circ.
PPO <sub>SP</sub>	198.8 ± 4.06	167.2 ± 3.63	<b>190.8 ± 4.25</b>	151.9 ± 3.28	122.3 ± 3.80
PBT	216.9 ± 1.31	190.1 ± 8.64	173.8 ± 18.27	169.5 ± 10.09	140.1 ± 13.86
LLM-Co	<b>220 ± 0</b>	<b>280 ± 0</b>	180 ± 0	<b>200 ± 0</b>	<b>160 ± 0</b>

Table 7.1: Comparison of game play between self-play baselines (PPO, and PBT) and LLM-Co Agents. LLM-Co agents outperform RL methods on 4 out of 5 layouts, demonstrating highly effective reasoning under sustained coordination.

for comparison, we use the method of using a **PPO agent trained with a human model** established in [5] and observed in the follow-up works [17, 47, 19, 44] approaching Zero Shot Coordination.

### 7.2.2 Results

**The LLM-Co agent efficiently completes the Overcooked-AI challenge over a long horizon** We pair two LLM-Co agents together to jointly coordinate and complete the cooking and delivery task in Overcooked-AI. This is analogous to testing agents trained with self-play methods being asked to jointly perform the task. We observe through visualizations of the gameplay that LLM-Co agents make effective use of all resources available to them to complete multiple deliveries effectively. In fact, without being trained or fine-tuned for the task, LLM-Co agents outperform or nearly match Self-Play baselines trained using Proximal Policy Optimization [30] or Population-Based Training [11] which are the gold standard for Multi-Agent Tasks on. Table 7.1 shows

a numerical summary of the scores obtained by agents. These scores represent averages obtained from 100 runs across with standard deviation across 5 seeds for MARL agents. For LLM-Co agents, the score obtained by agents for a fixed game description and directives remains the same as the agent always chooses to take the same medium-level action for a given state and history. This outcome is noteworthy because it demonstrates that Language Learning Models (LLMs), specifically GPT-4 [22] in this case, can outperform RL agents at cooperative multi-agent tasks with minimal scaffolding. We observed that LLM Agents are capable of achieving sustained coordination, adjusting to their partners, and correcting their own actions consistently.

**The LLM-Co agent generates explainable outputs through free-text** Reinforcement Learning (RL-based) agents cannot provide an **underlying rationale** for their actions, making it challenging to understand how their actions contribute to broader objectives. This understanding is crucial for the development of safer and more reliable agents, as well as for debugging when unexpected behaviors occur. The LLM-Co agent addresses this gap by generating medium-level actions and providing high-level reasoning for such a selection. This allows us to extract comprehensive insights into the decision-making processes under given conditions by examining the "explanation" generated by the language model during gameplay. We observed that LLM-Co agents (w. GPT-4) always select an action from the provided set of feasible actions without hallucinations across all layouts. The explanation itself serves as Chain-of-Thought strategy for enabling better reasoning in the LLM.

Agents	Layouts				
	Cramped Rm.	Asymm. Adv.	Coord. Ring	Forced Coord.	Counter Circ.
BC	103.5 $\pm$ 3.38	136.5 $\pm$ 7.00	59.0 $\pm$ 5.38	20.5 $\pm$ 4.33	38.0 $\pm$ 3.99
PPO <sub>BC</sub>	156.4 $\pm$ 1.48	72.6 $\pm$ 19.44	126.4 $\pm$ 3.24	58.9 $\pm$ 2.98	69.5 $\pm$ 2.18
LLM-Co	<b>160 <math>\pm</math> 0</b>	<b>180 <math>\pm</math> 0</b>	<b>160 <math>\pm</math> 0</b>	<b>120 <math>\pm</math> 0</b>	<b>140 <math>\pm</math> 0</b>
<b>Playing from swapped positions:</b>					
BC	110.0 $\pm$ 3.39	137.5 $\pm$ 8.40	70.0 $\pm$ 4.00	31.0 $\pm$ 5.00	44.0 $\pm$ 3.02
PPO <sub>BC</sub>	163.9 $\pm$ 1.61	<b>178.8 <math>\pm</math> 2.65</b>	129.8 $\pm$ 3.59	76.9 $\pm$ 2.29	57.6 $\pm$ 2.50
LLM-Co	<b>180 <math>\pm</math> 0</b>	140 $\pm$ 0	<b>160 <math>\pm</math> 0</b>	<b>80 <math>\pm</math> 0</b>	<b>120 <math>\pm</math> 0</b>

Table 7.2: Comparison of AI-Human Proxy Game play. We compare Behavior Cloning Agents, PPO<sub>BC</sub> Agents with LLM-Co agents utilizing the GPT-4 LLM. The LLM-Co agents are able to outperform or match the performance of Reinforcement Learning models, indicating that LLM agents are robust to the choice of partner agents.

## 7.3 Robustness to Partners

### 7.3.1 Results

**The LLM-Co agent is robust to the choice of partner.** It is highly likely that an agent is paired up with a biased or sub-optimal partner. It is known that self-play agents, when paired with humans, tend to struggle because their behavior diverges from what they consider to be the optimal strategy [5]. The LLM-Co agent, on the other hand, does not face this issue. Their actions are based on verbal reasoning, and they adapt to the current situation rather than adhering to a determined policy. Consequently,

they outperform Self-play-based methods trained with human data at AI-human proxy gameplay as shown in table 7.2. The table describe the average score obtained by agents over 400 timesteps when paired up with the proxy human agent from both positions.

## 7.4 Explicit Assistance

Conditions	Layouts	
	Locked	Gated Delivery
Without Helper Directive	160	0
With Helper Directive	240	180

Table 7.3: Comparison of Gameplay in the Overcooked-Assistance Layouts with and without Helper Directive. The results indicate that the Large Language Model needs to be prompted to be aware of situations where their partner might need assistance in order to be effective in the Overcooked-Assistance layouts.

Finally, we test the ability of the LLM-Co agent to provide explicit assistance to their partners in the new Overcooked layouts defined in 5.4, where proactive help is necessary to complete the task.

### 7.4.1 Results

**the LLM-Co agent requires a helping directive to choose to help** The LLM-Co agent, provided with the same prompt and directives as used in Overcooked-AI, struggles to recognize and help their partner agents in Locked and Gated Delivery environments. However, a simple directive informing the LLM-Co agent to "help their

partners when the situation demands” makes them actively look for opportunities to help their partners. We see that agents tend to help partner agents during the time they are waiting for their own soup to be cooked by choosing the open gates for the waiting agent. While this is not the most efficient strategy, which would have been to always help out a waiting agent, it still points to the agent’s ability to explicitly help out during coordination. Table 7.3 shows scores obtained by agents over 400 time steps.

Agents	Layouts	
	Locked	Gated Delivery
PPO <sub>SP</sub>	132.83 ± 7.31	134.88 ± 5.99
PBT	175.8 ± 1.69	178.6 ± 9.76
LLM-Co	<b>220 ± 0</b>	<b>180 ± 0</b>

Table 7.4: Comparison of Gameplay on Overcooked-Assistance Layouts between RL baselines and LLM Agents. The RL baselines being able to effectively solve the deliveries indicates that the environments are solvable through self-play training. The high scores achieved by LLM agents demonstrate that LLM agents are capable of reasoning for providing explicit assistance to their partners.

### The LLM-Co agent outperforms MARL methods at Overcooked-Co-op lay-

**outs** Table 7.4 shows the performance of Self Play agents trained using PPO and PBT

and compares it with the abilities of the LLM-Co agent provided with a helper directive.

Since a reward is provided to both agents for delivery in Self Play training, we expected

to see them gain the ability to open gates as it results in a reward after a short delay.

In spite of this, the LLM-Co agent has the upper hand in their ability to deduce the

right actions required to facilitate their partners.



# Chapter 8

## Case Studies

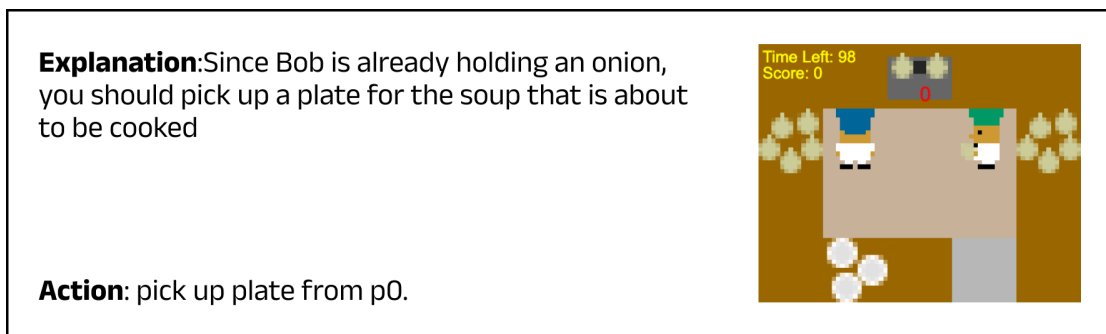


Figure 8.1: The LLM-Co agent (Blue) understands that its partner has already picked up an onion and will probably end up placing it in the cooker to complete the requirement. So, it chooses to "pick up a plate" which would help in the next stage.

**LLMs perform long-term reasoning while considering their partner's intentions.** The Overcooked-AI environment and cooperative tasks in general require agents to plan ahead while considering their partner agents' intentions. Figure 8.1 provides a visual summary of the LLM-Co agent's rationale behind selecting the action "pick up a plate from p0" when there are already two onions in the pot with the part-

ner agent having already picked up the third one. The LLM-Co agent reasons that their partner is probably going to place their onion in the cooker next and consequently decides to take the preemptive action of picking up a plate.

**Explanation:** You already have a plate with soup in your inventory, and c0 is still cooking. Bob is holding an onion and can add it to c1. Your best action is to deliver the soup you have to maximize deliveries. Bob's next action should be to add the onion to c1 and start cooking.

**Action:** move away.




Figure 8.2: The LLM-Co Agent (Blue) thinks about the partner’s mind states (shown in red) and makes an informed decision (highlighted in green) to move away to make way for the partner agent)

**LLM-Co Agents enable their partners.** Effective coordination requires agents to have an understanding of how they might be inhibiting their partners. Figure 8.2 shows the LLM-Co agent’s reasoning in a situation where two cookers are located in a corner (Coordination Ring environment), and one of them is cooking the soup while the other is missing some onions. Since the soup is not cooked, the agent with a plate is just waiting near the cooker to load the soup onto the plate. However, this time could be used by the other agent to put an onion in the adjoining cooker. In this situation, the LLM-Co agent is able to realize that it needs to move away to enable its partner.

**LLM-Co agents are sensitive to their partner’s actions and correct their behaviors based on the resulting state changes.** In the Overcooked-AI environ-

**Explanation:** Both cookers are already full and cooking, and holding onto the onion will prevent you from performing other actions. Placing the onion on the nearest counter will free up your hands for the next task.

**Action:** place onion on k5.

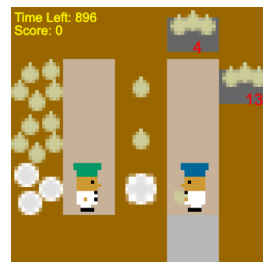


Figure 8.3: The LLM-Co Agent (Blue) realizes that the onion it picked up cannot be used and decides to correct itself by placing it on the kitchen counter to free up its hands.

ment, each agent gets to perform one action at every time step. This leads to situations where an agent might end up taking an action that seems optimal in the situation but is soon realized to be sub-optimal due to their partner's action. A situation like this is demonstrated in Figure 8.3. The Blue agent has picked up an onion but needs a plate. In such a situation we observe that the agent is able to realize its sub-optimal action selection and notice that a plate has been made available by the other agent. In this case, the LLM-Co agent places the object on the nearest available counter, freeing its hands. In the subsequent step, it goes and picks up a plate to load the soup.

**Explanation:** Since the soup in c0 is still cooking and Bob is already holding a cooked soup but can't deliver it because d0 is inaccessible to him, the best action would be to open g0 so Bob can deliver his soup and you can continue cooking.

**Action:** open g0.



Figure 8.4: The LLM-Co Agent (Blue) recognizes that its partner is stuck behind a closed door and chooses to open the gate g0 to enable them.

**LLM-Co agents explicitly help out their partners by opening gates** Figure 8.4 shows a visual summary of the reasoning followed by LLM-Co agents before choosing to open gates. It can be clearly seen that they realize the imminent rise in efficiency that will result from allowing their partners to access the delivery areas and recognize that the right way to do this is by opening the gate for them.

## Chapter 9

### Limitations

While LLM agents are a promising avenue for developing coordination agents, it is important to point out some limitations of these methods. First, LLM agents are much more expensive, in terms of latency as well as memory, compared to Reinforcement Learning agents, which are usually much smaller models. This limitation makes it difficult to utilize LLM Agents off the shelf in an online system.

Secondly, as of now, the action space of the LLM is hand-curated, as LLMs are not the best option for directly generating low-level control actions due to both latency and their inability to navigate in an environment.

# Chapter 10

## Conclusion

In this study, we probe the reasoning abilities of Large Language Models for achieving Multi-agent coordination. We introduce the LLM-Co Framework to augment large language models to complete multi-agent coordination games. We evaluate LLM Agents across five aspects of Coordination, namely Theory of Mind Inference, Situated Reasoning, Sustained Coordination, Robustness to Partners, and the ability to provide Explicit Assistance. We also designed the LLMTom-Reasoning set to assess and compare the Theory of Mind inference and Situated Reasoning Abilities of various Large Language Models. We show that LLM-Coordination Agents are capable of Sustained Coordination in the Overcooked Environment and are Robust to the choice of partner. Finally, we introduce two new layouts to the Overcooked-AI environment and demonstrate the ability and conditions required for Large Language Models to provide explicit assistance to their partners during coordination games.

# Bibliography

- [1] 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. The hanabi challenge: A new frontier for ai research. *Artificial Intelligence*, 280:103216, 2020.
- [2] 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. The hanabi challenge: A new frontier for ai research. *Artificial Intelligence*, 280:103216, 2020.
- [3] Craig Boutilier. Planning, learning and coordination in multiagent decision processes. In *Proceedings of the 6th Conference on Theoretical Aspects of Rationality and Knowledge*, TARK '96, page 195–210, San Francisco, CA, USA, 1996. Morgan Kaufmann Publishers Inc.
- [4] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Pra-

fulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners, 2020.

- [5] Micah Carroll, Rohin Shah, Mark K. Ho, Thomas L. Griffiths, Sanjit A. Seshia, Pieter Abbeel, and Anca Dragan. *On the Utility of Learning about Humans for Human-AI Coordination*. Curran Associates Inc., Red Hook, NY, USA, 2019.
- [6] Micah Carroll, Rohin Shah, Mark K. Ho, Thomas L. Griffiths, Sanjit A. Seshia, Pieter Abbeel, and Anca Dragan. `overcooked_ai`. [https://github.com/HumanCompatibleAI/overcooked\\_ai/tree/master](https://github.com/HumanCompatibleAI/overcooked_ai/tree/master), 2019.
- [7] 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. Vicuna: An open-source chatbot impressing gpt-4 with 90%\* chatgpt quality, March 2023.
- [8] Chuang Gan, Jeremy Schwartz, Seth Alter, Damian Mrowca, Martin Schrimpf, James Traer, Julian De Freitas, Jonas Kubilius, Abhishek Bhandwaldar, Nick Haber, Megumi Sano, Kuno Kim, Elias Wang, Michael Lingelbach, Aidan Curtis, Kevin Feiglis, Daniel M. Bear, Dan Gutfreund, David Cox, Antonio Torralba, James J. DiCarlo, Joshua B. Tenenbaum, Josh H. McDermott, and Daniel L. K.



- Yamins. Threedworld: A platform for interactive multi-modal physical simulation, 2021.
- [9] Hengyuan Hu, Adam Lerer, Alex Peysakhovich, and Jakob Foerster. "other-play" for zero-shot coordination, 2021.
- [10] Wenlong Huang, Pieter Abbeel, Deepak Pathak, and Igor Mordatch. Language models as zero-shot planners: Extracting actionable knowledge for embodied agents. *arXiv preprint arXiv:2201.07207*, 2022.
- [11] 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. Population based training of neural networks, 2017.
- [12] Unnat Jain, Luca Weihs, Eric Kolve, Ali Farhadi, Svetlana Lazebnik, Aniruddha Kembhavi, and Alexander G. Schwing. A cordial sync: Going beyond marginal policies for multi-agent embodied tasks. In *ECCV*, 2020. first two authors contributed equally.
- [13] Unnat Jain, Luca Weihs, Eric Kolve, Mohammad Rastegari, Svetlana Lazebnik, Ali Farhadi, Alexander G. Schwing, and Aniruddha Kembhavi. Two body problem: Collaborative visual task completion. In *CVPR*, 2019. first two authors contributed equally.

- [14] Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large language models are zero-shot reasoners, 2023.
- [15] Michal Kosinski. Theory of mind might have spontaneously emerged in large language models, 2023.
- [16] Wenhao Li, Dan Qiao, Baoxiang Wang, Xiangfeng Wang, Bo Jin, and Hongyuan Zha. Semantically aligned task decomposition in multi-agent reinforcement learning, 2023.
- [17] Yang Li, Shao Zhang, Jichen Sun, Yali Du, Ying Wen, Xinbing Wang, and Wei Pan. Cooperative open-ended learning framework for zero-shot coordination. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett, editors, *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, 2023.
- [18] Jacky Liang, Wenlong Huang, Fei Xia, Peng Xu, Karol Hausman, Brian Ichter, Pete Florence, and Andy Zeng. Code as policies: Language model programs for embodied control. In *arXiv preprint arXiv:2209.07753*, 2022.
- [19] Xingzhou Lou, Jiaxian Guo, Junge Zhang, Jun Wang, Kaiqi Huang, and Yali Du. 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, 2023. International Foundation for Autonomous Agents and Multiagent Systems.
- [20] Ryan Lowe, Yi Wu, Aviv Tamar, Jean Harb, Pieter Abbeel, and Igor Mordatch. 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, 2017. Curran Associates Inc.
- [21] Zhao Mandi, Shreeya Jain, and Shuran Song. Roco: Dialectic multi-robot collaboration with large language models, 2023.
- [22] OpenAI. Gpt-4 technical report, 2023.
- [23] 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. Training language models to follow instructions with human feedback, 2022.
- [24] 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. Training language models to follow instructions with human feedback, 2022.

- [25] Joon Sung Park, Joseph C. O’Brien, Carrie J. Cai, Meredith Ringel Morris, Percy Liang, and Michael S. Bernstein. Generative agents: Interactive simulacra of human behavior, 2023.
- [26] David Premack and Guy Woodruff. Does the chimpanzee have a theory of mind? *Behavioral and Brain Sciences*, 1(4):515–526, 1978.
- [27] Xavier Puig, Tianmin Shu, Shuang Li, Zilin Wang, Yuan-Hong Liao, Joshua B. Tenenbaum, Sanja Fidler, and Antonio Torralba. Watch-and-help: A challenge for social perception and human-ai collaboration. In *International Conference on Learning Representations*, 2021.
- [28] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding with unsupervised learning. 2018.
- [29] Shreyas Sundara Raman, Vanya Cohen, Eric Rosen, Ifrah Idrees, David Paulius, and Stefanie Tellex. Planning with large language models via corrective re-prompting, 2022.
- [30] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms, 2017.
- [31] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dhharshan Kumaran, Thore Graepel, Timothy Lillicrap, Karen Simonyan, and Demis Hassabis. A general rein-

- forcement learning algorithm that masters chess, shogi, and go through self-play. *Science*, 362(6419):1140–1144, 2018.
- [32] Chan Hee Song, Jiaman Wu, Clayton Washington, Brian M Sadler, Wei-Lun Chao, and Yu Su. Llm-planner: Few-shot grounded planning for embodied agents with large language models. *arXiv preprint arXiv:2212.04088*, 2022.
- [33] DJ Strouse, Kevin McKee, Matt Botvinick, Edward Hughes, and Richard Everett. Collaborating with humans without human data. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 14502–14515. Curran Associates, Inc., 2021.
- [34] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. Llama: Open and efficient foundation language models, 2023.
- [35] 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. Grandmaster level in StarCraft II using multi-agent reinforcement learning. *Nature*, 575(7782):350–354, 2019.
- [36] Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An open-ended embodied agent with large language models, 2023.
- [37] Rose E. Wang, Sarah A. Wu, James A. Evans, Joshua B. Tenenbaum, David C. Parkes, and Max Kleiman-Weiner. Too many cooks: Coordinating multi-agent collaboration through inverse planning. In *Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems*, AAMAS '20, page 2032–2034, Richland, SC, 2020. International Foundation for Autonomous Agents and Multiagent Systems.
- [38] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems*, 35:24824–24837, 2022.
- [39] Lilian Weng. Llm-powered autonomous agents. *lilianweng.github.io*, Jun 2023.
- [40] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Can-

- wen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. Huggingface’s transformers: State-of-the-art natural language processing, 2020.
- [41] Sarah A. Wu, Rose E. Wang, James A. Evans, Joshua B. Tenenbaum, David C. Parkes, and Max Kleiman-Weiner. Too many cooks: Bayesian inference for coordinating multi-agent collaboration. *Topics in Cognitive Science*, 13(2):414–432, 2021.
- [42] Yue Wu, Shrimai Prabhunoye, So Yeon Min, Yonatan Bisk, Ruslan Salakhutdinov, Amos Azaria, Tom Mitchell, and Yuanzhi Li. Spring: Gpt-4 out-performs rl algorithms by studying papers and reasoning, 2023.
- [43] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models, 2023.
- [44] Chao Yu, Jiaxuan Gao, Weilin Liu, Botian Xu, Hao Tang, Jiaqi Yang, Yu Wang, and Yi Wu. 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, 2023.
- [45] 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. Proagent: Building proactive cooperative ai with large language models, 2023.

- [46] Hongxin Zhang, Weihua Du, Jiaming Shan, Qinhong Zhou, Yilun Du, Joshua B. Tenenbaum, Tianmin Shu, and Chuang Gan. Building cooperative embodied agents modularly with large language models, 2023.
- [47] Rui Zhao, Jinming Song, Yufeng Yuan, Haifeng Hu, Yang Gao, Yi Wu, Zhongqian Sun, and Wei Yang. Maximum entropy population-based training for zero-shot human-ai coordination. *Proceedings of the AAAI Conference on Artificial Intelligence*, 37(5):6145–6153, Jun. 2023.



# Appendix A

## Prompt Details

### A.1 Overcooked Game Description Prompts

#### Overcooked Task Description Prompt

In the game Overcooked, I am Alice, my teammate is Bob.

LAYOUT\_DESCRIPTION.

We must coordinate to make onion soups with 3 onions each. Once a soup is cooked it needs to be placed on a plate and delivered. I can only carry one item at a time. My goal is to maximize the number of deliveries. I want to be efficient and prepare for the next soup while the current soup is cooking. 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 next action including a prediction of Bob's next action⟩. Action: ⟨action⟩. Only select one action. Do not say anything else. Got it?

## **Overcooked Task Description Prompt with Helper Directive**

In the game Overcooked, I am Alice, my teammate is Bob.

LAYOUT\_DESCRIPTION.

We must coordinate to make onion soups with 3 onions each. Once a soup is cooked it needs to be placed on a plate and delivered. I can only carry one item at a time. My goal is to maximize the number of deliveries. I want to be efficient and prepare for the next soup while the current soup is cooking. I'll provide my action history, current state, teammate's status, and my possible actions. I want to prefer helping the other player with their cooking and delivery if the situation arises. Help me select the best action from the list. Format your response as: Explanation:⟨Brief explanation for next action including a prediction of Bob's next action⟩. Action: ⟨action⟩. Only select one action. Do not say anything else. Got it?

## **A.2 Overcooked Layout Prompts**

### **Cramped Room**

The environment is rectangular with 2 onions dispensers (o0, o1), cooker (c0), plate dispenser (p0) and delivery area (d0). Additionally there are kitchen counters (k0 to k8) which can be used to temporarily store onions and plates while you do something else. Objects on counters can be picked up later and should be considered as they may be closer than items in dispensers.

### **Asymmetric Advantages**

There are two partitions in the current environment. Bob is in the left partition with access to onion dispenser (o0), delivery area (d0), plate dispenser (p0) and kitchen counters (k0, k1, k2, k3, k4, k11, k12, k16, k18, k20, k21, k22, k23). Alice is in the right partition and has access to onion dispenser (o1), delivery area (d1), plate dispenser (p1) and kitchen counters (k6, k7, k8, k9, k10, k14, k15, k17, k19, k25, k26, k27, k28). Both have access to both cookers (c0, c1). Kitchen counters (k0 to k28) can be used to temporarily store onions and plates while you do something else. Objects on counters can be picked up later and should be considered as they may be closer than items in dispensers.

### **Forced Coordination**

The environment is split into two partitions, one with each player. In the right partition, Alice has access to cookers (c0, c1), delivery area (d0) and kitchen counters (k6, k8, k12). In the left partition, Bob has access to onion dispensers (o0, o1), plate dispenser (p0) and kitchen counters (k1, k10). Kitchen counters can be used to temporarily store onions and plates while you do something else. Both players have access to shared counters (s0, s1, s2) which can be used to transfer onions and plates depending on the situation. Note that the objects on the shared counters can be accessed by both players. Objects on counters can be picked up later and should be considered as they may be closer than items in dispensers.

### **Coordination ring**

The environment is narrow and circular, with onion dispensers (o0, o1), plate dispenser (p0), cookers (c0, c1), and a delivery area (d0). Additionally there are kitchen counters

(k0 to k10) which can be used to temporarily store onions and plates while you do something else. Objects on counters can be picked up later and should be considered as they may be closer than items in dispensers.

### **Counter Circuit**

The environment is circular with two onion dispensers (o0, o1), plate dispenser (p0), cookers (c0, c1) and delivery area (d0). There are also the shared counters (s0, s1, s2, s3) which can be used to pass objects from one player to the other. Additionally there are kitchen counters (k0 to k15) which can be used to temporarily store onions and plates while you do something else. Objects on counters can be picked up later and should be considered as they may be closer than items in dispensers.

### **Gated Delivery**

The environment is rectangular with 2 onions dispensers (o0, o1), cooker (c0) and plate dispenser (p0). The delivery area (d0) is inaccessible behind closed gates and can be accessed by opening one of the gates (g0, g1). A gate can only be opened by a player if they are not carrying an object. Once the gate is opened it will only stay open for a brief time and then close on its own.

### **Locked**

The environment is divided into 2 partitions. Alice is in the left partition with access to onion dispenser o0, plate dispenser p1, cooker c0, and delivery area d0. Bob is in the right partition with access to onion dispenser o1, plate dispenser p0, and cooker c1. The two partitions are connected by gate g0 which can be opened if a player is holding nothing. Opening the gate will allow players to move freely between partitions. The

gate will close after enough time automatically.

# Appendix B

## Details of LLMs used in Experiments

We use the following Large Language Models in our Experiments

- GPT-4: We use the gpt-4-0613 [22] model from OpenAI through their API.
- ChatGPT: We use the gpt-3.5-turbo-0613 [23] model from OpenAI through their API.
- Vicuna13b, Vicuna33b: The Vicuna models developed by [7] are based on the LLaMA [34] architecture. We use the huggingface [40] library to implement the Vicuna models.