# LLM-Powered Hierarchical Language Agent for Real-time Human-AI Coordination

Anonymous Author(s)
Submission Id: 214

## ABSTRACT

AI agents powered by Large Language Models (LLMs) have made significant advances, enabling them to assist humans in diverse complex tasks and leading to a revolution in human-AI coordination. LLM-powered agents typically require invoking LLM APIs and employing artificially designed complex prompts, which results in high inference latency. While this paradigm works well in scenarios with minimal interactive demands, such as code generation, it is unsuitable for highly interactive and real-time applications, such as gaming. Traditional gaming AI often employs small models or reactive policies, enabling fast inference but offering limited task completion and interaction abilities. In this work, we consider Overcooked as our testbed where players could communicate with natural language and cooperate to serve orders. We propose a Hierarchical Language Agent (HLA) for human-AI coordination that provides both strong reasoning abilities while keeping real-time execution. In particular, HLA adopts a hierarchical framework and comprises three modules: a proficient LLM, referred to as Slow Mind, for intention reasoning and language interaction, a lightweight LLM, referred to as Fast Mind, for generating macro actions, and a reactive policy, referred to as Executor, for transforming macro actions into atomic actions. Human studies show that HLA outperforms other baseline agents, including slow-mind-only agents and fast-mind-only agents, with stronger cooperation abilities, faster responses, and more consistent language communications.

## KEYWORDS

Large Language Models, Language Agents, Real-time Human-AI Coordination, Hierarchical Reasoning and Planning

**ACM Reference Format:**
Anonymous Author(s). 2024. LLM-Powered Hierarchical Language Agent for Real-time Human-AI Coordination. In *Proc. of the 23rd International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2024), Auckland, New Zealand, May 6 – 10, 2024*, IFAAMAS, 35 pages.

## 1 INTRODUCTION

Developing Artificial Intelligence (AI) agents that can attain human-level performance has been a long-standing goal for AI research [20, 59]. Large Language Models (LLMs) [41] have emerged as promising tools in this endeavor, owing to their strong reasoning and generalization abilities. LLM-powered AI agents have exhibited significant potential across diverse domains, including code generation [30, 34, 46, 50], content creation [42, 44, 45], tool utilization [37, 43, 47, 70], and robotics [5, 13, 31, 54]. Meanwhile, LLM-powered agents have exhibited the ability to mimic human-like behaviors when interacting with other players [18, 42], leading to a revolution in human-AI coordination.

AI agents powered by LLMs commonly rely on LLM APIs and hand-crafted complex prompts. A notable challenge within this paradigm is the high latency associated with the inference process due to API calls, ranging from seconds to minutes [4]. The inference time may not be regarded as bottleneck in scenarios with low-frequency interactions, such as code generation. However, the limitation of high inference latency becomes apparent in human-AI coordination applications, which requires real-time responses and high-frequency interactions, such as video games [42, 54, 57].

In this work, we consider Overcooked as our real-time human-AI coordination testbed. Overcooked is a cooperative cooking game where players collaborate at a rate of approximately 3Hz to serve orders within a time budget. We further develop language-based communication in Overcooked to allow human-like cooperation. An ideal language agent is expected to exhibit real-time responsiveness, strong reasoning capabilities, and effective language-based communication with human players for the best performance in such a fast-paced game. Fig. 1 shows a concrete example. When paired with an AI player, a human player might instruct the AI player by saying “Chop 3 tomatoes.” The AI player needs to accurately interpret and follow the command by swiftly picking up and chopping the specified number of tomatoes. Upon completion, the AI player needs to inform the human player of the ongoing progress, e.g., by saying “I’ve chopped 3”, for future cooperation. Afterward, the human player might directly say “one more”, which itself looks semantically ambiguous. The AI player must correctly infer the true command from the history commands and promptly chop one more tomato. Traditional gaming AI usually employs smaller models or script policies, emphasizing fast inference for real-time responses to the game dynamics. Yet, this efficiency comes at the cost of limiting task completion and intra-player interaction abilities [3, 17, 51, 67, 69].

We propose a *Hierarchical Language Agent (HLA)* for real-time human-AI coordination in Overcooked. Inspired by System 1 and System 2 thinking [28], HLA combines both robust reasoning and interaction capabilities from a large model and real-time inference from a smaller model and a reactive policy. In particular, HLA employs a hierarchical framework that consists of three modules: a proficient LLM for intention reasoning and language-based communication, a lightweight LLM for interpreting commands and high-level planning, and a script policy for executing low-level actions swiftly. We denote the proficient LLM as *Slow Mind*. The lightweight LLM, referred to as *Fast Mind*, generates macro actions, and the reactive script policy is referred to as *Executor*, which transforms macro actions into atomic actions. Human commands are processed simultaneously through both the Slow Mind and Fast Mind to enhance real-time performance. The reactive policy further ensures the feasibility of actions and high-frequency interactions.

*Proc. of the 23rd International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2024), N. Alechina, V. Dignum, M. Dastani, J.S. Sichman (eds.), May 6 – 10, 2024, Auckland, New Zealand.* © 2024 International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org). This work is licenced under the Creative Commons Attribution 4.0 International (CC-BY 4.0) licence.

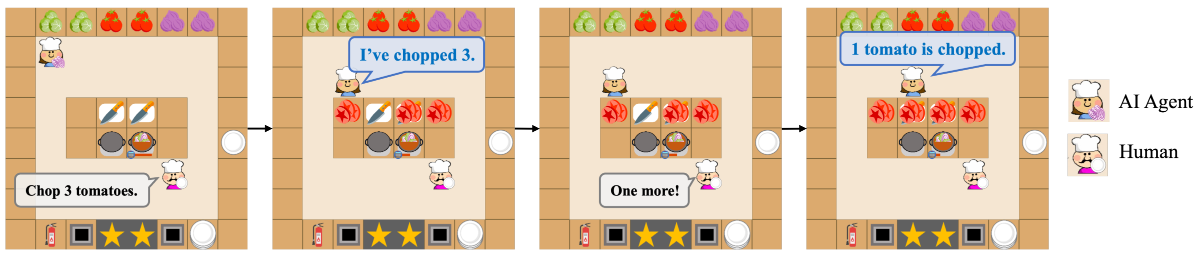


**Figure 1: A concrete example of cooperation and communication between a human player and an AI player in Overcooked.**

We consider three baseline agents, each lacking a specific HLA component. Our evaluation involves experiments on action latency, reasoning with simple and complex commands, and human studies. HLA demonstrates a remarkable advantage, being **an order of magnitude faster** than the best competitor in real-time action responsiveness. Furthermore, HLA outperforms the baseline agents significantly in command reasoning ability. Human studies confirm these findings, showing that HLA achieves approximately **50% higher** game scores and receives the highest human preference.

## 2 RELATED WORKS

**Language Agents.** Prior works train instruction-following agents with paired datasets of text and trajectories in games [15, 62], visual navigation [22, 61] and robotics [25, 27]. However, these works are limited to simple domains. Recently, with the advances of Large Language Models, a class of works start to utilize the strong reasoning power of LLMs to interact with complex domains including web scenarios [11, 38, 65], simulated environments [24, 42, 57, 71], real-world environments [5, 26, 53, 72]. These recent efforts use prompt engineering to elicit the power of LLMs [58, 66]. Some recent works try to combine LLMs that play different functionalities in a cooperative manner [8, 9, 29, 54]. In our work, we study the availability of language agents in response speed sensitive environments.

**Human-AI Cooperation.** Building AI agents that can cooperate with humans is a longstanding challenge. Prior works study human-AI cooperation without communication in games such as Hanabi [10, 23] and Overcooked [51, 67, 69], and in robotics [3, 17]. Language commands from humans are used to guide intelligent agents in visual navigation [22, 61] and robotics [25, 27]. Cicero [15] trains a language agent that can speak and make decisions like a human in the game of Diplomacy from a large volume of human play data. However, Diplomacy is a turn-based game and therefore has low real-time requirements. Recently, there are attempts in using LLM for human-AI interaction in domains including web scenarios [11, 21], health [2, 64], games [54, 57] and other applications [19, 33, 48]. Similar to us, there are some concurrent works that use LLM for decision-making in the game of Overcooked [1, 63, 67, 68]. We focus on improving real-time user experience in settings that demand rapid responses during human-AI interaction.

**LLM Inference Speedup.** The long inference latency of existing LLM-based agents is not desirable in domains that demand a fast response speed. “Skeleton-Of-Thought” [39] generates responses with a skeleton structure and uses parallel inference to reduce latency. Model compression methods achieve faster inference by distilling the knowledge of LLMs into smaller language models [6, 12, 55, 56, 58] and performing quantization [16, 32, 36]. In our work, we adopt a hierarchical design that leverages LLMs for reasoning and language interaction and small models for fast reaction in real-time gaming.

## 3 TESTBED: THE OVERCOOKED GAME

### 3.1 Environment Details

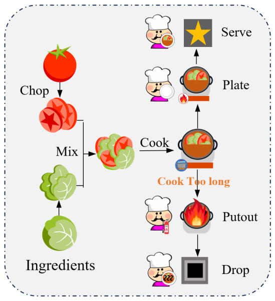
Overcooked is a cooperative cooking game where participants must work together to prepare, cook, and promptly serve a variety of dishes. The Overcooked environment [7, 60] simulates and simplifies the original Overcooked game and provides an RL training interface as a common testbed for real-time human-AI coordination. Throughout the game, orders continuously appear, each accompanied by a strict deadline. Players must prepare dishes in accordance with these orders and ensure they are delivered on time for rewards. Failed deliveries incur penalties. To finish an order, players must follow a precise sequence of steps: retrieving the necessary ingredients, chopping them, mixing them as per the recipe, cooking them to make a dish, plating the dish before it overcooks, and finally, serving it at the counter. Fig. 2a depicts the cooking process of the game. Each player can perform one of the 4 actions at each time step, i.e., *up, down, left, and right*, to control the character’s position and to interact with other objects.

We implemented a collection of enhancements to the environment with full details described in Appendix A. In particular, we further extend the original environment by developing a chat interface to allow natural language communication between human and AI players. Human players can either pause the game and send text messages to the AI through an additional chat window or directly speak to the AI player during gameplay. Similarly, the AI player can respond to human players through either text or speech.

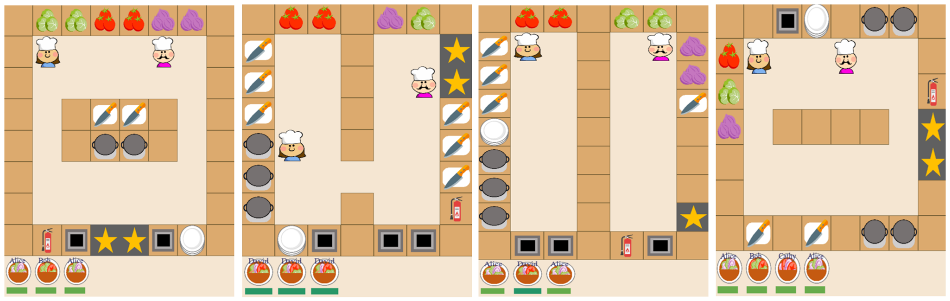
We also design 4 distinct maps as shown in Fig. 2b. In each game, a human player (the pink beard character) collaborates with an AI player (the blue character) to complete the orders. The first two maps, *Ring* and *Bottleneck*, assess general human-AI cooperation capabilities. The third map, *Partition*, completely separates the two players, therefore they must cooperate to finish any order. In the fourth map, *Quick*, we raise the number of concurrent orders and accelerate the action frequency to intensify the gameplay.

### 3.2 Challenges of Overcooked

**Real-time Cooperation.** The Ovecooked game is particularly time-sensitive. It requires all players to adjust their game strategy in time and take swift actions, so that the orders do not timeout, and the dishes are not overcooked. The real-time requirement of the



(a) Cooking process.



(b) Designed maps. From left to right are *Ring*, *Bottleneck*, *Partition* and *Quick*

**Figure 2: The cooking process and the maps in the Overcooked testbed.**

game indicates that, if an AI agent wants to leverage LLM's strong capabilities, it cannot adopt the conventional approach of directly prompting the LLM to generate its next action. This approach leads to substantial latency, making it impractical for this game.

**Command Reasoning.** When playing the original Overcooked game, human players often give commands that are semantically ambiguous or overly complex. We collect some common human commands and present the typical scenarios as follows.

- Quantity specification. The player asks others to "chop 3 tomatoes."
- Semantic analysis. Instead of giving direct commands such as "cook the soup," the player gives hints, "the soup order is about to timeout."
- Ambiguous reference. The player first asks others to "chop some tomatoes," then says "one more." Here the "one" implicitly refers to a chopped tomato.

In such scenarios, the AI agent must consider various factors, such as human commands, environment composition, and historical actions, to determine the true intentions of the human player and respond accordingly. A basic reactive agent lacking reasoning capabilities will struggle to comprehend human commands, let alone collaborate effectively with humans to complete the game.

## 4 METHOD

### 4.1 Overview

As described in Sec. 3, the AI agent needs real-time responsiveness, command reasoning ability, and bilateral communication capability. A key observation in our testbed is that the AI agent can perform reasoning about human commands and communicate with humans at a regular frequency while atomic actions should be generated at a high frequency. Meanwhile, LLM-based agents are better at generating high-level moves than producing atomic actions. Therefore we propose *Hierarchical Language Agent (HLA)* as depicted in Fig. 3. HLA consists of three components: a proficient LLM, i.e., *Slow Mind*, that interprets human commands from the full game history and generates chat feedback; a lightweight LLM, i.e., Fast Mind, that delivers high-level moves, which we call "macro actions", at a medium frequency; and a reactive policy, i.e., *Executor*, that is implemented as pre-defined scripts and transforms macro actions into atomic actions to interact with the environment at a high frequency. We will describe Slow Mind, Fast Mind, and Executor in the following sections respectively.

### 4.2 Slow Mind

Slow Mind is empowered by a proficient LLM. The main functionality of Slow Mind is to interpret vague human commands into concrete intentions, keep track of command completion progress, and provide responses containing key information and useful suggestions to the human partner. Also, Slow Mind sends the inferred human intention to Fast Mind and indicates to Fast Mind whether the command is successfully completed.

To this end, we devise Slow Mind to have two stages. The first stage, Intention Reasoning Stage, infers human intention given the command and command history when the human issues a new command. The second stage, Chat & Assessment Stage, periodically checks command completion and generates reply messages to the human partner based on the inferred intention. We use GPT-3.5 [40] here as GPT-3.5 features strong reasoning and communication power.

*4.2.1* Intention Reasoning Stage. To interpret human commands, the LLM is provided with the command history and environmental information. The LLM interprets the vague command into concrete intentions that best reflect needs of the human partner. For instance, when the current soup orders are "Alice Soup" and "David Soup", the command "Cook the first soup on the orders" should be interpreted as "Cook Alice Soup". We note that this intention reasoning stage is carried out only once when a command is issued. The inferred intention is then stored by Slow Mind and sent to Fast Mind. The prompt used in Intention Reasoning Stage is shown in Fig. 6.

**Input:**
You are in a virtual environment where ...
Current orders: Alice Soup with plenty of time, ...
Human's message: "Cook the first soup"
Human's last intention: ...

**Output:**
Human's intention now: Cook Alice Soup

**Figure 6: Slow Mind prompt in Intention Reasoning Stage.**

*4.2.2* Chat & Assessment Stage. In Chat & Assessment Stage, the LLM generates chat messages to the human partner and performs completion assessment. The chat messages cover different aspects of information including consent to the commands, planning of the AI agent, and information about the environment. The completion assessment is used as a tool to keep track of command completion

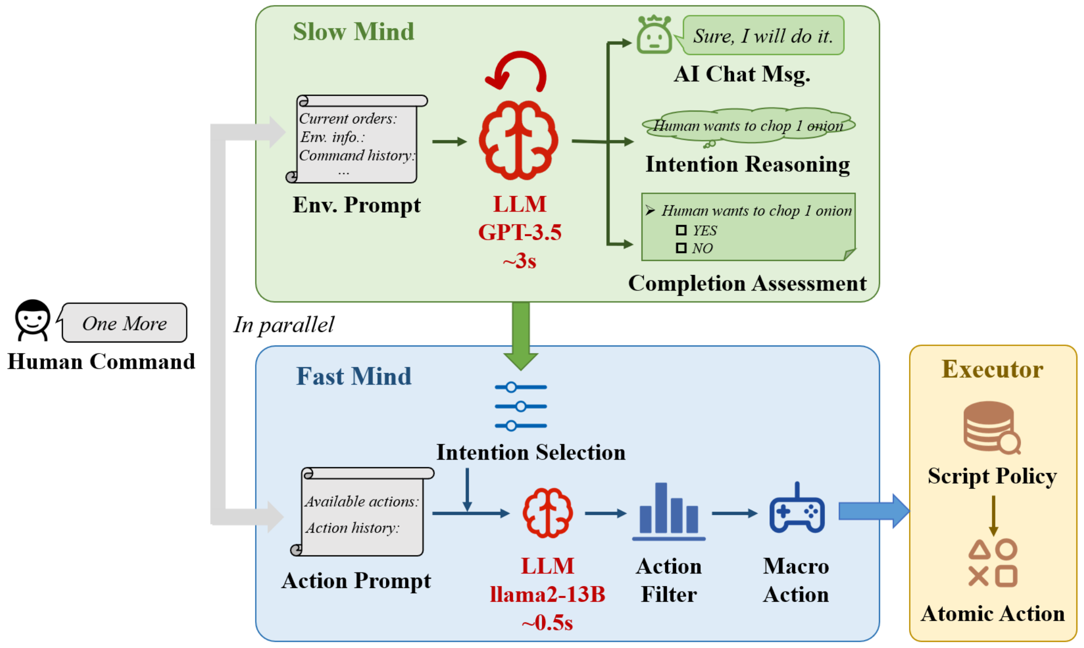


**Figure 3: Framework of Hierarchical Language Agent, including a Slow Mind for intention reasoning and language interaction, a Fast Mind for macro actions generation, and an Executor to execute atomic actions.**

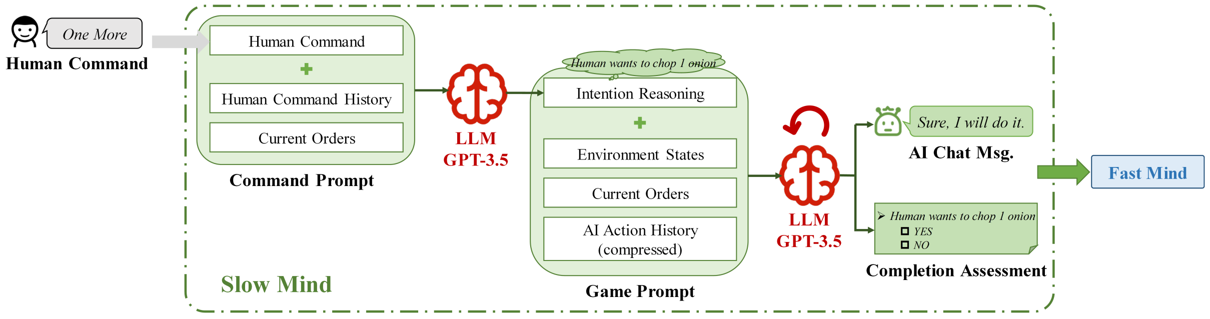


**Figure 4: Workflow of Slow Mind. Slow Mind employs a two-stage design. It reasons human intention according to human commands in the first stage, then generates chat message and performs completion assessment periodically in the second stage.**

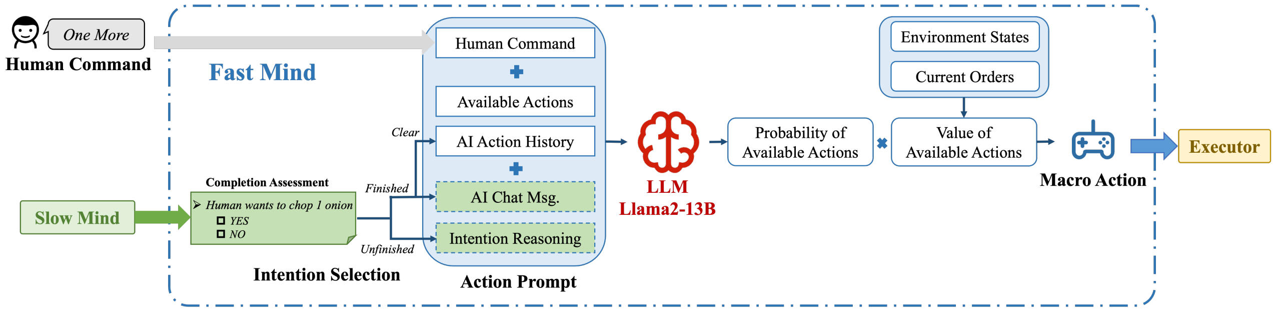


**Figure 5: Workflow of Fast Mind. Fast Mind is empowered by a lightweight LLM. It works with Slow Mind cooperatively with a conditional prompt mechanism and avoids sub-optimal moves with an action-filtering mechanism.**

progress. Once the human command is judged as completed, the inferred intention will be cleared and Fast Mind will be reminded of completion of the command.

The LLM takes as input the inferred intention, current orders, environment state, and action history. We note that we use a compact representation for the action history by expressing consecutive repeated actions in the form of *"action × repetition times"* to reduce the length of the action history.

Different from Intention Reasoning Stage that is only executed once when a new command arrives, Chat & Assessment Stage runs periodically to actively monitor the command completion progress and provide positive messages to the human partner. Notably, Chat & Assessment Stage runs in different modes according to whether there exists an unaccomplished human command. When a human command is not completed yet, Chat & Assessment Stage runs at full functionality. When there are no ongoing human commands, Slow Mind switches to a more casual mode that only generates chat messages and does not perform completion assessment.

Outline of the prompt used in Chat & Assessment Stage is shown in Fig. 7. We note that, when there are no ongoing human commands, completion assessment is disabled and the human's intention is ommited from the input prompt.

**Input:**
You are in a virtual environment where ...
Current orders: Alice Soup (plenty of time), ...
Environment state: ...
Human's intention is: Cook Alice Soup once
Action you've done: Chop Lettuce twice, ...

**Output:**
Reasoning: I haven't cook Alice Soup yet.
My chat message is: Sure, I will cook it for you.
Intention satisfied? No

**Figure 7: Slow Mind prompt in Chat & Assessment Stage.**

## 4.3 Fast Mind

As argued in Sec. 4, Fast Mind produces high-level moves at a medium frequency while following human commands. We call these high-level moves macro actions. The current macro action set in our testbed includes chopping vegetables, serving soup, putting out the fire, and so on. A detailed description of macro actions can be found in Sec. 4.4.

Despite having superior decision-making capabilities, a powerful LLM as used in Slow Mind is not applicable in Fast Mind due to huge inference latency. Therefore a lightweight LLM is more suitable. On the other hand, while a lightweight LLM such as LLaMA2-13B can successfully follow a human command when the command is concise and clear, it often generates sub-optimal moves that lead to a lower score when there are no ongoing commands and violates the command when the human partner gives a vague command.

Taking these factors into account, we design Fast Mind that generates macro actions to interact with the environment, as depicted in Fig. 5. Fast Mind is empowered by a lightweight LLM (we use the LLaMA2-13B[52] model here). To better ground human commands into moves, Fast Mind works with Slow Mind cooperatively with a conditional prompt mechanism. Lastly, Fast Mind avoids sub-optimal moves with an action-filtering mechanism.

### 4.3.1 *Intention Selection.*

To better align macro action generation with vague human commands, Fast Mind also uses the inferred intention from Slow Mind as input. To prevent macro action generation from being blocked by the intention reasoning stage of Slow Mind, Fast Mind uses the raw human command as input before Slow Mind successfully infers the intention. This asynchronous execution nature results in the conditional prompt mechanism used by Fast Mind as depicted in Fig. 8. When Slow Mind is analyzing the command, Fast Mind uses as input the raw command, action history, and availability of actions. As soon as Slow Mind completes Intention Reasoning Stage, the inferred intention is sent to Fast Mind, and Fast Mind switches to use the inferred intention instead of the raw command to generate macro actions. Later when the command is evaluated as completed by Slow Mind, the conditional input for Fast Mind is switched to the chat messages produced by Slow Mind, which helps the agent to better align its action with conversational response.

**Input:**
You are in a virtual environment where ...
[If intention is unclear] Human sends a message: ...
[If intention is reasoned] Human's intention is: ...
[If intention is satisfied] Your chat message is: ...

**Output:**
My actions are: Chop Lettuce, <next action>

**Figure 8: Fast Mind prompt.**

### 4.3.2 *Macro Action Generation.*

We obtain action probability by the output probability of each macro action conditioned on the prompt prefix. Fig. 8 illustrates the structure of the employed prompt. Each potential macro action candidate fills in the <next action> variable to evaluate its output probability. The action history contained in the output prompt is to remind the agent of its past actions.

Lightweight LLM used in Fast Mind may result in sub-optimal actions when managing complicated tasks. We therefore employ an action filter to filter out sub-optimal macro actions. The selection probabilities are calculated based on the probabilities output by the language model and the task-relevant value of each available macro action according to Eq. (1).

$$\log U(a|s) \propto \log P_{LLM}(a|s) + \alpha V(a|s) \quad (1)$$

Here, $U(a|s)$ represents the selection probability of macro action $a$ under current state $s$, $P_{LLM}(a|s)$ represents the probability of the language model outputting macro action $a$, $V(a|s)$ represents the value of macro action $a$ contributing to the task reward. The values for $V(a|s)$ are hardcoded, and their detailed information can be found in the Appendix B.2. $\alpha$ is a dynamic adjustment term and is smaller when the human command is not assessed to be met and is larger when it is. Finally, we adopt a greedy selection scheme, i.e. macro action with the greatest $U(a|s)$ is selected.

## 4.4 Executor

At the lowest level of HLA, Executor employs a script policy to convert macro actions generated by Fast Mind into atomic actions to interact with the environment. The established macro action set includes the following types.

- *Chop:* Transform an ingredient into its chopped form.
- *Mix:* Combine chopped ingredients for cooking.
- *Cook:* Utilize mixed ingredients to cook a soup.
- *Plate:* Transfer a ready soup to a plate.
- *Serve:* Deliver a plated soup to the delivery point.
- *Putout:* Extinguish a fire on a pot using an extinguisher.
- *Drop:* Plate charred soup and discard it into a bin.

Most macro actions are equipped with a specified target, such as *"Chop Lettuce"* and *"Cook Bob Soup"*, consequently resulting in a total set of 21 macro actions. Implementation details of the reactive policy can be found in Appendix B.2. Given a macro action as a high-level goal, Executor chooses the most appropriate move and performs path planning to the target. We empirically found this to be particularly beneficial as shown in Sec. 5.3.

Note that the reactive policy of Executor shares similarities with goal-conditioned reinforcement learning [14, 35], which can, in turn, be trained by typical reinforcement learning algorithms [49]. We remark this as a topic for future studies.

## 5 EXPERIMENT

In this section, we consider three baseline agents to verify the hierarchical structure of HLA. We first test the real-time responsiveness of HLA and baselines by measuring action response latency. Moreover, we use a simple command set to evaluate the cooperative ability of each agent and a more complex command set to test command reasoning ability. Finally, we conduct human studies to collect the game scores and the human preference of all agents.

### 5.1 Baseline

To validate the effectiveness of the proposed hierarchical framework, we introduce three baseline agents, each lacking a certain component of the original HLA.

- *Slow-Mind-Only Agent (SMOA).* We remove the Fast Mind and let the Slow Mind produce macro actions. Current available macro actions are added to the input of the Slow Mind. The action filter is disregarded as GPT-3.5 is incapable of evaluating the probability for each macro action.
- *Fast-Mind-Only Agent (FMOA).* We remove the Slow Mind, including both Intention Reasoning Stage and Chat & Assessment Stage. Due to the absence of Chat & Assessment Stage, the dynamic adjustment term $\alpha$ in held static in Eq. (1). We set a maximum length for action history, beyond which the human intention is assumed as fulfilled. History of human commands, current orders, and environment states are incorporated additionally into the input of Fast Mind.
- *No-Executor Agent (NEA).* We remove the Executor and let the Fast Mind choose atomic actions to control the agent directly. In addition to the original input, the Fast Mind of the NEA incorporates environmental state information, such as the positions of all items on the map.

Implementation details of the baseline agents can be found in Appendix C.

### 5.2 Latency

We use *macro action latency* and *atomic action latency* to measure the real-time responsiveness of HLA and baseline agents. The macro action latency is defined as the time interval between receiving a human command and subsequently generating a macro action. And the atomic action latency is, in turn, the latency of an atomic action. In order to simulate the natural human-AI coordination, we build a command set and issue each command to the AI agent to evaluate its response latency. Details about latency measurement method and the command set can be found in Appendix D.1.

The results of macro action latency and atomic action latency are depicted in Tab. 1, where lower number implies a faster action response time. The macro action latency of NEA is marked as "/" since it generates atomic actions directly. In HLA, when a human command is received, it is concurrently dispatched to both the Slow Mind and the Fast Mind therefore we report the macro action generation time of the Fast Mind.

Regarding the macro action latency, HLA produces 74.3% lower latency than SMOA and 53.5% lower latency than FMOA, suggesting that the hierarchical design significantly contributes to reducing response latency. FMOA exhibits a marginally shorter response time than SMOA, which can be attributed to the shorter inference time of the lightweight LLM, as well as the elimination of the intention completion assessment module in Slow Mind. Regarding the atomic action latency, HLA exhibits an order of magnitude advantage over the best competitor (0.28 vs. 0.08), demonstrating the real-time responsiveness of HLA. Among all agents, NEA performs the highest atomic action latency, indicating the importance of Executor that interacts with the environment with high frequency.

| | NEA | SMOA | FMOA | HLA |
|---|---|---|---|---|
| Mac. Act. Latency(s) | / | 4.16 (1.01) | 2.30 (1.81) | **1.07 (0.22)** |
| Ato. Act. Latency(s) | 0.71 (0.08) | 0.61 (0.65) | 0.28 (0.31) | **0.08 (0.06)** |

**Table 1: Macro action latency and atomic action latency of different agents. The format is "mean (standard deviation)".**

### 5.3 Performances with Simple Commands

An ideal AI agent should cooperate seamlessly with human players to achieve high game score, either in the absence of explicit human commands or after human commands is fulfilled. Thus, we design two test cases where the human player either remains silent or speaks very little. In both cases, the human player only chops ingredients and does not perform other tasks.

- *No Command*: The human player does not issue any command throughout the game.
- *One Command*: The human player asks the AI agent to prepare a specific soup at the start of the game, and the order for this soup only appears once at the start of the game.

We employ game score as a metric to assess the cooperative ability of HLA and baseline agents. The game score ascends every time an order is finished in time and descends whenever an order expires. We test both *No Command* and *One Command* on Map *Quick* and replicate the experiment five times with the same order sequence and keep the same experiment conditions. The average game scores under different test cases are depicted in Fig. 9, where the black line denotes standard deviation. Visualization of the gaming process can be found in Appendix E.2.

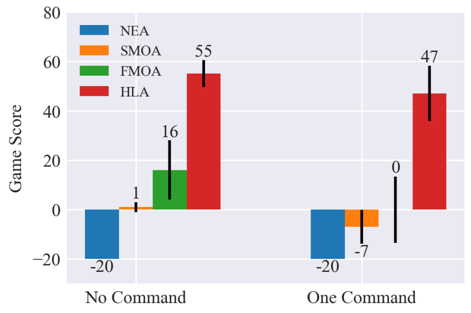


**Figure 9: Average game scores of HLA and baseline agents. Black line denotes standard deviation.**

NEA fails to complete basic tasks, such as picking up things or approaching the target, and frequently gets stuck, therefore misses all soup orders and achieves minimum score of −20. This underscores the vital role played by the Executor, which translates macro actions into atomic actions. SMOA adequately follows the human command in the *One Command* case, but often overcooks dishes due to its high response latency, thus also performs poorly. In the *No Command* case, FMOA serves the desired soups successfully but underperforms compared to HLA, primarily due to a high latency. However, in the *One Command* case, FMOA keeps making the requested soup even if it does not appear on the order list. In other words, FMOA fails to confirm the completion of a human command leads to suboptimal actions that do not fulfill the orders.

## 5.4 Interpreting Complex Commands

To see how well the agents comprehend and respond to commands of varying complexity, we design a complex command set comprising the following three challenges outlined in Sec. 3.2.

- *Quantity specification (Quantity).* We consider commands with specific numbers, e.g. "Chop 3 Lettuce." or "Cook Bob Soup once."
- *Semantic analysis (Semantics).* For example, the human gives a hint, "Alice Soup is about to timeout!" instead of a direct command like "Cook Alice Soup."
- *Ambiguous reference (Ambiguity).* For example, after asking the agent to chop two onion, the human player asks, "Chop 1 more." The AI agents need to infer the true intention based on environmental context and history commands.

We generate 10 different commands for each challenge. The details of the complex command set can be found in Appendix D.2. Each individual command is tested for 5 times. A command is considered successful if it succeeds at least 3 out of 5 attempts within 60s. In such cases, the command is labeled as passed, and its completion time is calculated as the average time taken for successful attempts. Otherwise, it is marked as failed and its completion time is marked as the maximum time of 60s. We report the average success rate and average completion time of commands in each challenge subset.

Tab. 2 shows the results of HLA and baseline agents. NEA fails to execute any complex command and thus produces the worst performance. Except NEA, FMOA exhibits the lowest success rate and the highest completion time when handling ambiguous commands, highlighting the significance of intention reasoning through the Slow Mind. SMOA performs poorly on quantity and semantics commands, displaying the longest completion time, indicating that although the Slow Mind can interpret commands, it suffers from prolonged latency. HLA surpasses baseline agents with a notable advantage in both success rate and completion time, which demonstrates the effectiveness of the hierarchical design.

| AI Agents | Quantity | | Semantics | | Ambiguity | |
|---|---|---|---|---|---|---|
| | Suc.↑ | Time↓ | Suc.↑ | Time↓ | Suc.↑ | Time↓ |
| NEA | 0.00 | 60.00 | 0.00 | 60.00 | 0.00 | 60.00 |
| SMOA | 0.40 | 47.14 | 0.60 | 40.20 | **0.70** | 39.57 |
| FMOA | 0.60 | 40.01 | 0.60 | 32.67 | 0.30 | 50.16 |
| HLA | **1.00** | **13.30** | **0.90** | **17.13** | **0.70** | **27.78** |

**Table 2: Success rate and completion time for complex commands of HLA and baseline agents.**

**Ablation study on the two-stage design of Slow Mind.** The AI agent needs to infer human intentions and evaluate whether human commands are completed. In HLA, these tasks are performed separately by the two stages of Slow Mind. In this section, we consider two variants of Slow Mind to validate our two-stage design on interpreting complex commands.

- *HLA without Intention Reasoning (no IR).* We remove the Intention Reasoning Stage and use human commands directly as intentions for Chat & Assessment Stage.
- *HLA with one-stage Slow Mind (one-stage).* We combine the Intention Reasoning Stage and the Chat & Assessment Stage within Slow Mind, which now infers human intention, chats, and assesses intention completion simultaneously.

The results are shown in Tab. 3. HLA with one-stage Slow Mind exhibits the longest completion time and the lowest success rate. We hypothesize that this is due to Slow Mind having to handle multiple tasks simultaneously, resulting in poorer performance and longer inference time. HLA without Intention Reasoning performs worse than HLA(full) across all challenge subsets, particularly on ambiguous commands, indicating the significance of incorporating the Intention Reasoning Stage.

| HLA variants | Quantity | | Semantics | | Ambiguity | |
|---|---|---|---|---|---|---|
| | Suc.↑ | Time↓ | Suc.↑ | Time↓ | Suc.↑ | Time↓ |
| no IR | 0.80 | 22.42 | 0.80 | 22.14 | 0.60 | 40.50 |
| one-stage | 0.50 | 35.13 | 0.50 | 36.90 | 0.50 | 44.36 |
| HLA(full) | **1.00** | **13.30** | **0.90** | **17.13** | **0.70** | **27.78** |

**Table 3: Success rate and completion time for complex commands of HLA and its two variants.**

## 5.5 Human Studies

*5.5.1 Experiment Setting.* We invite 60 volunteers for the human-AI experiment and divide them into 4 groups, each of which consists of 15 volunteers playing on a specific map. All volunteers are provided with a detailed introduction to the basic gameplay and the experiment process. They are fully aware of all their rights and experiments are approved with the permission of the department.

Considering the poor performance of NEA observed in Sec. 5.3, we only evaluate SMOA, FMOA, and HLA in this section. Each volunteer plays with the three AI players on the same map for two gaming phases, the preparation phase and the competition phase. During the preparation phase, volunteers are required to collaborate with each of the three AI players for at least one round. In this process, human players familiarize themselves with the environment and engage in casual interactions with the AI players to explore their behaviors. In the competition phase, volunteers can only play one round of the game with each of the three AI players to achieve the highest possible score. We record the game scores and ask the volunteers to rank the three AI players based on their gaming experience. We report the game scores and human preferences of SMOA, FMOA, and HLA in the following subsections.

#### 5.5.2 *Game Score.*

The average game scores on various maps from the competition phase are presented in Tab. 4. A high variance in scores can be attributed to the discrete nature of task rewards, where finishing an order yields 15 to 20 points. SMOA exhibits the lowest scores on all maps due to its high response latency. Compared to SMOA, FMOA performs slightly better, averaging a 10-point advantage in scoring on each map. HLA outperforms both SMOA and FMOA across all maps with $\sim$ 50% higher game scores, reflecting a significant advantage. In particular, HLA achieves a game score increase of over 40 points, i.e. serving at least two additional orders, on *Partition* and *Quick* when compared to baseline agents. This highlights HLA's effective collaboration ability and real-time responsiveness.

| AI Agents | *Ring* | *Partition* | *Bottleneck* | *Quick* |
|---|---|---|---|---|
| SMOA | 80.9 (29.1) | 33.0 (27.1) | 102.4 (35.6) | 60.8 (48.5) |
| FMOA | 92.5 (21.7) | 57.7 (37.9) | 103.8 (30.6) | 71.2 (50.2) |
| HLA | **114.4** **(19.4)** | **100.3** **(36.4)** | **130.3** **(19.7)** | **117.2** **(45.3)** |

**Table 4: Average game score when paired with different AI players. Standard deviations are shown in parentheses.**

**Behavior Analysis.** We additionally calculate the ratio of valuable macro actions performed by each AI player, as well as the frequency of fire accidents on all maps during the competition phase. A macro action is considered valuable if it produces a positive utility value upon execution. Conversely, useless macro actions may either be inherently invalid or become unexecutable due to high inference latency. We analyze three common macro actions, including *Chop* ingredients,*Cook* soups and *Serve* orders. The results are documented in Tab. 5. HLA stands out by executing the highest quantity of valuable macro actions and minimizing fire accidents, significantly outperforming SMOA and FMOA. Notably, the ratio of *Serve* macro action executed by HLA is 100%, indicating not only its capability to serve the correct order but also to do so promptly. This further helps to explain the excellent performance of HLA with regard to reasoning and real-time responsiveness.

| AI Players | *Chop* ↑ | *Cook* ↑ | *Serve* ↑ | *Fire* ↓ |
|---|---|---|---|---|
| SMOA | 0.569 | 0.755 | 0.273 | 0.118 |
| FMOA | 0.766 | 0.833 | 0.850 | 0.059 |
| HLA | **0.828** | **0.950** | **1.000** | **0.029** |

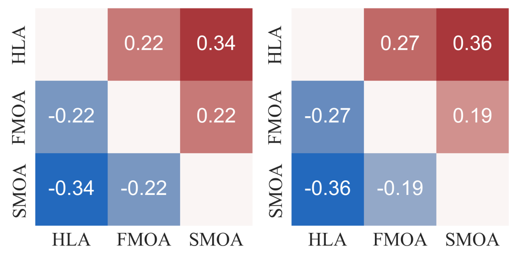
**Table 5: The ratio of valuable actions performed by different AI players and the frequency of fire accidents on all maps.**

#### 5.5.3 *Human Preference.*

In this section, we report the human preference on different AI players. Fig. 10 and Fig. 11 show the language communication preference and the overall preference of human participants respectively. Numbers indicate the difference of players who prefer row AI player over column AI player.

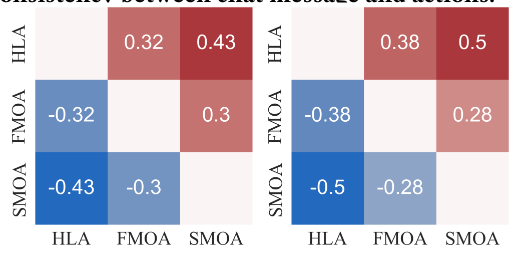
**Communication Preference.** Fig. 10 illustrates human feedback on communication accuracy, as well as consistency between chat messages and actions. The data is derived from both the preparation and competition phases. More than 20% of human players prefer HLA over SMOA and FMOA in terms of language communication accuracy as well as the consistency between chat messages and actions. We can also observe that FMOA outperforms SMOA with an advantage of ~20% human preference, primarily due to SMOA's high latency according to the collected feedback. This underscores the pivotal role of real-time responsiveness of effective language communication.

**Overall Preference.** Fig. 11 reports the overall human preference of the preparation phase and the competition phase. Human participants expressed a strong preference for HLA over SMOA and FMOA, with a preference rate exceeding 30%. This preference was particularly evident during the competition phase, where HLA was favored over SMOA by as much as 50%. Human preference for FMOA remains higher than that for SMOA, at around 30%, aligning with the conclusion of communication preferences. Overall preference is a comprehensive metric of how human players evaluate an AI player's actions and communication. The strong preference on HLA indicates that it exhibits superior cooperative skills, faster responsiveness, and more consistent language communication.



(a) Comm. Accuracy. (b) Comm. Consistency.

**Figure 10: Human preference on communication accuracy, and consistency between chat message and actions.**



(a) Preparation Phase (b) Competition Phase

**Figure 11: Overall human preference on different AI players.**

## 6 CONCLUSION

We propose Hierarchical Language Agent, an AI agent that can cooperate with humans using natural language in environments that require real-time execution. Throughout the comprehensive experiments in an extended Overcooked, our method consistently excels in terms of game score, response latency, and human preference, showcasing reliable real-time human-AI cooperation. Our method could be improved in a few key areas: substituting GPT-3.5 with GPT-4 in the Slow Mind for enhanced semantic analysis, and replacing the scripted executor with an automatic one developed through goal-conditioned reinforcement learning to streamline scripting and boost low-level execution performance.

## REFERENCES

[1] Saaket Agashe, Yue Fan, and Xin Eric Wang. 2023. Evaluating Multi-Agent Coordination Abilities in Large Language Models. *arXiv preprint arXiv:2310.03903* (2023).

[2] Mohammad Rafayet Ali, Seyedeh Zahra Razavi, Raina Langevin, Abdullah Al Mamun, Benjamin Kane, Reza Rawassizadeh, Lenhart K Schubert, and Ehsan Hoque. 2020. A virtual conversational agent for teens with autism spectrum disorder: Experimental results and design lessons. In *Proceedings of the 20th ACM International Conference on Intelligent Virtual Agents*. 1–8.

[3] Andrea Bajcsy, Dylan P Losey, Marcia K O'malley, and Anca D Dragan. 2017. Learning robot objectives from physical human interaction. In *Conference on Robot Learning*. PMLR, 217–226.

[4] Rishi Bommasani, Percy Liang, and Tony Lee. 2023. Holistic Evaluation of Language Models. *Annals of the New York Academy of Sciences* (2023).

[5] brian ichter, Anthony Brohan, Yevgen Chebotar, Chelsea Finn, Karol Hausman, Alexander Herzog, Daniel Ho, Julian Ibarz, Alex Irpan, Eric Jang, Ryan Julian, Dmitry Kalashnikov, Sergey Levine, Yao Lu, Carolina Parada, Kanishka Rao, Pierre Sermanet, Alexander T Toshev, Vincent Vanhoucke, Fei Xia, Ted Xiao, Peng Xu, Mengyuan Yan, Noah Brown, Michael Ahn, Omar Cortes, Nicolas Sievers, Clayton Tan, Sichun Xu, Diego Reyes, Jarek Rettinghouse, Jornell Quiambao, Peter Pastor, Linda Luu, Kuang-Huei Lee, Yuheng Kuang, Sally Jesmonth, Kyle Jeffrey, Rosario Jauregui Ruano, Jasmine Hsu, Keerthana Gopalakrishnan, Byron David, Andy Zeng, and Chuyuan Kelly Fu. 2022. Do As I Can, Not As I Say: Grounding Language in Robotic Affordances. In *6th Annual Conference on Robot Learning*. https://openreview.net/forum?id=bdHkMjBJG_w

[6] Tim Brooks, Aleksander Holynski, and Alexei A Efros. 2023. Instructpix2pix: Learning to follow image editing instructions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 18392–18402.

[7] Can Chang, Ni Mu, Jiajun Wu, Ling Pan, and Huazhe Xu. 2022. E-MAPP: Efficient Multi-Agent Reinforcement Learning with Parallel Program Guidance. In *Advances in Neural Information Processing Systems*, S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh (Eds.), Vol. 35. Curran Associates, Inc., 12154–12168. https://proceedings.neurips.cc/paper_files/paper/2022/file/4f2accafe6fa355624f5ee42207cc7b8-Paper-Conference.pdf

[8] Guangyao Chen, Siwei Dong, Yu Shu, Ge Zhang, Jaward Sesay, Börje F Karlsson, Jie Fu, and Yemin Shi. 2023. AutoAgents: A Framework for Automatic Agent Generation. *arXiv preprint arXiv:2309.17288* (2023).

[9] Weize Chen, Yusheng Su, Jingwei Zuo, Cheng Yang, Chenfei Yuan, Chen Qian, Chi-Min Chan, Yujia Qin, Yaxi Lu, Ruobing Xie, et al. 2023. Agentverse: Facilitating multi-agent collaboration and exploring emergent behaviors in agents. *arXiv preprint arXiv:2308.10848* (2023).

[10] Brandon Cui, Hengyuan Hu, Luis Pineda, and Jakob Foerster. 2021. K-level reasoning for zero-shot coordination in hanabi. *Advances in Neural Information Processing Systems* 34 (2021), 8215–8228.

[11] Xiang Deng, Yu Gu, Boyuan Zheng, Shijie Chen, Samuel Stevens, Boshi Wang, Huan Sun, and Yu Su. 2023. Mind2Web: Towards a Generalist Agent for the Web. *arXiv preprint arXiv:2306.06070* (2023).

[12] Qingxiu Dong, Lei Li, Damai Dai, Ce Zheng, Zhiyong Wu, Baobao Chang, Xu Sun, Jingjing Xu, and Zhifang Sui. 2022. A survey for in-context learning. *arXiv preprint arXiv:2301.00234* (2022).

[13] Danny Driess, Fei Xia, Mehdi SM Sajjadi, Corey Lynch, Aakanksha Chowdhery, Brian Ichter, Ayzaan Wahid, Jonathan Tompson, Quan Vuong, Tianhe Yu, et al. 2023. Palm-e: An embodied multimodal language model. *arXiv preprint arXiv:2303.03378* (2023).

[14] Benjamin Eysenbach, Tianjun Zhang, Sergey Levine, and Russ R Salakhutdinov. 2022. Contrastive learning as goal-conditioned reinforcement learning. *Advances in Neural Information Processing Systems* 35 (2022), 35603–35620.

[15] Meta Fundamental AI Research Diplomacy Team (FAIR)†, Anton Bakhtin, Noam Brown, Emily Dinan, Gabriele Farina, Colin Flaherty, Daniel Fried, Andrew Goff, Jonathan Gray, Hengyuan Hu, et al. 2022. Human-level play in the game of Diplomacy by combining language models with strategic reasoning. *Science* 378, 6624 (2022), 1067–1074.

[16] Jun Fang, Ali Shafiee, Hamzah Abdel-Aziz, David Thorsley, Georgios Georgiadis, and Joseph H Hassoun. 2020. Post-training piecewise linear quantization for deep neural networks. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part II 16*. Springer, 69–86.

[17] Jaime F Fisac, Andrea Bajcsy, Sylvia L Herbert, David Fridovich-Keil, Steven Wang, Claire J Tomlin, and Anca D Dragan. 2018. Probabilistically safe robot planning with confidence-based human predictions. *arXiv preprint arXiv:1806.00109* (2018).

[18] Chen Gao, Xiaochong Lan, Zhihong Lu, Jinzhu Mao, Jinghua Piao, Huandong Wang, Depeng Jin, and Yong Li. 2023. $S^3$: Social-network Simulation System with Large Language Model-Empowered Agents. *arXiv preprint arXiv:2307.14984* (2023).

[19] Difei Gao, Lei Ji, Luowei Zhou, Kevin Qinghong Lin, Joya Chen, Zihan Fan, and Mike Zheng Shou. 2023. AssistGPT: A General Multi-modal Assistant that can Plan, Execute, Inspect, and Learn. *arXiv preprint arXiv:2306.08640* (2023).

[20] Richard Goodwin. 1995. Formalizing properties of agents. *Journal of Logic and Computation* 5, 6 (1995), 763–781.

[21] Izzeddin Gur, Hiroki Furuta, Austin Huang, Mustafa Safdari, Yutaka Matsuo, Douglas Eck, and Aleksandra Faust. 2023. A real-world webagent with planning, long context understanding, and program synthesis. *arXiv preprint arXiv:2307.12856* (2023).

[22] Karl Moritz Hermann, Felix Hill, Simon Green, Fumin Wang, Ryan Faulkner, Hubert Soyer, David Szepesvari, Wojciech Marian Czarnecki, Max Jaderberg, Denis Teplyashin, et al. 2017. Grounded language learning in a simulated 3d world. *arXiv preprint arXiv:1706.06551* (2017).

[23] Hengyuan Hu, Adam Lerer, Alex Peysakhovich, and Jakob Foerster. 2020. "other-play" for zero-shot coordination. In *International Conference on Machine Learning*. PMLR, 4399–4410.

[24] Wenlong Huang, Pieter Abbeel, Deepak Pathak, and Igor Mordatch. 2022. Language models as zero-shot planners: Extracting actionable knowledge for embodied agents. In *International Conference on Machine Learning*. PMLR, 9118–9147.

[25] Wenlong Huang, Chen Wang, Ruohan Zhang, Yunzhu Li, Jiajun Wu, and Li Fei-Fei. 2023. VoxPoser: Composable 3D Value Maps for Robotic Manipulation with Language Models. In *7th Annual Conference on Robot Learning*. https://openreview.net/forum?id=9_8LF30mOC

[26] Wenlong Huang, Fei Xia, Ted Xiao, Harris Chan, Jacky Liang, Pete Florence, Andy Zeng, Jonathan Tompson, Igor Mordatch, Yevgen Chebotar, et al. 2023. Inner Monologue: Embodied Reasoning through Planning with Language Models. In *Conference on Robot Learning*. PMLR, 1769–1782.

[27] Yunfan Jiang, Agrim Gupta, Zichen Zhang, Guanzhi Wang, Yongqiang Dou, Yanjun Chen, Li Fei-Fei, Anima Anandkumar, Yuke Zhu, and Linxi Fan. 2023. VIMA: Robot Manipulation with Multimodal Prompts. In *International Conference on Machine Learning*.

[28] Daniel Kahneman. 2011. *Thinking, fast and slow*. macmillan.

[29] Guohao Li, Hasan Abed Al Kader Hammoud, Hani Itani, Dmitrii Khizbullin, and Bernard Ghanem. 2023. Camel: Communicative agents for" mind" exploration of large scale language model society. *arXiv preprint arXiv:2303.17760* (2023).

[30] Raymond Li, Loubna Ben Allal, Yangtian Zi, Niklas Muennighoff, Denis Kocetkov, Chenghao Mou, Marc Marone, Christopher Akiki, Jia Li, Jenny Chim, et al. 2023. StarCoder: may the source be with you! *arXiv preprint arXiv:2305.06161* (2023).

[31] Jacky Liang, Wenlong Huang, Fei Xia, Peng Xu, Karol Hausman, Brian Ichter, Pete Florence, and Andy Zeng. 2023. Code as Policies: Language Model Programs for Embodied Control. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*. 9493–9500. https://doi.org/10.1109/ICRA48891.2023.10160591

[32] Zhenhua Liu, Yunhe Wang, Kai Han, Wei Zhang, Siwei Ma, and Wen Gao. 2021. Post-training quantization for vision transformer. *Advances in Neural Information Processing Systems* 34 (2021), 28092–28103.

[33] Bo-Ru Lu, Nikita Haduong, Chia-Hsuan Lee, Zeqiu Wu, Hao Cheng, Paul Koester, Jean Utke, Tao Yu, Noah A Smith, and Mari Ostendorf. 2023. DIALGEN: Collaborative Human-LM Generated Dialogues for Improved Understanding of Human-Human Conversations. *arXiv preprint arXiv:2307.07047* (2023).

[34] Ziyang Luo, Can Xu, Pu Zhao, Qingfeng Sun, Xiubo Geng, Wenxiang Hu, Chongyang Tao, Jing Ma, Qingwei Lin, and Daxin Jiang. 2023. WizardCoder: Empowering Code Large Language Models with Evol-Instruct. *arXiv preprint arXiv:2306.08568* (2023).

[35] Jason Yecheng Ma, Jason Yan, Dinesh Jayaraman, and Osbert Bastani. 2022. Offline goal-conditioned reinforcement learning via $f$-advantage regression. *Advances in Neural Information Processing Systems* 35 (2022), 310–323.

[36] Markus Nagel, Rana Ali Amjad, Mart Van Baalen, Christos Louizos, and Tijmen Blankevoort. 2020. Up or down? adaptive rounding for post-training quantization. In *International Conference on Machine Learning*. PMLR, 7197–7206.

[37] Reiichiro Nakano, Jacob Hilton, Suchir Balaji, Jeff Wu, Long Ouyang, Christina Kim, Christopher Hesse, Shantanu Jain, Vineet Kosaraju, William Saunders, et al. 2021. Webgpt: Browser-assisted question-answering with human feedback. *arXiv preprint arXiv:2112.09332* (2021).

[38] Reiichiro Nakano, Jacob Hilton, Suchir Balaji, Jeff Wu, Long Ouyang, Christina Kim, Christopher Hesse, Shantanu Jain, Vineet Kosaraju, William Saunders, et al. 2021. Webgpt: Browser-assisted question-answering with human feedback. *arXiv preprint arXiv:2112.09332* (2021).

[39] Xuefei Ning, Zinan Lin, Zixuan Zhou, Huazhong Yang, and Yu Wang. 2023. Skeleton-of-thought: Large language models can do parallel decoding. *arXiv preprint arXiv:2307.15337* (2023).

[40] OpenAI. 2022. Introducing ChatGPT. https://openai.com/blog/chatgpt.

[41] R OpenAI. 2023. GPT-4 technical report. *arXiv* (2023), 2303–08774.

[42] Joon Sung Park, Joseph O'Brien, Carrie Jun Cai, Meredith Ringel Morris, Percy Liang, and Michael S Bernstein. 2023. Generative agents: Interactive simulacra of human behavior. In *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology*. 1–22.

[43] Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, et al. 2023. Toolllm: Facilitating large language models to master 16000+ real-world apis. *arXiv preprint arXiv:2307.16789* (2023).

[44] Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. 2022. Hierarchical text-conditional image generation with clip latents. *arXiv preprint arXiv:2204.06125* 1, 2 (2022), 3.

[45] Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. 2021. Zero-shot text-to-image generation. In *International Conference on Machine Learning*. PMLR, 8821–8831.

[46] Baptiste Roziere, Jonas Gehring, Fabian Gloeckle, Sten Sootla, Itai Gat, Xiaoqing Ellen Tan, Yossi Adi, Jingyu Liu, Tal Remez, Jérémy Rapin, et al. 2023. Code llama: Open foundation models for code. *arXiv preprint arXiv:2308.12950* (2023).

[47] Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. 2023. Toolformer: Language models can teach themselves to use tools. *arXiv preprint arXiv:2302.04761* (2023).

[48] Timo Schick, Jane A. Yu, Zhengbao Jiang, Fabio Petroni, Patrick Lewis, Gautier Izacard, Qingfei You, Christoforos Nalmpantis, Edouard Grave, and Sebastian Riedel. 2023. PEER: A Collaborative Language Model. In *The Eleventh International Conference on Learning Representations*. https://openreview.net/forum?id=KbYevcLjnc

[49] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347* (2017).

[50] Bo Shen, Jiaxin Zhang, Taihong Chen, Daoguang Zan, Bing Geng, An Fu, Muhan Zeng, Ailun Yu, Jichuan Ji, Jingyang Zhao, et al. 2023. Pangu-coder2: Boosting large language models for code with ranking feedback. *arXiv preprint arXiv:2307.14936* (2023).

[51] DJ Strouse, Kevin McKee, Matt Botvinick, Edward Hughes, and Richard Everett. 2021. Collaborating with humans without human data. *Advances in Neural Information Processing Systems* 34 (2021), 14502–14515.

[52] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288* (2023).

[53] Sai Vemprala, Rogerio Bonatti, Arthur Bucker, and Ashish Kapoor. 2023. Chatgpt for robotics: Design principles and model abilities. *Microsoft Auton. Syst. Robot. Res* 2 (2023), 20.

[54] Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. 2023. Voyager: An open-ended embodied agent with large language models. *arXiv preprint arXiv:2305.16291* (2023).

[55] Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2023. Self-Consistency Improves Chain of Thought Reasoning in Language Models. In *The Eleventh International Conference on Learning Representations*. https://openreview.net/forum?id=1PL1NIMMrw

[56] Xinyi Wang, Wanrong Zhu, and William Yang Wang. 2023. Large language models are implicitly topic models: Explaining and finding good demonstrations for in-context learning. *arXiv preprint arXiv:2301.11916* (2023).

[57] Zihao Wang, Shaofei Cai, Guanzhou Chen, Anji Liu, Xiaojian Ma, and Yitao Liang. 2023. Describe, Explain, Plan and Select: Interactive Planning with LLMs Enables Open-World Multi-Task Agents. In *Thirty-seventh Conference on Neural Information Processing Systems*. https://openreview.net/forum?id=KtvPdGb31Z

[58] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, brian ichter, Fei Xia, Ed H. Chi, Quoc V Le, and Denny Zhou. 2022. Chain of Thought Prompting Elicits Reasoning in Large Language Models. In *Advances in Neural Information Processing Systems*, Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho (Eds.). https://openreview.net/forum?id=_VjQlMeSB_J

[59] Michael Wooldridge and Nicholas R Jennings. 1995. Intelligent agents: Theory and practice. *The knowledge engineering review* 10, 2 (1995), 115–152.

[60] Sarah A Wu, Rose E Wang, James A Evans, Joshua B Tenenbaum, David C Parkes, and Max Kleiman-Weiner. 2021. Too Many Cooks: Bayesian Inference for Coordinating Multi-Agent Collaboration. *Topics in Cognitive Science* 13, 2 (2021), 414–432.

[61] yi Wu, Yuxin Wu, Gkioxari Georgia, and Yuandong Tian. 2018. Building Generalizable Agents with a Realistic and Rich 3D Environment. (2018). https://openreview.net/forum?id=rkaT3zWCZ

[62] Shusheng Xu, Huaijie Wang, and Yi Wu. 2022. Grounded reinforcement learning: Learning to win the game under human commands. *Advances in Neural Information Processing Systems* 35 (2022), 7504–7519.

[63] Xue Yan, Yan Song, Xinyu Cui, Filippos Christianos, Haifeng Zhang, David Henry Mguni, and Jun Wang. 2023. Ask more, know better: Reinforce-Learned Prompt Questions for Decision Making with Large Language Models. *arXiv preprint arXiv:2310.18127* (2023).

[64] Songhua Yang, Hanjia Zhao, Senbin Zhu, Guangyu Zhou, Hongfei Xu, Yuxiang Jia, and Hongying Zan. 2023. Zhongjing: Enhancing the Chinese Medical Capabilities of Large Language Model through Expert Feedback and Real-world Multi-turn Dialogue. *arXiv preprint arXiv:2308.03549* (2023).

[65] Shunyu Yao, Howard Chen, John Yang, and Karthik Narasimhan. 2022. Webshop: Towards scalable real-world web interaction with grounded language agents. *Advances in Neural Information Processing Systems* 35 (2022), 20744–20757.

[66] Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L Griffiths, Yuan Cao, and Karthik Narasimhan. 2023. Tree of thoughts: Deliberate problem solving with large language models. *arXiv preprint arXiv:2305.10601* (2023).

[67] 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*. https://openreview.net/forum?id=TrwE8l9aJzs

[68] Ceyao Zhang, Kaijie Yang, Siyi Hu, Zihao Wang, Guanghe Li, Yihang Sun, Cheng Zhang, Zhaowei Zhang, Anji Liu, Song-Chun Zhu, et al. 2023. Proagent: Building proactive cooperative ai with large language models. *arXiv preprint arXiv:2308.11339* (2023).

[69] 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. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 37. 6145–6153.

[70] Wangchunshu Zhou, Yuchen Eleanor Jiang, Long Li, Jialong Wu, Tiannan Wang, Shi Qiu, Jintian Zhang, Jing Chen, Ruipu Wu, Shuai Wang, et al. 2023. Agents: An open-source framework for autonomous language agents. *arXiv preprint arXiv:2309.07870* (2023).

[71] Xizhou Zhu, Yuntao Chen, Hao Tian, Chenxin Tao, Weijie Su, Chenyu Yang, Gao Huang, Bin Li, Lewei Lu, Xiaogang Wang, et al. 2023. Ghost in the Minecraft: Generally Capable Agents for Open-World Enviroments via Large Language Models with Text-based Knowledge and Memory. *arXiv preprint arXiv:2305.17144* (2023).

[72] Brianna Zitkovich, Tianhe Yu, Sichun Xu, Peng Xu, Ted Xiao, Fei Xia, Jialin Wu, Paul Wohlhart, Stefan Welker, Ayzaan Wahid, Quan Vuong, Vincent Vanhoucke, Huong Tran, Radu Soricut, Anikait Singh, Jaspiar Singh, Pierre Sermanet, Pannag R Sanketi, Grecia Salazar, Michael S Ryoo, Krista Reymann, Kanishka Rao, Karl Pertsch, Igor Mordatch, Henryk Michalewski, Yao Lu, Sergey Levine, Lisa Lee, Tsang-Wei Edward Lee, Isabel Leal, Yuheng Kuang, Dmitry Kalashnikov, Ryan Julian, Nikhil J Joshi, Alex Irpan, brian ichter, Jasmine Hsu, Alexander Herzog, Karol Hausman, Keerthana Gopalakrishnan, Chuyuan Fu, Pete Florence, Chelsea Finn, Kumar Avinava Dubey, Danny Driess, Tianli Ding, Krzysztof Marcin Choromanski, Xi Chen, Yevgen Chebotar, Justice Carbajal, Noah Brown, Anthony Brohan, Montserrat Gonzalez Arenas, and Kehang Han. 2023. RT-2: Vision-Language-Action Models Transfer Web Knowledge to Robotic Control. In *7th Annual Conference on Robot Learning*. https://openreview.net/forum?id=XMQgwiJ7KSX