We would suggest to visit our website https://sites.google.com/view/hierarchical-language-agent/ for more information. Source code is released at https://github.com/anony214/HLA.

A ENVIRONMENT DETAILS

The original Overcooked benchmark [60] features a relatively straightforward menu and cooking process. In each gameplay session, a human player collaborates with an AI player to fulfill the orders. There is only one type of order, which is an onion soup made from three onions. Completing an order involves merely four steps, i.e., retrieving the necessary ingredients, mixing them in a pot to create a soup, plating it, and serving at the counter. Afterwards, the players receive a reward of 20. Subsequent research expands upon this environment. For example, in [7], players must use a fire extinguisher to put out the randomly appearing fire.

To make the benchmarks more challenging, we incorporate mechanisms from the original Overcooked game and make the following extensions:

- Diverse Ingredients. We offer three ingredients, i.e., onion, tomato, and lettuce.
- Chopping Mechanism. All ingredients must be chopped on the chop board before they are mixed and put in the pot. Player needs to interacts with the chop board 8 times to chop the ingredient.
- New Dishes. We design four dishes: *Alice Soup*, made of one onion and one lettuce; *Bob Soup*, made of one tomato and one lettuce; *Cathy Soup*, made of one tomato and one onion; and *David Soup*, which includes all three ingredients. The cooking time for all soups is 15 seconds.
- Fire Mechanism. Leaving the cooked soup in pot for 25 seconds would overcook the soup and set the pot on fire. Players must use a fire extinguisher to put out the fire. The putout process takes 5 seconds. After the fire is put out, players can plate the overcooked food and discard it.
- Trash Can. Players can dispose of any unwanted ingredients or overcooked dishes in a trash can to free up space.
- Order Timeout. During the game, soup orders appear randomly. Orders for Alice, Bob, and Cathy Soup are valid for 60 seconds, while orders for David Soup last 70 seconds. If finished in time, Alice, Bob, and Cathy soups give a reward of 15, and David soups give a reward of 20. Otherwise, the order disappears and gives a negative reward of -5.
- Human-AI Chat Interface: To simulate natural human interactions in the original game, we add a chat interface for communication between the human player and the AI player. The human player can either pause the game and communicate with the AI using a chat window or speak directly during gameplay. The AI can respond via text or speech.

The action space of the extended Overcooked testbed is thus composed of two parts: the atomic actions and the texts for communication. Specifically, the atomic actions include only four elements, i.e., *up, down, left, and right*, which control the character's position and its interaction with other objects.

We design 4 distinct maps for the extended Overcooked testbed. In each game, a human player (the pink beard character) collaborates with an AI player (the blue character) to complete the orders. Whenever an order is fulfilled or timeouts, a new order appears, ensuring that there are 3 valid orders in the game at any given time. Each game lasts for 100 seconds, and the default action frequency of the AI player is 2.5 Hz, resembling the natural action frequency of human players. The first two maps assess the general cooperation. *Ring* employs a ring-like layout, while *Bottleneck* creates a bottleneck that encourages the human player and the AI player to stay within a certain area. The third map, *Partition*, separates the two players, necessitating task division and cooperative coordination. In the fourth map, *Quick*, we raise the number of concurrent orders to 4 and the speed of the player to 3.5 Hz to intensify the gameplay.

To finish an order, players must follow a precise sequence of steps: retrieving the necessary ingredients, chopping them, assembling them as per the recipe, cooking them to make a dish, plating the dish before it is charred, and finally, serving it at the counter. If a soup cooks for a long time, it gets overcooked and the pot catches fire. Players must putout the fire with a fire extinguisher, plate the charred soup, and drop it into the trash bin.

B METHOD DETAILS

B.1 Prompt Details

B.1.1 Prompt of Slow Mind in Intention Reasoning Stage. As discussed in Sec. 4.2, in Intention Reasoning Stage, HLA takes in the human's chat message and reasons the human intention. Here, we divide the chat message into 6 categories, including useless message, short-term request, complicated instruction, long-term request, questions, and others. If the human player asks a question, Intention Reasoning Stage passes the question directly to Chat & Assessment Stage, which then generates a chat response.

The full prompt is shown in Fig. 10. Mutable prompt components are marked in < >, such as < *SoupOrders* >.

B.1.2 Prompt of Slow Mind in Chat & Assessment Stage. As discussed in Sec. 4.2, Slow Mind generates chat messages, whereas Chat & Assessment Stage assesses intention completion. Depending on whether the human intention is satisfied, Slow Mind works in different mode. If the human intention is not satisfied, we ask the Slow Mind to output its inner reasoning, intention completion assessment, and chat message. The full prompt of this mode is shown in Fig. 11. If the human intention is satisfied, we prompt the Slow Mind to output chat message response only, as shown in Fig. 12.

B.1.3 Fast Mind. As discussed in Sec. 4.3, the Fast Mind uses a conditional prompt depending on whether the human intention is inferred and whether it is satisfied.

The full prompt is shown in Fig. 13, where the control conditions are marked in orange color. Fast Mind takes human chat message, human intention reasoned by Slow Mind, or chat message generated by Slow Mind as input, depending on the condition.

B.2 Script Policy

As detailed in Sec. 4.4, we have designed seven types of macro actions and implemented a hard-coded script policy to perform them in the Executor. Each macro action type serves a specific function. Certain types of macro actions necessitate a specific target.

In a given environmental state, the utility of each macro action can be assessed, as represented by $V(a|s)$ in Eq. 1. This assessment

is informed by human expert knowledge and takes into account factors such as the items present on the map, the current soup orders, and the availability of each macro action. For instance, the macro action "Plate Alice Soup" is assigned a base utility value of 0.5 when there is an outstanding order for Alice Soup. If no such order exists, the base utility value defaults to 0. However, should an Alice Soup remain in the pot for an extended period and risk becoming overcooked, the utility value of this macro action increases to 1.0. This heightened value serves as an incentive for the agent to prioritize plating the soup promptly.

The seven types of macro actions are as follows.

- *Chop:* Transform an ingredient into its chopped form. The target can be Onion, Lettuce or Tomato. The value can be 0 or 0.5, depending on whether the ingredient needs to chop in current situation.
- *Mix:* Combine chopped ingredients for cooking. The target can be Alice Ingredients, Bob Ingredients, Cathy Ingredients, or David Ingredients. The value can be 0 or 0.52, depending on whether there is an unfinished order that needs the specified mixed ingredients.
- *Cook:* Utilize the mixed ingredients to cook a soup. The target can be Alice Soup, Bob Soup, Cathy Soup, or David Soup. The value can be 0 or 0.54, depending whether there is an unfinished order that needs cooking the specified soup.
- *Plate:* Plate a soup. The target can be Alice Soup, Bob Soup, Cathy Soup or David Soup. The value can be 0 or $0.56 - 1$, depending on whether the soup needs serving in hurry, and how long it overcooks.
- *Serve:* Deliver a plated soup to the delivery point. The target can be Alice Soup, Bob Soup, Cathy Soup, or David Soup. The value can be 0 or $0.58 - 1$, depending on the remaining time of soup order.
- *Putout:* Extinguish a fire on a pot using an extinguisher. No target needs to be specified. The value is always 0.6.
- *Drop:* Plate overcooked soup and discard it into a bin. No target needs to be specified. The value is always 0.6.

Most macro actions require at least one empty grid for execution. We have integrated an additional function to all of the macro actions to clean the grids if necessary.

Each macro action is flagged as available when it can be executed immediately. For instance, "Plate Alice Soup" becomes available when there is a reachable pot with cooked Alice Soup and a reachable plate. There might be instances when a macro action cannot proceed halfway, for example, the soup gets overcooked when the agent is plating it. Under these circumstances, the macro action will return a "Failed" message, prompting the agent to generate a new macro action. The implementation details of the script policy can be found in the open source code https://github.com/anony214/HLA.

A macro action like "Chop Tomato" involves a sequence of atomic actions including directional movements such as up, down, left, or right. The script policy in this context employs Breadth-First Search (BFS) for efficient pathfinding, while also continuously tracking the environment's state during interaction. For example, in executing the macro action "Chop Tomato," the agent follows a structured procedure: it navigates to the tile where tomatoes are located, picks up a tomato, places it on an unoccupied cutting board, and then chops it. After chopping, the agent moves the chopped tomato to a vacant counter area for future use. Within this workflow, specific actions like "move to tomato tile," "place tomato on cutting board," and "transfer chopped tomato to counter" are managed by the BFS algorithm. These actions are further broken down into a series of atomic actions, each involving movement in designated directions. A key aspect of this script policy is its design to minimize interference with the human player's actions.

C IMPLEMENTATION OF BASELINE AGENTS

C.1 Slow-Mind-Only Agent

In Slow-Mind-Only Agent (SMOA), Fast Mind is discarded. Thus, Chat & Assessment Stage in Slow Mind takes in available macro actions as additional input and generates macro actions directly. If the generated macro action is not legal, the LLM regenerates a new one. The Action Filter in Slow Mind of HLA needs to combine the value of each macro action and the probability of LLM choosing the specific action. Since GPT-3.5 does not provide a valid API to evaluate such probability, we discard Action Filter in SMOA.

The workflow of Slow-Mind-Only Agent is shown in Fig. 1, where Fast Mind is discarded. The full prompt of Chat & Assessment Stage in Slow-Mind-Only Agent is shown in Fig. 14. In the prompt, we append an additional chat round to let the LLM generate a macro action.

C.2 Fast-Mind-Only Agent

In Fast-Mind-Only Agent (FMOA), Slow Mind is discarded, including both Intention Reasoning Stage and Chat & Assessment Stage. The Fast Mind in FMOA takes in current soup orders, current items, and the history of human commands as additional input. The LLM also generates a chat message aside from a macro action in Fast Mind. Due to the absence of Chat & Assessment Stage, FMOA cannot tell whether the human intention is satisfied, and therefore cannot calculate the dynamic adjustment term α in Eq. (1). We set α to be the same as in HLA when human intention is not satisfied. We set the maximum length for macro action history to be 9, which is roughly the length of finishing 2 soup orders from scratch (Cook 1 Alice Soup needs 2 Chop, 1 Mix and 1 Cook, while David Soup needs 3 Chop, 1 Mix and 1 Cook). The action history is cleared once the human issues a new command. We assume the human intention is satisfied once the action history reaches the maximum length.

The workflow of Fast-Mind-Only Agent is shown in Fig. 2, where Slow Mind is discarded. The full prompt is shown in Fig. 15. The prompt conditions on whether the human intention is satisfied.

C.3 No-Executor Agent

In No-Executor Agent (NEA), the Executor is discarded, and the Fast Mind has to generate atomic actions instead of macro actions. To provide spatial information for the Fast Mind, we append the positions of items and players to the input of Fast Mind.

The workflow of No-Executor Agent is shown in Fig. 3, where Executor is discarded. The full prompt is shown in Fig. 16. In the prompt, we keep the conditional prompt module and add additional position information.

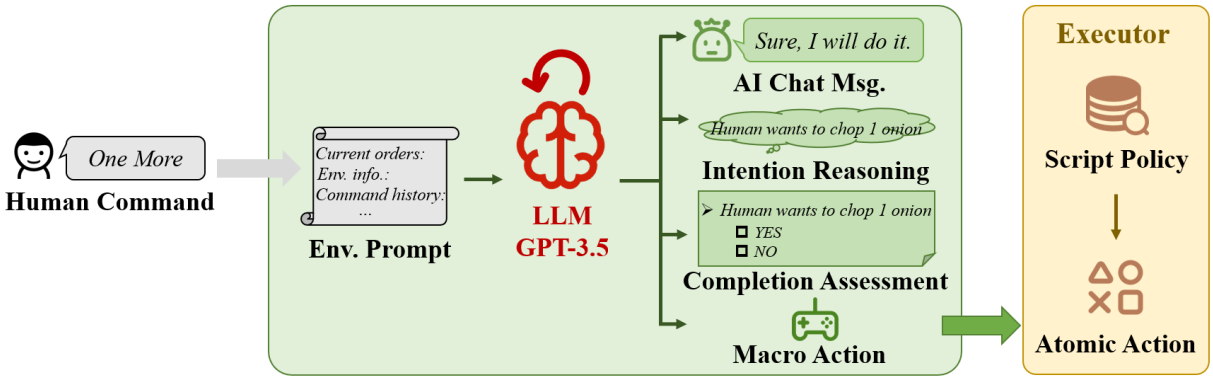


Figure 1: Workflow of Slow-Mind-Only Agent. The Fast Mind is discarded, and Slow Mind generates macro actions directly.

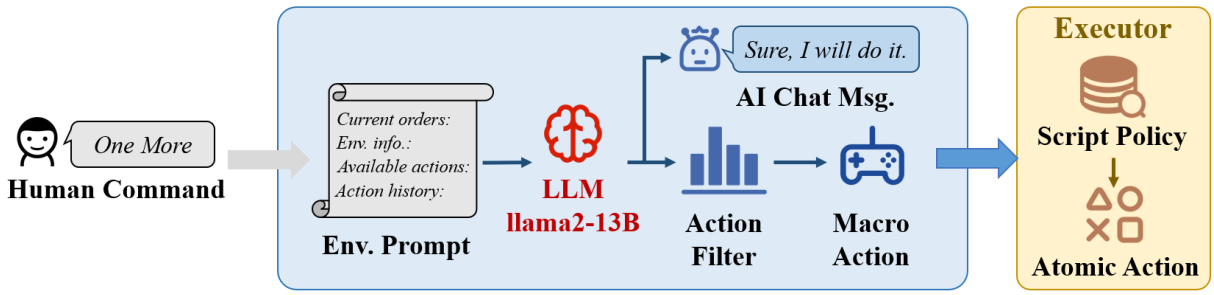


Figure 2: Workflow of Fast-Mind-Only Agent. The Slow Mind is discarded, and Fast Mind generates chat message directly.

D IMPLEMENTATION DETAILS OF COMMAND SET

D.1 Command Set for Latency Test

We construct a command set to simulate the real human commands. During the latency test, we do not concern about the actual actions of agents but the latency of their response, so we issue the commands sequentially with a 20-second interval in one round of game play. The response time of each agent is recorded. The test cases are outlined below.

(1) You are free to do anything.
(2) Try your best to earn more points.
(3) Focus on the orders.
(4) Chop 3 Lettuce.
(5) Chop 1 Onion.
(6) Chop 2 more.
(7) Cook Bob Soup.
(8) Cook it again.
(9) Alice soup is about to timeout!
(10) Watch out for the Cathy Soup order.
(11) Help me with the third soup on the orders please.
(12) Chop more vegetables.
(13) Aba Aba. Chop 1 potato.
(14) What are the orders?
(15) What is Alice Soup?

Metric. 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.

The atomic action latency is the latency of an atomic action. For SMOA, FMOA and HLA, the AI agent only suffers from the latency of generating an macro action. In other words, these agents has the latency of generating the first atomic action after receiving a human command, but can output the following atomic actions swiftly once the macro action is generated. This is because Executor can translate macro action in to atomic action with minimal latency. For NEA, the atomic action is directly generated by Fast Mind, which means NEA suffers from a latency every time it produces an atomic action.

To fairly compare the atomic action latency, we calculate the mean atomic action generation latency for SMOA, FMOA and HLA, which is more consistent with the subjective feelings of human players. For SMOA, FMOA and HLA, the atomic action latency is defined as $T_a = \frac{T_m}{N_a}$, where T_m is the generation latency of the first macro action after receiving a human command, and N_a is the number of atomic actions in this macro action. For NEA, the atomic action latency is defined as the generation latency of the first atomic action after receiving a human command.

D.2 Complex Command Set

As discussed in Sec. 5.4, we construct a command set consist of a total of 30 complex commands for 3 challenges mentioned in Sec. 3.2.

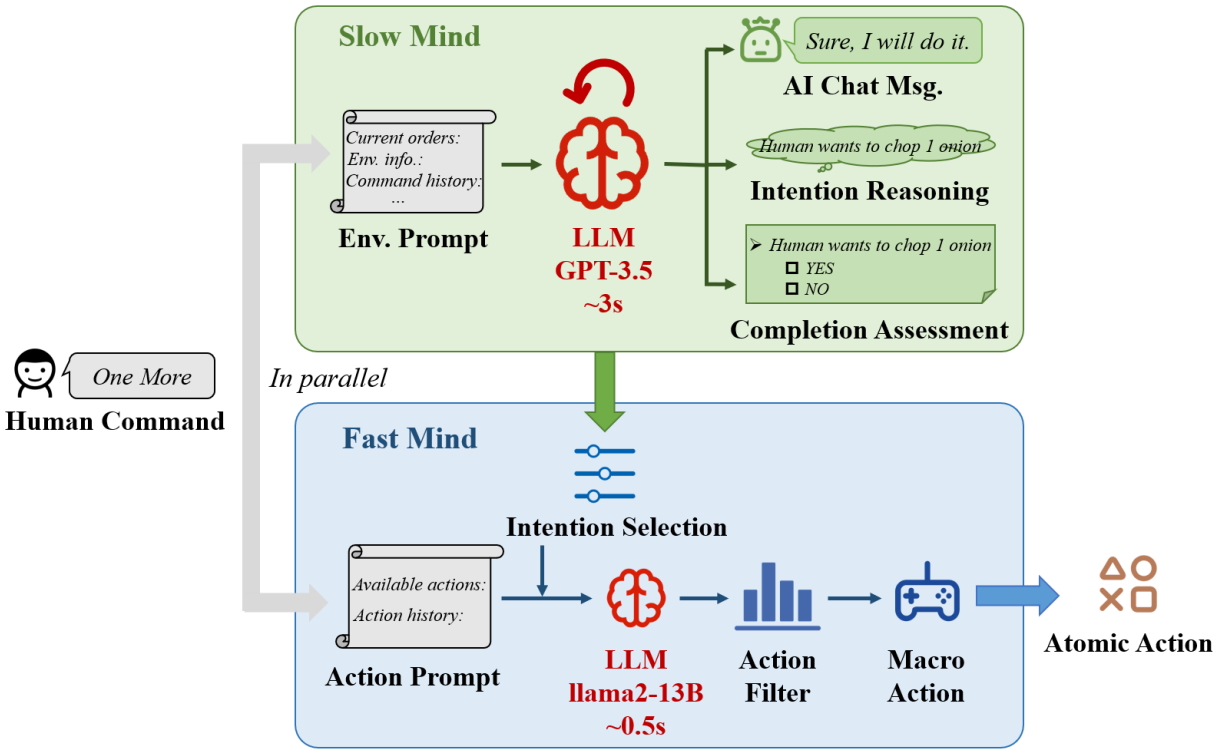


Figure 3: Workflow of No-Executor Agent. The Executor is discarded, and Fast Mind generates atomic actions directly.

Experiment are conducted using the Map *Quick*. The commands for quantity specification challenge are as follows.

(1) Chop 1 Onion.
(2) Chop two onions.
(3) Please chop 3 onions.
(4) Cut one Tomato.
(5) Help me cut 2 tomatoes.
(6) 3 chopped tomatoes please.
(7) Chop 1 Lettuce.
(8) 2 lettuces chop.
(9) help me to chop 3 lettuces.
(10) Cook Alice Soup once.

The commands for semantic analysis challenge are as follows.

(1) I need more onions
(2) Chop but except tomato and lettuce.
(3) Why are we always short of tomatoes?
(4) Just pass me toma and don't ask why.
(5) Can't you see the lettuce, uh?
(6) The green cabbage looks perfect!
(7) Bob Soup is about to timeout!
(8) Oh god, I forget the alice soup order.
(9) Come on! There is a cathy order!
(10) D soup!

The commands for ambiguous reference challenge are as follows.

(1) Chop 2 Onions -> Chop it again.
(2) Chop 3 Tomatoes -> One more please.
(3) Cut one lettuce -> Cut more!
(4) Cook Bob Soup. -> Cook it again!
(5) Cook 2 cathy soup, -> Can you do it again?
(6) Cook david soup once -> Help me with that again.
(7) Cook the first soup in the orders
(8) Cook the second order now!
(9) The third soup order should be cooked
(10) Please help me cook the last soup order

During the test of each command, the soup orders are carefully designed and fixed. For chopping commands, the target ingredient doesn't appear in any soup orders. For cooking commands, we ensure that the required soup is not part of any soup orders, except for the final 4 commands in ambiguous reference challenge, where the commands directly correspond with the orders.

E ADDITIONAL RESULTS

All experiments were conducted on a computer equipped with A100-80G GPU.

E.1 Latency of Each Module

In addition to the end-to-end latency discussed in Sec. 5.2, we also measure the latency of each module of HLA and baseline agents, as shown in Tab. 5, where IR denotes the Intention Reasoning Stage, CA denotes the Chat & Assessment Stage, and MA denotes Macro Action Generation in the Fast Mind. Both SMOA and FMOA suffers from high latency, which originates from generating macro actions and giving chat message at the same time. This further validates the hierarchical design of HLA which can decouple these components.

E.2 Visualization of Simple Commands

As discussed in Sec. 5.3, we use two simple commands, *No Command* and *One Command*, to test the cooperative ability of HLA and baseline agents. The visualization results of No-Executor Agent (NEA), Slow-Mind-Only Agent (SMOA), Fast-Mind-Only Agent (FMOA), and HLA are shown in Fig. 4, Fig. 5, Fig. 6, and Fig. 7, respectively. The game screenshots are captured at 1/3 of gameplay, 2/3 of gameplay, and upon completion of the game. The game score can be found in the bottom-left corner of the screenshot. The blue text within the screenshot shows the human player chat message and AI agent chat response.

NEA is stuck at the top left corner of the map. This is because NEA continuously gives the atomic action of moving *left* and is stuck next to the tomato tile. SMOA successfully mixes the ingredients and cooks soups. But upon completion of the game, the AI player fails to plate the soup in time, and thus overcooks the soup and sets the pot on fire. FMOA gives hallucinating chat response. In *No Command*, at 2/3 of gameplay, FMOA talks about "the burning pot", but no pot is on fire at that time. In *One Command*, at 2/3 of gameplay, FMOA still talks about "Bob Soup is almost ready" and tries to cook Bob Soup, but the Bob Soup is already served prior and there is no Bob Soup order at that moment.

HLA manages to use all 3 pots simultaneously upon completion of the game in *One Command*. In *One Command*, at 2/3 of gameplay, HLA talks about "Cathy Soup is almost charred" and plates Cathy Soup afterwards, showing a consistency of its action and chat message. This chat message comes out before the screenshot is taken.

We suggest visiting our website for more video demonstrations.

E.3 Comparison of Different LLMs

In this part, we analyze the performance differences when using various large language models (LLMs) within the Slow Mind and the Fast Mind. Specifically, "HLA-Fast7B" indicates the use of LLaMA2-7B as the Fast Mind instead of LLaMA2-13B, while "HLA-Slow70B" refers to the use of LLaMA2-70B in the Slow Mind, replacing GPT-3.5. The results regarding macro action latency and atomic action latency are detailed in Tab. 1. Tab. 2 presents the average game scores for scenarios under *No Command* and *One Command* test cases on Map *Quick*. Additionally, Tab. 3 outlines the average success rate and completion time when processing complex commands.

Tab. 2, Tab. 2 and Tab. 3, illustrate the performance of HLA-Fast7B, revealing a reduction of 29% in macro action latency and 25% in atomic action latency compared to HLA. This enhanced performance is primarily attributed to the lower computational requirements of LLaMA2-7B. However, a notable limitation of HLA-Fast7B is its diminished efficacy in handling "One Command" test cases and in interpreting semantically complex commands. As a result, for the role of the Fast Mind, we have opted for LLaMA2-13B, which offers a well-rounded balance between the reasoning capability and the inference speed. On the other hand, HLA-Slow70B exhibits similar macro and atomic action latencies but struggles significantly with both simple and complex commands. This underscores the critical role of the Slow Mind's strong reasoning capabilities. Consequently, in our study, we have chosen GPT-3.5 as the Slow Mind due to its superior performance in these areas.

E.4 Interpreting Complex Commands

Tab. 2 in the main paper illustrates the success rate of HLA in interpreting different subsets of complex commands. It was observed that 10% of semantically complex commands and 30% of ambiguous commands were not successfully interpreted. Additionally, Tab. 4 provides a detailed analysis of the failure rates attributable to the Slow Mind and the Fast Mind for these two subsets of commands. *HLA(slowmind)* denotes the failure rate linked to the Slow Mind, focusing on incorrect reasoning of intentions or assessment of task completion. *HLA(fastmind)* indicates the failure rate associated with the Fast Mind, where incorrect reasoning and assessment are identified and replaced with correct ones, followed by a recalculation of the failure rate.

As indicated in Tab. 4, the Slow Mind is identified as the cause of all observed failures. The Slow Mind incorrectly interprets human intentions as "None" or wrong intentions, which leads to the failures. For example, the semantic command (3), "Why are we always short of tomatoes?". The Slow Mind interprets the intention behind this command as "None". Similarly, with the ambiguous command (6), "Cook David soup once -> Help me with that again", the intention deduced by the Slow Mind is incorrectly identified as "Cook Bob Soup once and Cook Alice Soup once."

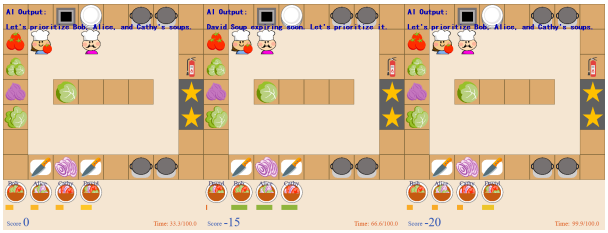
Further examination reveals that when we provide the correct human intentions and task completions to the Fast Mind, it still exhibits a 33% failure rate. For example, consider the ambiguous command (5), "Cook 2 Cathy soup, -> Can you do it again?". Both the Slow Mind and Fast Mind fail in this instance. The Slow Mind misinterprets the human intention as "None." However, even when supplied with the correct intention, the Fast Mind still executes some unnecessary macro actions before cooking the second Cathy Soup. This inefficiency results in time wastage and failure to complete the command within the designated time limit.

E.5 Human Studies

We additionally report the latency of each module, end-to-end latency, hit rate of macro actions generated by Fast Mind, detailed human preference, and visualized replay in the following subsections.

E.5.1 Details on Behavior Analysis. As mentioned in Sec. 5.5.2, we calculate the rate of valuable macro actions, i.e., $p_{valuable} = \frac{N_{valuable}}{N_{all}}$, where $N_{valuable}$ is the number of the specified macro actions that are valuable, and N_{all} is the total number of the specified macro actions. The detailed definition of useful macro actions are as follows.

- *Chop.* The moment the ingredient is put onto the cut board, a relevant unfinished order that hasn't been cooked exists, and no chopped form of this ingredient is on the map for this order.
- *Cook.* The moment the mixed ingredients are put into an empty pot, an unfinished order that hasn't been cooked exists.
- *Serve.* The moment the soup is served to the delivery point, an unfinished order of the specified soup exists.



(a) No Command

(b) One Command

Figure 4: Visualization result of NEA for latency test on simple command set. Screenshots are captured at $1/3$ of gameplay, $2/3$ of gameplay, and upon completion of the game.

(a) No Command

(b) One Command

Figure 5: Visualization result of SMOA for latency test on simple command set. Screenshots are captured at $1/3$ of gameplay, $2/3$ of gameplay, and upon completion of the game.

(a) No Command

(b) One Command

Figure 6: Visualization result of FMOA for latency test on simple command set. Screenshots are captured at $1/3$ of gameplay, $2/3$ of gameplay, and upon completion of the game.

(a) No Command

(b) One Command

Figure 7: Visualization result of HLA for latency test on simple command set. Screenshots are captured at $1/3$ of gameplay, $2/3$ of gameplay, and upon completion of the game.

Besides, we also examine the rate of fire accidents, i.e., $p_{fire} = \frac{R_{fire}}{R_{all}}$, where R_{fire} is the number of game rounds that at least one pot catches fire, and R_{all} is the number of total game rounds.

Additional results of each map can be found in Tab. 6, Tab. 7, Tab. 8 and Tab. 9. In *Partition*, the AI player cannot get access to

	Mac. Act. Latency(s)	Ato. Act. Latency(s)
HLA	1.07 (0.22)	0.08 (0.06)
HLA-Fast7B	**0.76 (0.11)**	**0.06 (0.07)**
HLA-Slow70B	1.11 (0.20)	**0.06 (0.02)**

Table 1: Macro action latency and atomic action latency of employing different LLM in HLA. The format is "mean (standard deviation)".

	No Command	One Command
HLA	55.0 (5.5)	**47.0 (11.2)**
HLA-Fast7B	**64.0 (2.0)**	40.0 (16.4)
HLA-Slow70B	43.0 (16.3)	31.0 (16.4)

Table 2: Average game score of employing different LLM in HLA. The format is "mean (standard deviation)".

AI Agents	Quantity		Semantics		Ambiguity	
	Suc.↑	Time↓	Suc.↑	Time↓	Suc.↑	Time↓
HLA	**1.00**	**13.30**	**0.90**	**17.13**	**0.70**	27.78
HLA-Fast7B	**1.00**	14.71	0.70	24.12	**0.70**	**26.95**
HLA-Slow70B	0.50	37.14	0.40	38.04	0.40	44.98

Table 3: Success rate and completion time for complex commands of employing different LLM in HLA.

	Semantics	Ambiguity
HLA (slowmind)	100%	100%
HLA (fastmind)	33%	0%

Table 4: Failure rate of Slow Mind and Fast Mind in HLA for two subsets of complex commands.

the delivery point and doesn't serve any soup. The result of "Serve" is set to "/" in the table.

E.5.2 Latency of Each Module. Tab. 10 shows the latency of each module of different AI players in the preparation phase of human studies, and Tab. 11 shows the results in the competition phase. IR denotes Intention Reasoning Stage, CA denotes Chat & Assessment Stage, and MA denotes Macro Action Generation in Fast Mind. The results are consistent with the previous results in Sec. E.1. Simultaneous generation of macro actions and chat messages introduces significant latency into SMOA and FMOA.

E.5.3 End-to-end Latency. We report the end-to-end latency during the preparation phase and the competition phase, as shown in Tab. 13. Due to the varying network conditions among participants, the latency of chat generation for both the HLA and SMOA increases.

E.5.4 Hit Rate of Immediate Macro Action in the Fast Mind. When human's chat message arrives, Fast Mind generates a macro action directly based on it, which is referred to as an immediate macro action. Additionally, we measure the hit rate of immediate macro actions in HLA, which denotes the proportion of immediate macro actions that are consistent with final macro actions generated according to the human intention. The results are shown in Tab. 14. Despite notably faster, immediate macro actions generated by Fast Mind satisfy human intentions most of the time. This suggests that HLA is capable of adhering to human commands with a rapid action response in most scenarios.

E.5.5 Human Preference. We ask the volunteers to rank and comment on HLA, SMOA, and FMOA both after the preparation phase and the competition phase. We provide 6 ranking metrics.

- AI Effectiveness: Whether the AI player can complete the dishes and obtain high scores.
- AI Assistance: Whether the AI player can help human complete orders.
- AI Responsiveness: Whether the AI player can respond in action quickly after human issues a command.
- AI Text Communication Accuracy: Whether the information conveyed by the AI player is correct.
- AI Text Communication and Action Consistency: Whether the answers given by the AI player is consistent with the actions it takes.
- Overall Performance: Evaluate the overall performance of the AI player based on gameplay experiences.

	SMOA		FMOA	HLA		
	IR	CA+MA		IR	CA	MA
Latency(s)	0.90 (0.34)	2.87 (0.50)	3.59 (0.87)	0.74 (0.26)	1.88 (0.69)	1.00 (0.21)

Table 5: Latency of each module in latency test. IR denotes for Intention Reasoning Stage, CA denotes Chat & Assessment Stage and MA denotes Macro Action Generation in Fast Mind. Standard deviations are shown in parentheses.

AI Players	*Chop* ↑	*Cook* ↑	*Serve* ↑	*Fire* ↓
SMOA	0.50	**1.00**	0.00	**0.00**
FMOA	0.74	0.85	0.80	**0.00**
HLA	**0.80**	0.98	**1.00**	**0.00**

Table 6: The ratio of valuable macro actions and the frequency of fire accidents of different AI players in *Ring* during the competition phase.

AI Players	*Chop* ↑	*Cook* ↑	*Serve* ↑	*Fire* ↓
SMOA	0.59	0.88	0.67	**0.00**
FMOA	0.80	0.67	0.92	**0.00**
HLA	**0.82**	**0.96**	**1.00**	**0.00**

Table 7: The ratio of valuable macro actions and the frequency of fire accidents of different AI players in *Bottleneck* during the competition phase.

AI Players	*Chop* ↑	*Cook* ↑	*Serve* ↑	*Fire* ↓
SMOA	0.55	0.73	/	0.47
FMOA	0.66	0.84	/	0.13
HLA	**0.79**	**0.90**	/	**0.00**

Table 8: The ratio of valuable macro actions and the frequency of fire accidents of different AI players in *Partition* during the competition phase. *Serve* macro action is marked as "/" since only the human player can serve orders in this map.

AI Players	*Chop* ↑	*Cook* ↑	*Serve* ↑	*Fire* ↓
SMOA	0.61	0.73	0.20	**0.05**
FMOA	0.81	0.88	0.83	0.10
HLA	**0.87**	**0.98**	**1.00**	0.10

Table 9: The ratio of valuable actions and the frequency of fire occurrences of different AI players in *Quick* in competition phase.

Detailed results of human preference of both phases are shown in Fig. 8 and Fig. 9. HLA remains the most preferred in all aspects. FMOA is more preferred than SMOA in all aspects.

E.5.6 Replay. We suggest visiting our website for more video demonstrations.

We present visualizations depicting four typical human players on different maps during the competition phase. These visualizations are shown Fig. 17, Fig. 18, Fig. 19, and Fig. 20 respectively. The player-128 remains silent through the whole game play. The player-221 and player-421 instruct AI players by texting messages while the player-322 instructs the AI player via speaking. The game screenshots are captured at 1/3 of gameplay, 2/3 of gameplay, and upon completion of the game. The game score can be found in the bottom-left corner of the screenshot. The blue text within the screenshot shows the human player chat message and AI agent chat response.

Cooperating with silent players, AI agent should mainly help the human player get the highest score possible. Moreover, the pace of the game is faster when use speaking to communicate, and texting can the give human player more time to think. As illustrated in the replay, collaborating with HLA results in the highest game scores. The lower latency of HLA enables more ingredients to be chopped and placed on the counters compared to other AI players in both *Bottleneck* and *Partition*. Furthermore, HLA utilizes more pots simultaneously in *Ring* and *Partition*.

	SMOA		FMOA	HLA		
	IR	CA+MA		IR	CA	MA
Latency(s)	1.04 (0.70)	3.55 (1.09)	3.27 (0.98)	1.08 (0.88)	1.80 (1.23)	0.77 (0.34)

Table 10: Latency of each module in the preparation phase of human studies. IR denotes Intention Reasoning Stage, CA denotes Chat & Assessment Stage and MA denotes Macro Action Generation in Fast Mind. Standard deviations are shown in parentheses.

	SMOA		FMOA	HLA		
	IR	CA+MA		IR	CA	MA
Latency(s)	1.07 (0.81)	3.49 (1.04)	3.25 (0.94)	0.97 (0.68)	1.73 (1.20)	0.76 (0.26)

Table 11: Latency of each module in the competition phase of human studies. IR denotes Intention Reasoning Stage, CA denotes Chat & Assessment Stage and MA denotes Macro Action Generation in Fast Mind. Standard deviations are shown in parentheses.

	SMOA	FMOA	HLA
Latency(s)	9.20 (18.74)	3.19 (1.35)	1.88 (7.55)

Table 12: End-to-end latency in the preparation phase of human studies. Standard deviations are shown in parentheses.

	SMOA	FMOA	HLA
Latency(s)	6.90 (5.96)	3.03 (1.48)	0.97 (0.99)

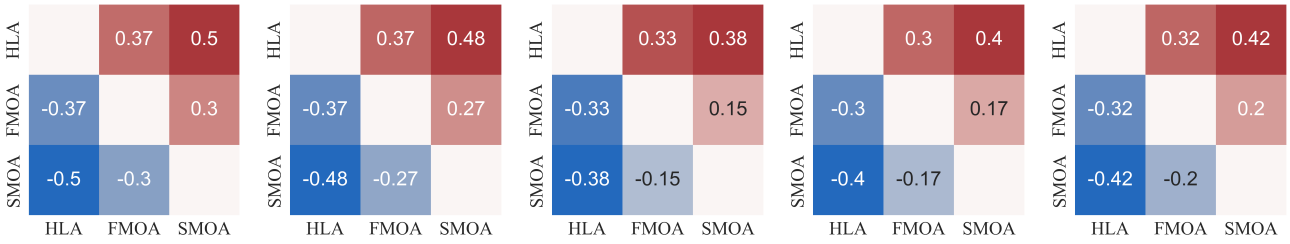
Table 13: End-to-end latency of HLA and baseline agents in human studies. Standard deviations are shown in parentheses.

Phase	*Preparation*	*Competition*
Hit Rate	0.878	0.841

Table 14: Hit Rate of Fast Mind in human studies.

(a) Effectiveness **(b) Responsiveness** **(c) Response Time** **(d) Comm. Accuracy** **(e) Comm. Consistency**

Figure 8: Human preference in the preparation stage of human studies. Numbers indicates difference of players who prefer row AI player over column AI player.



(a) Effectiveness **(b) Responsiveness** **(c) Response Time** **(d) Comm. Accuracy** **(e) Comm. Consistency**

Figure 9: Human preference in the competition stage of human studies. Numbers indicates difference of players who prefer row AI player over column AI player.

Input:
Game Scenario:
As an AI assistant in a simplified Overcooked game, work with a human player to complete soup orders. Focus on cooperation, player engagement, fulfillment, and point accrual.
Game Guidelines:
Current orders for soup vary, each with a time limit. Earn a bonus for on-time completion.
To make a soup:
a. Chop fresh vegetables - Tomato, Lettuce, Onion to obtain chopped vegetables.
b. Prepare soup ingredients with chopped vegetables once all required types are ready.
Alice: Chopped Lettuce, Onion.
Bob: Chopped Lettuce, Tomato.
Cathy: Chopped Onion, Tomato.
David: Chopped Lettuce, Onion, Tomato.
c. Cook the soup. Cooking starts once the required ingredients are ready.
Alice Soup: Alice Ingredients.
Bob Soup: Bob Ingredients.
Cathy Soup: Cathy Ingredients.
David Soup: David Ingredients.
d. Plate the cooked soup.
e. Serve the plated soup in the serving area for a shared bonus.
If a soup stays in the pot too long, it gets charred.
a. Putout: If the pot catches fire, extinguish it.
b. Drop: Discard charred soup. Put out the fire in the pot if needed.

In-game Decision:
You need to interpret the human player's message into a simpler form, which will be sent to a downstream AI without access to human message history. Your answer must be clear and succinct.

The human's message can be:
1. Useless message: Message that has no specific demand such as "Enough", "Never mind", "You are free to do anything else" or "Try your best to earn more points." translates to "None."
2. Short-term request: "Chop 4 more" means "Chop xxx 4 times.", where "xxx" should be the vegetable in past intention. Keep you answer concise and make sure the numbers are corrent. "Plate the soup now" should be "Plate Soup once."
3. Intention needs to be inferred: Sometimes you need to infer about the hidden meaning of messages. For instance, "I will cook the first order. Can you take charge of the rest?" implies "Cook xxx once and Cook xxx once." where "xxx" are the subsequent soup orders. Similarly, "xxx is handled by me." implies "Cook xxx." where the two "xxx" are different soup in the orders. Emotional and cryptic message like "The David Soup is about to timeout!" suggest "Serve David Soup once."
4. Long-term request: Messages such as "Keep chopping tomatoes" become "Always keep chopping tomatoes, and don't stop."
5. Questions: Like "What are the orders", "What is xxx Soup" or any question-like queries. You must repeat the original question completely in your output. You must leave the question to the downstream AI intactly who will answer it.
6. Special case: Messages related to asking for orders, like "Tell me the orders", "Keep telling me the orders" or "I want to know the orders" should be translated to "What are the orders now?".

If the human's intention conflicts with soup orders, you should follow the human's intention even if it is not on the orders. Always prioritize the human's message.

Any explanations, comments, tips or modal particles must not be included.

Current soup orders: <Soup Orders>

The human player's intention in the last round (which has already been satisfied):
<Human Message History>

The human player's message now:

<Human Message>

Be very careful that if the message is a question, you must repeat it completely in your answer. DO NOT ANSWER IT. You need to interpret the human player's message only if it is not a question.

Now, your answer is:

Output:

Figure 10: Prompt of Intention Reasoning Stage in Slow Mind of HLA.

Input:
Game Scenario:
As an AI assistant in a simplified Overcooked game, work with a human player to complete soup orders. Focus on cooperation, player engagement, fulfillment, and point accrual.
Game Guidelines:
Current orders for soup vary, each with a time limit. Earn a bonus for on-time completion.
To make a soup:
a. Chop fresh vegetables - Tomato, Lettuce, Onion to obtain chopped vegetables.
b. Prepare soup ingredients with chopped vegetables once all required types are ready.
Alice: Chopped Lettuce, Onion.
Bob: Chopped Lettuce, Tomato.
Cathy: Chopped Onion, Tomato.
David: Chopped Lettuce, Onion, Tomato.
c. Cook the soup. Cooking starts once the required ingredients are ready.
Alice Soup: Alice Ingredients.
Bob Soup: Bob Ingredients.
Cathy Soup: Cathy Ingredients.
David Soup: David Ingredients.
d. Plate the cooked soup.
e. Serve the plated soup in the serving area for a shared bonus.
If a soup stays in the pot too long, it gets charred.
a. Putout: If the pot catches fire, extinguish it.
b. Drop: Discard charred soup. Put out the fire in the pot if needed.

Gameplay Rounds:
Round One - Action Summary: In this stage, your task is to summarize the actions you've made that are directly beneficial to the human player's request.
Round Two - Communication: Here, you generate your chat message to be sent to the human player.
Round Three - Satisfaction Evaluation: In this round, it's your responsibility to judge whether the player's request has been fully met based on your actions.

Note that there are multiple types of human's incoming message:
1. Short term request: Like "Chop 4 times", "Chop once", "Cook 2 Soup" or "Plate once". If you have done ALL actions he requests, then it is satisfied. It is OK if you've done more than he asks. If there are still actions to be done, then it is not satisfied.
2. Long term request: Like "Always prepare", "Keep chopping", "Plating continuously", "Cook don't stop" or "Avoid serving". In these cases, the requests will never be satisfied because they need to be done continuously, even if your actions conflict with them,
3. Question: Like "What are the current orders?" or "What is xxx Soup?" You need to answer to the question in the chat message. And you must give "Yes" in the Satisfaction Evaluation round.
4. Useless message: Like "None", "Free to do anything", "No specific intention", or statement of fact like "The orders are xxx". You must "Yes" in the Satisfaction Evaluation round.

Current soup orders:
<Soup Orders with Time Limit>

Items on the map:
<Items on Map>

The human player's incoming message:
<Human Intention>

Actions you've done since the human gave the message:
<Compressed Action History>

You need to examine the current state of the game environment, the human player's message, and actions you've taken so far. Now summarize the actions you've done that are directly beneficial to the human player's request. Any action not related to

their request can be ignored.
If the human player's request is "None" or a question, just briefly summarize your current actions.
You must be honest and give actions that is surely done by yourself. Do not make up!
Keep your answer short and concise. No more than 20 words.

Output:

Generate your chat message to be send to the human. Your communication should be polite, helpful, and limited to 20 words max. Aim to demonstrate your enthusiasm and friendliness while assisting the player.
If the human player asks a question, ensure to provide an appropriate response. For example, if he asks "What are the current orders?", you should respond with the current orders and their time remaining. You also have the opportunity to inform the player of your current and planned actions.
Just give your message, with no quotation marks or emojis.

Output:

Judge whether the player's request has been fulfilled by your actions. The possible responses are "Yes" or "No".
If the human's incoming message is a question or a useless message, give "Yes".

Output:

Figure 11: Prompt of Chat & Assessment Stage in Slow Mind of HLA (if human's intention is not satisfied).

Input:
Game Scenario:
As an AI assistant in a simplified Overcooked game, work with a human player to complete soup orders. Focus on cooperation, player engagement, fulfillment, and point accrual.
Game Guidelines:
Current orders for soup vary, each with a time limit. Earn a bonus for on-time completion.
To make a soup:
a. Chop fresh vegetables - Tomato, Lettuce, Onion to obtain chopped vegetables.
b. Prepare soup ingredients with chopped vegetables once all required types are ready.
Alice: Chopped Lettuce, Onion.
Bob: Chopped Lettuce, Tomato.
Cathy: Chopped Onion, Tomato.
David: Chopped Lettuce, Onion, Tomato.
c. Cook the soup. Cooking starts once the required ingredients are ready.
Alice Soup: Alice Ingredients.
Bob Soup: Bob Ingredients.
Cathy Soup: Cathy Ingredients.
David Soup: David Ingredients.
d. Plate the cooked soup.
e. Serve the plated soup in the serving area for a shared bonus.
If a soup stays in the pot too long, it gets charred.
a. Putout: If the pot catches fire, extinguish it.
b. Drop: Discard charred soup. Put out the fire in the pot if needed.

Assuming that you have been playing the game for a while. Now you will be informed of the current situation, and need to generate your chat message to be sent to the human player.
You are recommended to give your future plan. Giving information about current orders and their time limit is also a good idea. You shouldn't focus on the Fire Extinguisher.
You answer must be concrete and informative with no more than 10 words. Just give your chat message with no explanation, no comments, no quotation marks and no emojis.

Current soup orders:
<Soup Orders with Time Limit>

Items on the map:
<Items on Map>

Actions you've done recently:
<Compressed Action History>

Now give your chat message to be sent to the human.

Output:

Figure 12: Prompt of Chat & Assessment Stage in Slow Mind of HLA (if human's intention is satisfied).

Input:
Game Situation:
You and another human player are playing a simplified version of the video game Overcooked. Your goal is to cooperatively finish a dynamically changing list of soup orders as fast as possible. The game has different orders from the original video game. There are two players: you (an AI assistant) and another human player. Your primary goal is to cooperate and make the human player feel engaged, happy, and satisfied while also earning more points.

Game Rules:
1. All available actions are: Chop Tomato, Chop Lettuce, Chop Onion, Prepare Alice Ingredients, Prepare Bob Ingredients, Prepare Cathy Ingredients, Prepare David Ingredients, Putout, Cook Alice Soup, Cook Bob Soup, Cook Cathy Soup, Cook David Soup, Plate Alice Soup, Plate Bob Soup, Plate Cathy Soup, Plate David Soup, Serve Alice Soup, Serve Bob Soup, Serve Cathy Soup, Serve David Soup, Drop.
2. There is a changing list of soup orders, each with a time limit for completion. Completing an order on time earns a bonus, while failing to do so results in losing the bonus.
3. The inverse action sequence to finish soup orders:
To finish Alice Soup order, you need to Serve Alice Soup, which needs Plate Alice Soup and Cook Alice Soup. Alice Soup can be done after you Prepare Alice Ingredients, which needs Chop Lettuce and Chop Onion.
To finish Bob Soup order, you need to Serve Bob Soup, which needs Plate Bob Soup and Cook Bob Soup. Bob Soup can be done after you Prepare Bob Ingredients, which needs Chop Lettuce and Chop Tomato.
To finish Cathy Soup order, you need to Serve Cathy Soup, which needs Plate Cathy Soup and Cook Cathy Soup. Cathy Soup can be done after you Prepare Cathy Ingredients, which needs Chop Onion and Chop Tomato.
To finish David Soup order, you need to Serve David Soup, which needs Plate David Soup and Cook David Soup. David Soup can be done after you Prepare David Ingredients, which needs Chop Lettuce, Chop Onion, and Chop Tomato.
4. If a cooked soup remains in the pot for a long time, it becomes charred, and the pot catches fire.
a. Putout: To regain the pot, you must extinguish the fire.
b. Drop: If a soup becomes charred, you must discard it.

Let's say you're playing this game and it's been a while. The human may specify his demand, and maybe you have some planning. Now you need to give your actions based on them. Please note that when you carry out an action, you just do it once. If you want to do it multiple times, you need to repeat it multiple times. If there is many subtasks in the human's demand, you need to finish them in order.
If the demand contains "Stop xxx" or "Avoid xxx", you should never do it.
If the demand contains "Focus xxx", "Keep xxx" or "Always xxx", then you should always do it, and never doing other actions.

[If intention is not reasoned]
The human's demand is:
<Human Message>
[Else If intention is reasoned and not satisfied]
The human's demand is:
<Human Intention>
[Else If intention is reasoned and satisfied]
Your planning:
<Chat Message Generated by Slow Mind>
[EndIf]

Output:
My actions are: <Macro Action History>, <next action>

Figure 13: Prompt in Fast Mind of HLA.

Input:
Game Scenario:
As an AI assistant in a simplified Overcooked game, work with a human player to complete soup orders. Focus on cooperation, player engagement, fulfillment, and point accrual.
Game Guidelines:
Current orders for soup vary, each with a time limit. Earn a bonus for on-time completion.
To make a soup:
a. Chop fresh vegetables - Tomato, Lettuce, Onion to obtain chopped vegetables.
b. Prepare soup ingredients with chopped vegetables once all required types are ready.
Alice: Chopped Lettuce, Onion.
Bob: Chopped Lettuce, Tomato.
Cathy: Chopped Onion, Tomato.
David: Chopped Lettuce, Onion, Tomato.
c. Cook the soup. Cooking starts once the required ingredients are ready.
Alice Soup: Alice Ingredients.
Bob Soup: Bob Ingredients.
Cathy Soup: Cathy Ingredients.
David Soup: David Ingredients.
d. Plate the cooked soup.
e. Serve the plated soup in the serving area for a shared bonus.
If a soup stays in the pot too long, it gets charred.
a. Putout: If the pot catches fire, extinguish it.
b. Drop: Discard charred soup. Put out the fire in the pot if needed.

Gameplay Rounds:
Round One - Action Summary: In this stage, your task is to summarize the actions you've made that are directly beneficial to the human player's request.
Round Two - Communication: Here, you generate your chat message to be sent to the human player.
Round Three - Satisfaction Evaluation: In this round, it's your responsibility to judge whether the player's request has been fully met based on your actions.
Round Four - Action Execution: You are to give your action to be carried out next.

Note that there are multiple types of human's incoming message:
1. Short term request: Like "Chop 4 times", "Chop once", "Cook 2 Soup" or "Plate once". If you have done ALL actions he requests, then it is satisfied. It is OK if you've done more than he asks. If there are still actions to be done, then it is not satisfied.
2. Long term request: Like "Always prepare", "Keep chopping", "Plating continuously", "Cook don't stop" or "Avoid serving". In these cases, the requests will never be satisfied because they need to be done continuously, even if your actions conflict with them,
3. Question: Like "What are the current orders?" or "What is xxx Soup?" You need to answer to the question in the chat message. And you must give "Yes" in the Satisfaction Evaluation round.
4. Useless message: Like "None", "Free to do anything", "No specific intention", or statement of fact like "The orders are xxx". You must "Yes" in the Satisfaction Evaluation round.

Current soup orders:
<Soup Orders with Time Limit>

Items on the map:
<Items on Map>

The human player's incoming message:
<Human Intention>

Actions you've done since the human gave the message:
<Compressed Action History>

You need to examine the current state of the game environment, the human player's message, and actions you've taken

so far. Now summarize the actions you've done that are directly beneficial to the human player's request. Any action not related to their request can be ignored.
If the human player's request is "None" or a question, just briefly summarize your current actions.
You must be honest and give actions that is surely done by yourself. Do not make up!
Keep your answer short and concise. No more than 20 words.

Output:

Generate your chat message to be send to the human. Your communication should be polite, helpful, and limited to 20 words max. Aim to demonstrate your enthusiasm and friendliness while assisting the player.
If the human player asks a question, ensure to provide an appropriate response. For example, if he asks "What are the current orders?", you should respond with the current orders and their time remaining. You also have the opportunity to inform the player of your current and planned actions.
Just give your message, with no quotation marks or emojis.

Output:

Judge whether the player's request has been fulfilled by your actions. The possible responses are "Yes" or "No".
If the human's incoming message is a question or a useless message, give "Yes".

Output:

Give your action to be carried out next. You should try to serve, plate and cook soup when possible. Select it from <Available Actions>. You can only choose one from it and not allowed to make up new action. Explanation or comment is not needed.

Output:

Figure 14: Prompt of Chat & Assessment Stage in Slow-Mind-Only Agent.

Input:
Game Situation:
You and another human player are playing a simplified version of the video game Overcooked. Your goal is to cooperatively finish a dynamically changing list of soup orders as fast as possible. The game has different orders from the original video game. There are two players: you (an AI assistant) and another human player. Your primary goal is to cooperate and make the human player feel engaged, happy, and satisfied while also earning more points.

Game Rules:
1. All available actions are: Chop Tomato, Chop Lettuce, Chop Onion, Prepare Alice Ingredients, Prepare Bob Ingredients, Prepare Cathy Ingredients, Prepare David Ingredients, Putout, Cook Alice Soup, Cook Bob Soup, Cook Cathy Soup, Cook David Soup, Plate Alice Soup, Plate Bob Soup, Plate Cathy Soup, Plate David Soup, Serve Alice Soup, Serve Bob Soup, Serve Cathy Soup, Serve David Soup, Drop.
2. There is a changing list of soup orders, each with a time limit for completion. Completing an order on time earns a bonus, while failing to do so results in losing the bonus.
3. The inverse action sequence to finish soup orders:
To finish Alice Soup order, you need to Serve Alice Soup, which needs Plate Alice Soup and Cook Alice Soup. Alice Soup can be done after you Prepare Alice Ingredients, which needs Chop Lettuce and Chop Onion.
To finish Bob Soup order, you need to Serve Bob Soup, which needs Plate Bob Soup and Cook Bob Soup. Bob Soup can be done after you Prepare Bob Ingredients, which needs Chop Lettuce and Chop Tomato.
To finish Cathy Soup order, you need to Serve Cathy Soup, which needs Plate Cathy Soup and Cook Cathy Soup. Cathy Soup can be done after you Prepare Cathy Ingredients, which needs Chop Onion and Chop Tomato.
To finish David Soup order, you need to Serve David Soup, which needs Plate David Soup and Cook David Soup. David Soup can be done after you Prepare David Ingredients, which needs Chop Lettuce, Chop Onion, and Chop Tomato.
4. If a cooked soup remains in the pot for a long time, it becomes charred, and the pot catches fire.
a. Putout: To regain the pot, you must extinguish the fire.
b. Drop: If a soup becomes charred, you must discard it.

Let's say you're playing this game and it's been a while. The human may specify his demand, and maybe you have some planning. Now you need to give your actions based on them. Please note that when you carry out an action, you just do it once. If you want to do it multiple times, you need to repeat it multiple times. If there is many subtasks in the human's demand, you need to finish them in order.
If the demand contains "Stop xxx" or "Avoid xxx", you should never do it.
If the demand contains "Focus xxx", "Keep xxx" or "Always xxx", then you should always do it, and never doing other actions.

[If intention is not satisfied]
The human player's demand in the last round (which has already been satisfied):
<Human Message History>
The human's demand is:
<Human Message>
[EndIf]

Current soup orders:
<Soup Orders with Time Limit>

Items on the map:
<Items on Map>

Output:
My actions are: <Macro Action History>, <next action>

[If intention is not satisfied]
Generate your chat message to be send to the human. Your communication should be polite, helpful. Aim to demonstrate your enthusiasm and friendliness while assisting the player.
If the human player asks a question, ensure to provide an appropriate response. For example, if he asks "What are the current

orders?", you should respond with the current orders and their time remaining.
You also have the opportunity to inform the player of your current and planned actions.
Just give your message, with no quotation marks or emojis.
[Else If intention is satisfied]
Now give your chat message to be sent to the human.
[EndIf]

Output:

Figure 15: Prompt of Fast-Mind-Only Agent.

Input:
Game Situation:
You and another human player are playing a simplified version of the video game Overcooked. Your goal is to cooperatively finish a dynamically changing list of soup orders as fast as possible. The game has different orders from the original video game. There are two players: you (an AI assistant) and another human player. Your primary goal is to cooperate and make the human player feel engaged, happy, and satisfied while also earning more points.

Game Rules:
1. All available actions are: left, right, up, down, which will change your location by (-1, 0), (1, 0), (0, 1) and (0, -1) respectively. When you stand next to a grid, you can move towards it to interactive with it, for example, pick up things from table or cook a soup. 2. There is a changing list of soup orders, each with a time limit for completion. Completing an order on time earns a bonus, while failing to do so results in losing the bonus.
3. The inverse action sequence to finish soup orders:
To finish Alice Soup order, you need to Serve Alice Soup, which needs Plate Alice Soup and Cook Alice Soup. Alice Soup can be done after you Prepare Alice Ingredients, which needs Chop Lettuce and Chop Onion.
To finish Bob Soup order, you need to Serve Bob Soup, which needs Plate Bob Soup and Cook Bob Soup. Bob Soup can be done after you Prepare Bob Ingredients, which needs Chop Lettuce and Chop Tomato.
To finish Cathy Soup order, you need to Serve Cathy Soup, which needs Plate Cathy Soup and Cook Cathy Soup. Cathy Soup can be done after you Prepare Cathy Ingredients, which needs Chop Onion and Chop Tomato.
To finish David Soup order, you need to Serve David Soup, which needs Plate David Soup and Cook David Soup. David Soup can be done after you Prepare David Ingredients, which needs Chop Lettuce, Chop Onion, and Chop Tomato.
4. If a cooked soup remains in the pot for a long time, it becomes charred, and the pot catches fire.
a. Putout: To regain the pot, you must extinguish the fire.
b. Drop: If a soup becomes charred, you must discard it.

Let's say you're playing this game and it's been a while. The human may specify his demand, and maybe you have some planning. Now you need to give your actions based on them. Please note that when you carry out an action, you just do it once. If you want to do it multiple times, you need to repeat it multiple times. If there is many subtasks in the human's demand, you need to finish them in order.
If the demand contains "Stop xxx" or "Avoid xxx", you should never do it.
If the demand contains "Focus xxx", "Keep xxx" or "Always xxx", then you should always do it, and never doing other actions.

Items on the map:
<Position Information of Items on Map>
<Position Information of All Players>

[If intention is not reasoned]
The human's demand is:
<Human Message>
[Else If intention is reasoned and not satisfied]
The human's demand is:
<Human Intention>
[Else If intention is reasoned and satisfied]
Your planning:
<Chat Message Generated by Slow Mind>
[EndIf]

Output:
My action is to move towards <next action>

Figure 16: Prompt of Fast Mind in No-Executor Agent.

(a) SMOA

(b) FMOA

(c) HLA

Figure 17: Visualization result of player-128 in *Ring* during the competition phase. Screenshots are captured at 1/3 of gameplay, 2/3 of gameplay, and upon completion of the game.

(a) SMOA

(b) FMOA

(c) HLA

Figure 18: Visualization result of player-322 in *Bottleneck* during the competition phase. Screenshots are captured at 1/3 of gameplay, 2/3 of gameplay, and upon completion of the game.

(a) SMOA

(b) FMOA

(c) HLA

Figure 19: Visualization result of player-221 in *Partition* during the competition phase. Screenshots are captured at $1/3$ of gameplay, $2/3$ of gameplay, and upon completion of the game.

(a) SMOA

(b) FMOA

(c) HLA

Figure 20: Visualization result of player-421 in *Quick* during the competition phase. Screenshots are captured at $1/3$ of gameplay, $2/3$ of gameplay, and upon completion of the game.