Conversation-based Multi-agent Reinforcement Learning with LLM as a Policy

Anonymous ACL submission

Abstract

Reinforcement learning (RL) benefits from Large language models (LLMs) for improved reasoning and planning, but their application in Multi-agent reinforcement learning (MARL) remains challenging due to communication conflicts. We propose a novel framework where agents engage in structured multi-round conversations before taking actions, ensuring better coordination and decision-making. By leveraging LLMs' reasoning capabilities and integrating techniques like Chain of Thought reasoning, our approach enhances collaboration in MARL. Experimental results show improved efficiency and scalability, bridging the gap between single-agent and multi-agent LLM applications.

1 Introduction

Reinforcement Learning (RL) has been widely adopted to tackle sequential decision-making problems. However, traditional RL algorithms often suffer from low sample efficiency and lack explainability, limiting their applicability in complex, real-world scenarios. To address these issues, researchers have begun integrating Large Language Models (LLMs) into RL frameworks. LLMs have demonstrated strong reasoning and planning capabilities, making them a promising tool for enhancing RL performance.

Existing research has explored the use of LLMs in various roles, such as planners, coders, and executors(Cao Yuji, et al., 2024). While these studies have shown promising results in single-agent settings, extending LLM-based frameworks to Multi-Agent Reinforcement Learning (MARL) presents unique challenges. One critical issue is the potential for conflicts in communication when multiple agents exchange messages simultaneously. For instance, agents may send conflicting instructions, leading to inconsistencies in actions and reduced collaboration efficiency. Addressing this problem requires designing effective communication mechanisms that can handle coordination and information-sharing among agents.

Several techniques, such as Chain of Thought (CoT) (Jason Wei, et al., 2023) reasoning, Reflection(Noah Shinn, et.al, 2023) , and Monte Carlo Tree Search, have been proposed to enhance the decision-making capabilities of LLMs. These approaches have proven effective in single-agent contexts by improving reasoning and learning efficiency. However, their application to MARL remains underexplored. In multi-agent systems, actions taken by one agent can directly influence the environment and the decisions of other agents. Therefore, it is essential to incorporate mechanisms that allow agents to exchange and utilize each other's information, such as intentions, thoughts, and actions, to achieve optimal collaboration.

In this paper, we propose a novel method to address the communication challenges in MARL. Before deciding on actions, agents engage in structured conversations over a specified number of rounds. Through these conversations, agents can exchange information, align their strategies, and reach agreements. By sharing sufficient information, the proposed method reduces inconsistencies between messages and actions, thereby enhancing coordination and decision-making efficiency. Our approach leverages the reasoning capabilities of LLMs and integrates them into a collaborative framework designed specifically for multi-agent environments.

We evaluate our method in multi-agent BabyAI environments, comparing different communication strategies with Reflexion and Llama 3.1 (70B). Results show that structured multi-round communication, especially the two-round alternating proposal method, significantly improves coordination efficiency. Compared to MAPPO, our approach achieves faster learning and higher performance without complex reward design, highlighting the

potential of LLM-driven MARL systems.

2 Related Work

2.1 Communication in MARL

Effective communication among agents in MARL is a crucial aspect that has gained significant attention in recent years. The ability for agents to exchange information enables coordination, mitigates conflicts, and facilitates the accomplishment of complex tasks in shared environments. Over the years, various methods have been developed to tackle the challenges associated with communication, ranging from centralized to decentralized frameworks, and adaptive protocols.

Centralized Communication Approaches. Centralized methods rely on a central entity to aggregate and distribute information among agents. For instance, DIAL (Differentiable Inter-Agent Learning) (Jakob N. Foerster, et al., 2016) employs centralized training with decentralized execution (CTDE). During training, agents share information to learn implicit communication protocols, which are later executed independently. While these methods enable effective learning in smaller setups, scalability becomes a major challenge as the number of agents increases, often leading to communication bottlenecks and computational overhead.

Decentralized Communication Architectures. In decentralized settings, agents communicate directly with one another without a central mediator. Techniques leveraging graph neural networks (GNNs) are often used to model inter-agent interactions (Jiechuan Jiang, et al., 2020). Frameworks such as CommNet embed messages exchanged between agents into their policy networks, enabling decentralized information sharing. While these approaches improve scalability and remove reliance on a central hub, they may face difficulties in environments requiring global coordination, as local interactions might not always capture the bigger picture.

Comma: A Coordinated Multi-Agent Approach. Comma, short for Coordinated Multi-Agent Actor-Critic, is a MARL algorithm designed to enhance coordination among agents in complex environments. It employs CTDE, where a shared critic uses global information during training, while agents operate independently during execution. By incorporating an attention mechanism, Comma allows agents to prioritize relevant information from peers, addressing non-stationarity and improving coordination. Its scalability and efficiency make it suitable for cooperative tasks such as multi-agent navigation, resource allocation, and robotics (Lianyu Hu, et al., 2024).

Emerging Trends in Adaptive Communication. Recent advances focus on enabling agents to learn their own communication protocols dynamically. Methods like MADDPG (Multi-Agent Deep Deterministic Policy Gradient) with communication extensions (Ryan Lowe, et al, 2017) allow agents to determine what information to share and how to interpret received messages. These learned protocols are highly adaptable, reducing redundant communication and improving efficiency in dynamic environments. By enabling agents to refine their communication strategies based on environmental feedback, these methods pave the way for robust, scalable MARL solutions.

In summary, communication in MARL has evolved from centralized frameworks to decentralized architectures and adaptive protocols. While centralized methods are effective in smaller, controlled environments, decentralized and learned communication approaches address scalability and adaptability challenges. These advancements collectively enhance the coordination capabilities of agents in increasingly complex and dynamic settings.

2.2 Employing LLM for Multiagent Communication

Leveraging Large Language Models (LLMs) for multi-agent communication has emerged as a promising approach in reinforcement learning. In GLAM (Grounded Language Model) (Thomas Carta, et al., 2023), the agent's state, obtained from the environment, is input into the LLM to calculate probabilities for each possible action. The action with the highest probability is selected, as defined by Equation 1. Here, $LLM(a|s)$ represents the probability of action a (in text form) being output by the LLM when the state s (in text form) is input, and $\mathcal{A}$ denotes the agent's action space.

$$\pi(s) = \underset{a \in \mathcal{A}}{\operatorname{argmax}} LLM(a|s) \tag{1}$$

Using the reward r received as a result of the action, the policy is updated through fine-tuning with PPO (John Schulman, et al., 2017). This framework enables agents to quickly adapt to novel tasks by leveraging the extensive knowledge embedded

in the LLM, producing a versatile and robust agent capable of addressing a variety of challenges.

Building upon GLAM, the FAMA (Fine-Tuned Agent with Multi-Agent Learning) method (Oliver Slumbers, et al., 2024) extends this approach to a multi-agent setting. In FAMA, the LLM serves as the policy model for each agent, enabling online multi-agent learning with message exchanges. This framework enhances the independent GLAM model by incorporating inter-agent communication. The overall architecture for two agents is depicted in Figure 1.

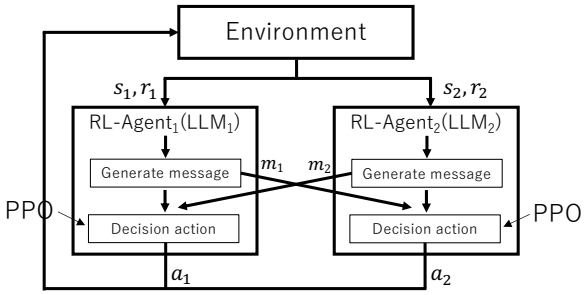

Figure 1: Overview of FAMA

In FAMA, each agent inputs its state from the environment into the LLM to generate a message intended for the other agent. Upon receiving a message, the agent combines its state with the received message, inputs this combined information into the LLM, and calculates probabilities for each action. The action with the highest probability is then selected, as defined by Equations 2 and 3. Here, $LLM_i(a|s)$ represents the probability of action a being output by agent i's LLM when state s is input. $\mathcal{A}_i$ denotes the action space of agent i, and m_i is the message sent by agent i.

$$\pi_1(s_1) = \underset{a \in \mathcal{A}_1}{\text{argmax}} LLM_1(a|s_1, m_2) \quad (2)$$

$$\pi_2(s_2) = \underset{a \in \mathcal{A}_2}{\text{argmax}} LLM_2(a|s_2, m_1) \quad (3)$$

The messages m_1 and m_2 are generated as shown in Equations 4, where $LLM_i(\cdot|s_i)$ produces the message to be sent by agent i based on its state s_i.

$$m_i = LLM_i(\cdot|s_i) \quad (4)$$

This approach leverages the knowledge and reasoning capabilities of LLMs, allowing agents not only to adapt quickly to the environment but also to learn cooperative behaviors through efficient message exchanges. By combining the agents' states with received messages, the system achieves a higher level of coordination and flexibility in multi-agent tasks.

2.3 Text-Based Learning Framework

Reflexion (Noah Shinn, et.al, 2023) introduces a novel framework to enhance language agents by leveraging language-based feedback instead of conventional weight updates through supervised or reinforcement learning. This innovative approach allows language models to refine their problem-solving abilities by modifying their inputs dynamically.

In the Reflexion framework, agents utilize a short-term memory buffer to store information gathered during each episode, including environmental observations and the agent's actions. Feedback from the environment, whether in the form of scalar values or free-form text, is processed linguistically. Agents analyze this feedback to identify potential improvements and store reflective insights in a long-term memory buffer, which persists across episodes. This reflective process enables agents to iteratively refine their decision-making strategies, outperforming traditional methods in tasks such as sequential decision-making, coding, and language reasoning (Noah Shinn, et.al, 2023).

Additionally, the Reflexion framework introduces two key communication strategies for multi-agent systems: (1) unidirectional communication and (2) alternating proposal communication. Unidirectional communication minimizes conflicts and inconsistencies by eliminating simultaneous bidirectional exchanges, ensuring clearer and more reliable interactions. Alternating proposal communication allows agents to exchange and refine action policies through iterative discussions, fostering improved coordination.

To further enhance the learning process, Reflexion replaces traditional PPO-based reinforcement learning with prompt-based adjustments driven by language feedback. This adjustment not only improves agents' inference capabilities but also enhances the explainability of their decisions, making the framework well-suited for complex reasoning and coordination tasks.

3 Proposed Method for Changing Communication Between Agents

We introduce two novel communication strategies for multi-agent systems using LLM-based agents: (1) unidirectional message transmission and (2) alternating proposals. These methods aim to address limitations in existing frameworks like FAMA by improving coordination and reducing conflicts. Additionally, the proposed approaches integrate advanced reasoning techniques such as CoT (Chain of Thought) (Jason Wei, et al., 2023) and Reflexion (Noah Shinn, et.al, 2023) to enhance decision-making and explainability.

3.1 Challenges in Bidirectional Communication

Bidirectional communication, as seen in FAMA, enables agents to exchange messages during decision-making. However, it often leads to conflicts and inconsistencies between communicated intents and actual actions (Zeyang Liu, et al., 2021). For instance, if two agents send contradictory instructions to each other, they may act based on conflicting assumptions, resulting in unintended behaviors.

This issue is illustrated in Equations 5, where actions $\pi_i'(s_i)$ are derived without messages m_i:

$$\pi_1'(s_i) = \underset{a \in \mathcal{A}_i}{\text{argmax}} LLM_i(a|s_i) \tag{5}$$

Dependencies between states s_i, messages m_i, and actions $\pi_i(s_i)$ can cause misalignment between communicated intentions and actual behaviors.

Such contradictions hinder effective coordination, slowing learning and degrading performance in cooperative tasks.

3.2 Unidirectional Message Transmission

To address the issues in bidirectional communication, we propose a unidirectional communication method, where only one agent generates messages. This approach eliminates conflicts and inconsistencies while retaining relevant information exchange. The process is illustrated in Figure 2.

Let $\mathcal{N} = \{1, 2, \ldots, N\}$ denote the set of N agents, and $\mathcal{A}_i$ be the action space of agent i. The process is as follows:

1. Each agent i computes its message m_i using its state s_i:

$$m_i = LLM_i(\cdot|s_i) \tag{6}$$

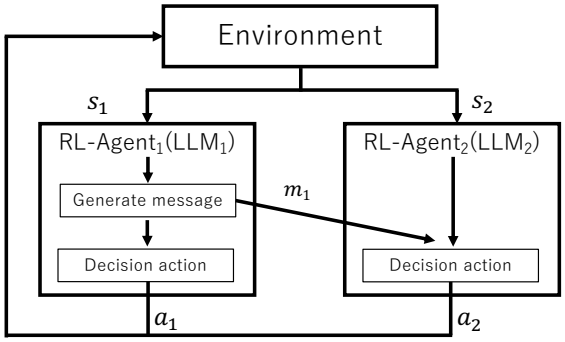

Figure 2: Overview of unidirectional communication

2. Each agent receives messages from all other agents and combines them with its state to compute the probabilities of actions:

$$\pi_i(s_i, \mathbf{m}_{-i}) = \underset{a \in \mathcal{A}_i}{\text{argmax}} LLM_i(a|s_i, \mathbf{m}_{-i}) \tag{7}$$

Here, $\mathbf{m}_{-i}$ represents the set of messages from all other agents:

$$\mathbf{m}_{-i} = \{m_j : j \in \mathcal{N}, j \neq i\}$$

The following pseudocode demonstrates the computation of actions for all agents:

Algorithm 1 Calculate Actions for N Agents

Require: $\{s_1, s_2, \ldots, s_N\} \in$ States
Ensure: $\{\pi_1(s_1), \pi_2(s_2), \ldots, \pi_N(s_N)\} \in \mathcal{A}_1 \times \mathcal{A}_2 \times \cdots \times \mathcal{A}_N$

Initialize $\mathbf{m} \leftarrow [\,]$ {Message list for all agents}
for all $i \in \mathcal{N}$ **do**
  $m_i \leftarrow LLM_i(\cdot|s_i)$
  Append m_i to $\mathbf{m}$
end for
for all $i \in \mathcal{N}$ **do**
  $\mathbf{m}_{-i} \leftarrow \mathbf{m} \setminus \{m_i\}$
  $\pi_i(s_i, \mathbf{m}_{-i}) \leftarrow \underset{a \in \mathcal{A}_i}{\text{argmax}} LLM_i(a|s_i, \mathbf{m}_{-i})$
end for
return $\{\pi_1(s_1), \pi_2(s_2), \ldots, \pi_N(s_N)\}$

Example: Let us consider a two agent case. Using their state s_1, Agent 1 generates a message m_1 for Agent 2. Both agents then compute their actions based on their states and, in the case of Agent 2, the received message m_1:

$$\pi_1(s_1) = \underset{a \in \mathcal{A}_1}{\text{argmax}} LLM_1(a|s_1) \tag{8}$$

$$\pi_2(s_2) = \underset{a \in \mathcal{A}_2}{\text{argmax}} LLM_2(a|s_2, m_1) \tag{9}$$

3.3 Alternating Proposals for Multiple Agents

While unidirectional communication resolves conflicts, it lacks the collaborative dynamics of bidirectional communication. To address this, we propose an alternating proposals method where N agents iteratively exchange messages for n rounds. This iterative exchange allows agents to reach a consensus by sharing information and strategies. The process, depicted in Figure 3, enhances coordination and reduces inconsistencies.

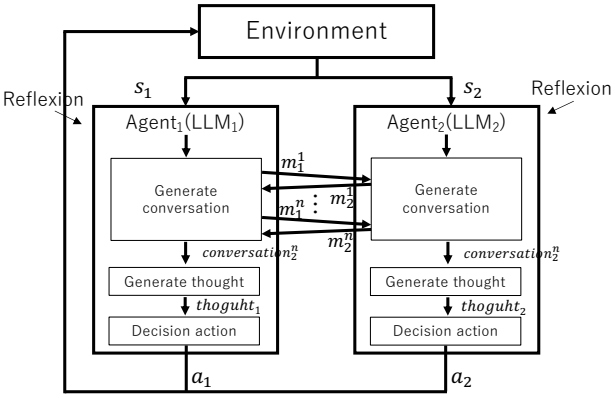

Figure 3: Overview of alternating proposals for multiple agents

Messages exchanged during each round are recorded in a global conversation log as follows:

$$conversation^k = (m_1^1, m_2^1, \ldots, m_N^1, \ldots, m_1^k, m_2^k, \ldots, m_N^k), \tag{10}$$

where m_i^k is the message generated by agent i during the k-th round.

After the n-th round, each agent uses the complete conversation history, $conversation^n$, along with their respective states s_i, to perform CoT reasoning and determine their actions. The action for agent i is computed as:

$$\pi_i(s_i) = \underset{a \in \mathcal{A}_i}{\text{argmax}} LLM_i(a|s_i, conversation^n), \tag{11}$$

where $\mathcal{A}_i$ represents the action space of agent i.

The pseudocode for this approach is shown in Figure 4.

This approach generalizes the alternating proposals framework to any number of agents, enabling collaborative behavior in environments requiring complex coordination. Each agent contributes to and benefits from the collective conversation, ensuring consistent and coordinated action decisions across the system.

Algorithm 2 Calculate $\pi_1, \pi_2, \ldots, \pi_N$

Require: $\{s_1, s_2, \ldots, s_N\} \in String^N$
Ensure: $\{\pi_1(s_1), \pi_2(s_2), \ldots, \pi_N(s_N)\} \in \mathcal{A}_1 \times \mathcal{A}_2 \times \ldots \times \mathcal{A}_N$

$conversation \leftarrow []$
for $k = 1$ to n **do**
  for $i = 1$ to N **do**
    $m_i^k \leftarrow LLM_i(\cdot|s_i, conversation)$
    $conversation \leftarrow conversation + m_i^k$
  end for
end for
for $i = 1$ to N **do**
  $a_i \leftarrow \underset{a \in \mathcal{A}_i}{\text{argmax}} LLM_i(a|s_i, conversation)$
end for
return $\{a_1, a_2, \ldots, a_N\}$

Figure 4: Pseudocode for alternating proposals with N agents

4 Evaluation

4.1 BabyAI and BabyAI-Text Environments

BabyAI (Maxime Chevalier-Boisvert, et al., 2019) is a single-agent environment designed for language-based learning, constructed by Chevalier-Boisvert et al. As illustrated in Figure 5, the environment comprises a grid populated with agents (triangles) and various objects (e.g., circles, boxes, and keys). The agent performs actions repeatedly based on limited observational information to accomplish diverse tasks.

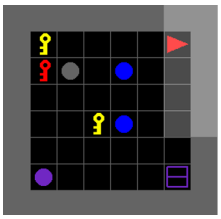

Figure 5: Example of BabyAI environment (Maxime Chevalier-Boisvert, et al., 2019)

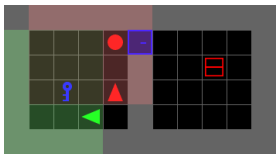

Figure 6: Example of BabyAI multi-agent environment

In the BabyAI environment, agents can observe specific information about objects within their immediate vicinity, including their position and direction. Based on this observational information, agents choose actions from the following: "go forward", "turn right", "turn left", "pick up" (pick

up an object in front), "drop" (place a previously picked object in front), "toggle" (open/close a box or door in front).

These actions enable the agent to navigate the field, transport objects, and move between rooms by opening/closing doors. Combining these actions allows the agent to accomplish tasks such as:

- GoToObj: Move next to a specific object
- PutNext: Place a specific object next to another specific object
- UnlockPickup: Unlock a door and pick up an object inside
- Pickup (BlockedUnlockPickup): Similar to UnlockPickup, but with additional obstacles in front of the door

Besides these predefined tasks, varying the grid size, object placement, and completion conditions enables the implementation of various tasks.

Because this environment facilitates the textual description of states, BabyAI-Text was developed by adding functionality to convert observations from real-number vectors to text, as used in experiments with GLAM and FAMA. Following these studies, we also add a functionality to convert observations into text.

Additionally, experiments were conducted in a multi-agent environment expanded from the original setting, similar to previous studies (Oliver Slumbers, et al., 2024). Each agent operates within its observable range, and multiple agents cooperate to achieve a single goal. A visualization of the multi-agent environment is shown in Figure 6.

Figure 6 shows the task “Pickup", which is accomplished by having one of the agents pick up the red box in the room on the right. To reach the box, the red ball in front of the blue door must be picked up and removed, and the door must be opened with the blue key.

4.2 Experiment Settings

In the experiment, for each condition in Table 1, the task in "Pickup","PutNext" is performed until learning converges. For each method using LLM, Llama3.1 (70B) (AI@Meta, 2024)is used as the language model. As a conventional reinforcement learning method that does not use LLM, we use an algorithm called MAPPO(Chao Yu, et al., 2022) for training and comparison.

Table 1: Settings for each experiment

ID	Method	Learning algorithm	Other setting
(1)	No Communication	Reflexion	-
(2)	Bidirectional	Reflexion	-
(3)	Unidirectional	Reflexion	-
(4)	Alternating Proposals	Reflexion	1 round
(5)	Alternating Proposals	Reflexion	2 rounds
(6)	Alternating Proposals	Reflexion	3 rounds
(7)	Alternating Proposals	Reflexion	4 rounds
(8)	-	MAPPO	No pseudo reward
(9)	-	MAPPO	Pseudo reward

4.3 Experiment Results

Table 2 shows the averages over the 100-episode runs for the number of steps accomplished in the task with the learned model in each experiment.

Table 2: Average number of steps achieved in each experiment

Task	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)
Pickup	45.6	40.2	42.7	40.1	**36.8**	37.9	44.4	47.6	42.5
PutNext	98.1	90.2	90.0	84.4	**72.5**	76.5	93.4	99.2	76.3

From Table 2, we can see that the two-round alternating proposal communication in (5) accomplished the two tasks in the shortest number of steps.

For (2) and (3), both tasks are accomplished with fewer steps than (1). However, the change from bidirectional to unidirectional does not improve the performance. In the alternately proposed communication in (5), we can see that the average number of steps required for one episode is smaller than the others for all tasks.

Figures 7 and 8 show the average number of steps taken by the model at each learning stage to accomplish the task when the model was trained with the PickUp task for (2), (3), (5), (8), and (9) in Table 1.

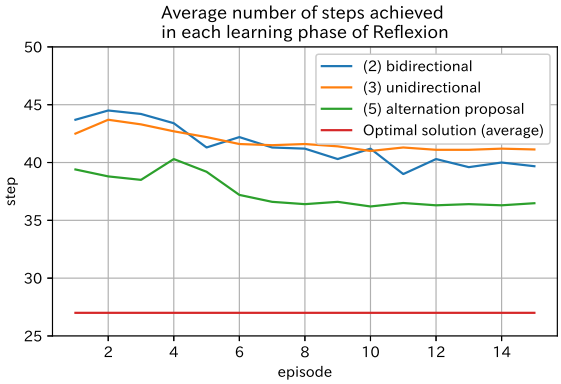

Figure 7: Comparison of each communication method in learning

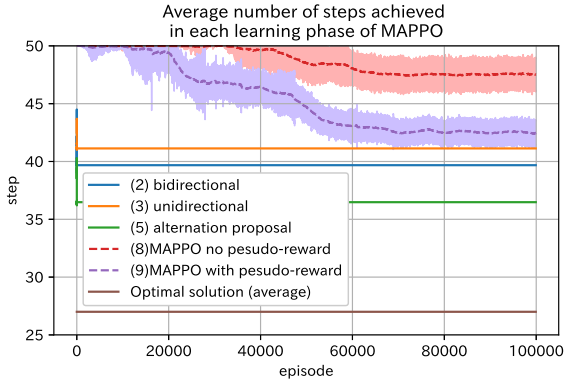

Figure 8: Comparison of LLM-based methods with MAPPO in learning

4.4 Analysis

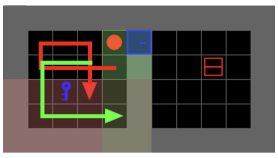

Figure 9: Example of search wasted by both agents due to contradiction between message and action

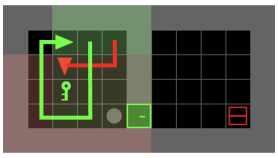

Figure 10: Example of search wasted by one agent due to lack of information from green agent

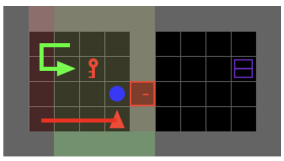

Figure 11: Example where conversation and agreement led both agents to the key and the ball

Below are examples of messages and actions in the Pickup task for each method. The following messages were exchanged in the bidirectional communication in (2).

Examples of messages in (2)

agent1: ... I'm planning to head to (2, 3) to pick up the blue key next ...
agent2: ... I'm currently at (3, 2). I'm planning to pick up the blue key ...

Both agents send a message that they are going to retrieve the key themselves. The subsequent thoughts of the agents by CoT are as follows.

Output of CoT

agent1: I should find something that clues. After agent2 gets the key, I'll go to the door ...
agent2: ... I go forward to find something ...

Both agents decide that the other agent will take the key and choose to search the surrounding area, which is inconsistent with the message they sent. The behaviors after the message are shown in Figure 9. All the agents do not pick up the key that was in their sight, but move around and search the surrounding area.

On the other hand, in the unidirectional communication in (3), a message is sent from agent 1 to agent 2, asking agent 2 to look for other clues as agent 1 picks up the key. Since agent 2 did not give any feedback on the search results, the same instructions were repeatedly sent to agent 1. Therefore, agent 2 continued his search in vain. The behavior after this message is shown in Figure 10.

In the communication of the alternate proposal in (5), the following conversation took place.

Examples of conversation in (5)

agent1: ... there is the key at (3, 2), so I'll pick up the key. Please find something that clues.
agent2: I'm at (2, 1) and near the green key. I'll move to (3, 2) and pick up the key.
agent1: Sounds like a good plan, I'll head east and search for the box or any clues ...
agent2: If I pick up the key, I'll head to ...

Agent 1 sends a message to agent 2 asking him to pick up the key and search for other clues. Agent 2 then reports that there is a key nearby, and as a result of the alternation of suggestions, agent 2 picks up the key and agent 1 searches for it, indicating that the policy is well shared. The behavior after this message is shown in Figure 11.

Figure 7 shows that the performance of Reflexion is improved after about 15 episodes of learning. In the case of bidirectional communication in (2), the accuracy is not stable, going up and down. The following is an example of output from Reflexion, which seems to be the cause of the lower accuracy.

Example of Reflexion output in (2)

Trial1: ... I should go to the key as agent2 said, then turn right to get a better angle ...
Trial2: ... I should moved to position next to the ball, then pick up it to find the box ...

Trial 1 shows that he should have picked up the key himself, while Trial 2 shows the opposite, that he should have checked the ball and other objects. Due to the instability of the policy, Reflexion outputs the opposite, which reduces the accuracy.

Although unidirectional communication in (3) avoided unstable outputs compared to the bidirectional communication, the lack of sufficient information exchange resulted in repeated ambiguous

outputs in Reflexion, such as "we should have coordinated more," and the accuracy did not improved.

In the communication of the alternation proposal in (5), the accuracy increased and converged faster than the other methods. It is considered that the stable learning is due to the fact that the output is consistent and gradually becomes more specific, as a result of referring to a stable history with conversation.

A comparison with MAPPO is made based on Figure 8. The MAPPO method is not as good as the LLM-based method. This is due to the sparse reward in (8), where the reward is given only when the task is accomplished. Therefore, the accuracy was improved by adding a pseudo-reward as in (9). Thus, there is a possibility that MAPPO may outperform the proposed method, depending on the pseudo-rewards. On the other hand, it can be said that the proposed method can achieve high accuracy in a very short learning time without complicated reward design.

5 Limitations

In this study, verification with realistic tasks is insufficient. This is due to the lack of time and money to conduct experiments in various conditions. There are two main issues. The first is the need for humans to adjust the prompts and the mechanism for converting the state to text, which is as time-consuming as designing rewards. Second, the number of calls to the LLM is large even though the output of the LLM is slow. Shortening the prompts, decreasing the number of calls, and so on can save time and money.

6 Conclusion

We proposed a structured communication method using an alternating proposal scheme for multi-agent reinforcement learning with LLMs. Our experiments in the BabyAI multi-agent environment demonstrate that two-round alternating proposal communication significantly enhances coordination, improving both learning speed and accuracy compared to conventional reinforcement learning and communication strategies.

References

AI@Meta. 2024. Llama 3 model card. *https://github.com/meta-llama/llama3/blob/main/MODEL_CARD.md*.

Cao Yuji, et al. 2024. Survey on large language model-enhanced reinforcement learning: Concept, taxonomy, and methods. *IEEE Transactions on Neural Networks and Learning Systems*.

Chao Yu, et al. 2022. The surprising effectiveness of ppo in cooperative, multi-agent games. *Advances in Neural Information Processing Systems*, 35:24611–24624.

Jakob N. Foerster, et al. 2016. Learning to communicate with deep multi-agent reinforcement learning. *30th Conference on Neural Information Processing Systems*.

Jason Wei, et al. 2023. Chain-of-thought prompting elicits reasoning in large language models. *Proceedings of the 36th International Conference on Neural Information Processing Systems*.

Jiechuan Jiang, et al. 2020. Graph convolutional reinforcement learning for multi-agent cooperation. *International Conference on Learning Representations*.

John Schulman, et al. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.

Lianyu Hu, et al. 2024. Comma: Co-articulated multimodal learning. *Proceedings of the AAAI Conference on Artificial Intelligence*.

Maxime Chevalier-Boisvert, et al. 2019. Babyai: A platform to study the sample efficiency of grounded language learning. *In International Conference on Learning Representations, ICLR*.

Noah Shinn, et.al. 2023. Reflexion: Language agents with verbal reinforcement learning. *Proceedings of the 37th Conference on Neural Information Processing Systems*.

Oliver Slumbers, et al. 2024. Leveraging large language models for optimised coordination in textual multi-agent reinforcement learning. *arXiv:2304.07297v2, cs.AI*.

Ryan Lowe, et al. 2017. Multi-agent actor-critic for mixed cooperative-competitive environments. *31st Conference on Neural Information Processing Systems, NIPS 2017*.

Thomas Carta, et al. 2023. Grounding large language models in interactive environments with online reinforcement learning. *Proceedings of the 40th International Conference on Machine Learning*, pages 3676–3713.

Zeyang Liu, et al. 2021. Multi-agent intention sharing via leader-follower forest. *arXiv preprint arXiv:2112.01078*.