Rethinking the Bounds of LLM Reasoning:
Are Multi-Agent Discussions the Key?

Qineng Wang[1*]   **Zihao Wang**[2*]   **Ying Su**[2]   **Hanghang Tong**[3]   **Yangqiu Song**[2]

[1]Zhejiang University   [2]HKUST   [3]UIUC

wongqineng73@zju.edu.cn, htong@illinois.edu

{zwanggc,ysuay,yqsong}@cse.ust.hk

Abstract

Recent progress in LLMs discussion suggests that multi-agent discussion improves the reasoning abilities of LLMs. In this work, we reevaluate this claim through systematic experiments, where we propose a novel group discussion framework to enrich the set of discussion mechanisms. Interestingly, our results show that a single-agent LLM with strong prompts can achieve almost the same performance as the best existing discussion approach on a wide range of reasoning tasks and backbone LLMs. We observe that the multi-agent discussion performs better than a single agent only when there is no demonstration in the prompt. Further study reveals the common interaction mechanisms of LLMs during the discussion.

1 Introduction

Large Language Models (LLMs) demonstrate strong abilities in language understanding and generation (OpenAI, 2022, 2023; Touvron et al., 2023a; Zhang et al., 2022a; Chowdhery et al., 2022; Team et al., 2023). However, LLMs still fall short for reasoning tasks due to model deficiencies like hallucination and reasoning perspective mistakes (Xu et al., 2023a). To overcome these issues, numerous works have been proposed by simulating human reasoning. Inspired by *Society of Mind* (Minsky, 1988), multi-agent discussion frameworks such as Debate (Du et al., 2023), MAD (Liang et al., 2023), and ReConcile (Chen et al., 2023a) present a novel approach by involving multiple AI agents. Powered by LLMs, these agents autonomously engage in discussions on given topics, improving the reasoning abilities of LLMs by emulating the human discussion process. To further improve the performance, most of multi-agent discussion frameworks leverage task-specific examples, which are often termed as *demonstrations* (Min et al., 2022). This is

---

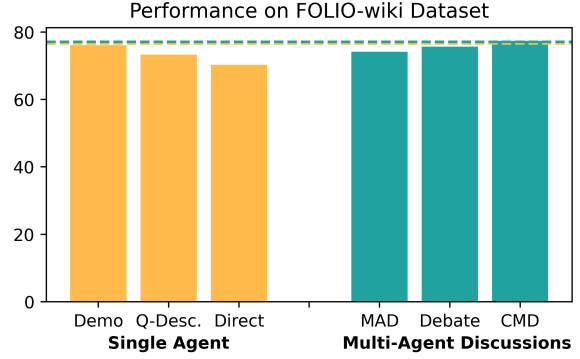*These authors contributed equally to this work.

Figure 1: Comparative performance of single-agent settings and multi-agent discussion frameworks on FOLIO-wiki dataset.

based on the insights that LLMs can learn from the context demonstrations (Brown et al., 2020). We note that *all these discussion frameworks claim that they outperform the conventional methods with a single agent*, such as Chain-of-Thought (Wei et al., 2022).

Figure 1 presents a comparison between single-agent settings and multi-agent discussion frameworks on FOLIO-wiki dataset (Zhang et al., 2023b; Han et al., 2022). In this figure, 'Demo' means that the tested single agent is provided with a demonstration case, 'Q-Desc.' indicates that the single agent is provided with detailed question description, while 'Direct' refers to the single agent without demonstrations. We observe that the single agent 'Demo' tends to reach a performance upper bound similar to that of all discussion frameworks, which will be further elaborated on later. This result suggests that the previous *claim* is NOT fully established.

Based on the observation from Figure 1, in this paper, we conduct systematic experiments to provide in-depth analysis. As a newly emergent topic, the number of available discussion frameworks still remains limited. To provide a more comprehensive and detailed comparison between the
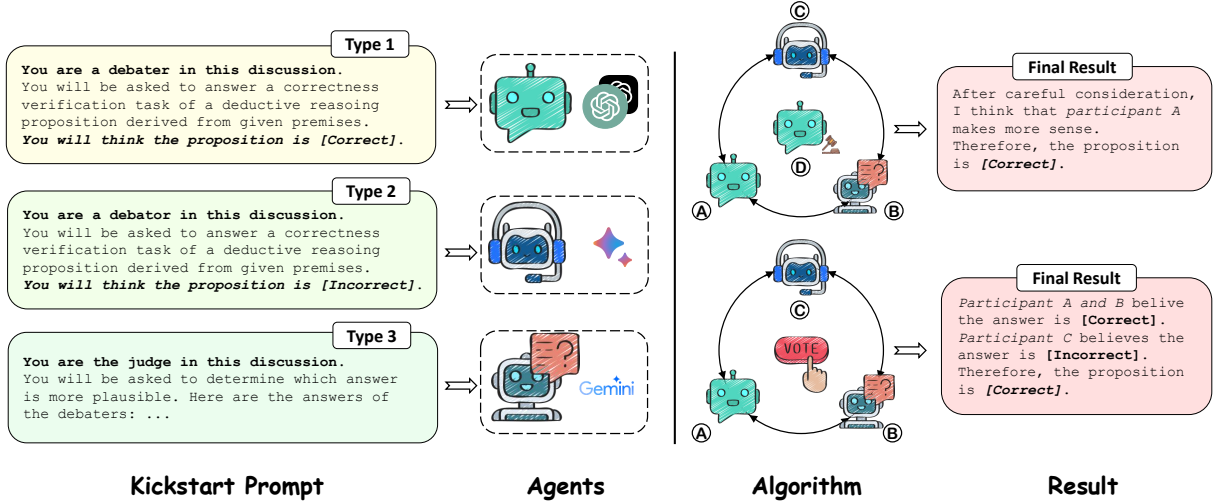
1

Figure 2: Our proposed design pipeline of multi-agent discussion frameworks. This pipeline operates by having agents starting with a kick-start prompt. Then, agents will start discussion by obeying the rules defined in the algorithm and come to a result in the end.

single-agent settings and discussion frameworks, as demonstrated in Figure 3, we propose a new discussion framework named CMD, which is inspired by human group discussion process. Our discussion pipeline considers multiple design aspects shown in Figure 2 (see Section 2.1 for more details). We further carry out a range of experiments over standard reasoning benchmarks (ECQA (Aggarwal et al., 2021), GSM8k (Cobbe et al., 2021), and FOLIO-wiki (Han et al., 2022; Zhang et al., 2023b)) using various configurations within different parts of this pipeline.

We find that multi-agent discussion does *not* necessarily enhance reasoning when the prompt provided to an agent is sufficiently robust, which aligns with the observation from Figure 1. Additionally, our experimental results reveal that multi-agent discussion frameworks outperform single-agent setups when no demonstrations are provided. We also find that agents powered by weaker LLMs like Bard (Anil et al., 2023) can improve its performance on reasoning with the assistance of the stronger LLMs like Gemini Pro (Team et al., 2023) during interaction.

In summary, our analysis provides a new way of understanding when to use multi-agent discussion on reasoning. Our contributions are listed as follows: (1) We propose a new multi-agent discussion framework CMD, which simulates human group discussion process; (2) We observe that single agents with a strong prompt match the performance of a multi-agent discussion using equally robust prompts; (3) We identify two common types of discussion errors: judge mistake and wrong an-

swer propagation; (4) We find that without demonstrations, multi-agent discussions surpass the single agent ; (5) In multi-LLMs multi-agent discussions, agents with stronger LLMs help improve performance of agents with weaker LLMs.

2 Preliminary

2.1 What is Multi-Agent Discussion?

Multi-agent discussion refers to an interactive setup where multiple agents, each powered by an LLM, engage in an autonomous dialogue. Each agent is given a prompt that outlines the necessary background knowledge and guides its behavior throughout the discussion. Once the topic is given, these agents can carry on the conversation independently.

Figure 2 illustrates a discussion design pipeline, which is structured into four main elements: the Kick-start prompt, agents, algorithm, and result. The Kick-start prompt supplies essential details such as background knowledge and the topic for discussion. This information is then fed to an agent, which operates based on various LLMs. Next, the algorithm lays out the specific rules of discussion, including the number of rounds, decision-making processes (whether by a judge or through voting), and the structure of the discussion (such as hierarchical). Lastly, 'Result' describes how the final decisions are rendered and presented. We further provide a view from symmetry for prompt-related mechanism (including discussion) in Appendix A.
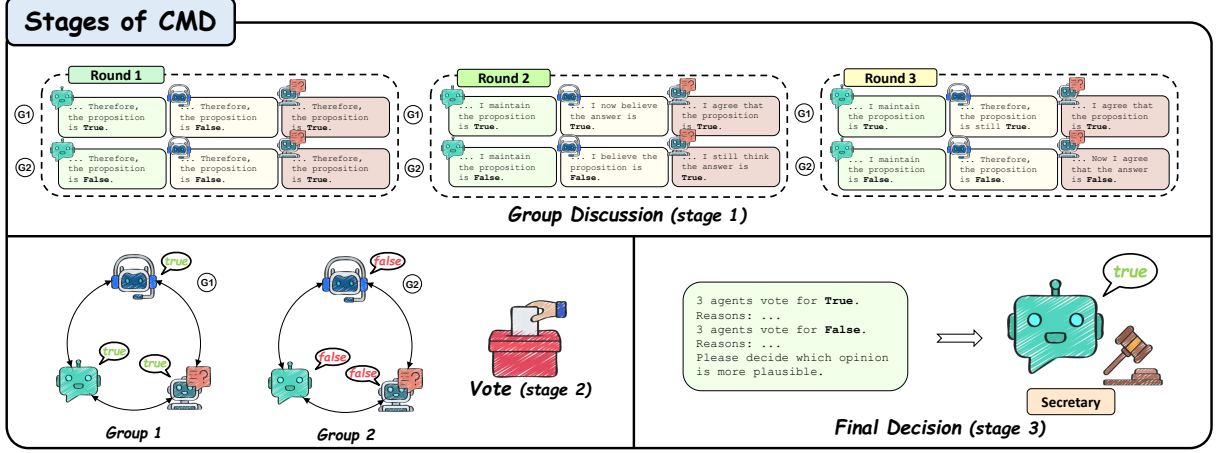
2

Figure 3: Overview of the Conquer-and-Merge Discussion (CMD) Framework.

2.2 Existing Discussion Frameworks

Previous works on multi-agent discussion frameworks have already covered several common discussion paradigms. In this paper, we consider the following discussion frameworks for our experiments: Debate (Du et al., 2023), MAD (Liang et al., 2023) and ReConcile (Chen et al., 2023a).

Debate replicates a simple turn-based discussion among agents. Initially, all agents are prompted to address the assigned task, and their respective responses are then incorporated into each agent's input for the subsequent round. Typically, this process consists of three rounds of discussion.

MAD introduces divergent thinking by involving two participants to discuss a task from opposing perspectives. A judge then steps in to evaluate which viewpoint seems more plausible or if further discussion is needed. The discussion is repeated until a consensus is reached or the judge favors a particular solution.

ReConcile implements a round-table discussion with the agents powered by multiple LLMs. It reaches a decision through a weighted voting system. In this process, agents provide a confidence level for their answers, and agents in ReConcile utilize these confidence levels as weights to cast votes and arrive at a final decision.

Despite these advancements, the number of multi-agent frameworks is still limited, and none has yet to consider one of the most prevalent forms of discussion: group discussion.

3 CMD: Conquer-and-Merge Discussion

To provide a thorough comparison in following experiments, we identify and rectify the gap in pre-

vious research, which has not suggested a group-discussion-based framework. To address this, we introduce a novel framework for multi-agent group discussion, referred to as CMD. In CMD, suppose there are n agents $\mathcal{A} = \{A_i\}_{i=1}^n$ discussing a question $\mathcal{Q}$, and agent A_i is powered by an LLM $\mathcal{L}_i$. We use an array H to store the history responses. The agents will discuss for $\mathcal{R}$ rounds. During each turn r, every agent A_i generates a response $(v_i, e_i) = \text{CMD}(H|\mathcal{Q}, r)$, where v_i is the viewpoint and e_i denotes the explanation. Detailed description can be seen in Appendix B.

3.1 Message-Passing Algorithm

For the synchronization of agent communications, we propose a message-passing algorithm. Previous works on multi-agent discussion frameworks focus exclusively on specific scenarios without designing a universal algorithm to synchronize agent messages across various discussions. Moreover, discussion forms vary in architecture and agents can be powered by different LLMs, where each LLM usually possesses different calling protocols. Therefore, we design a message-passing algorithm using a multi-threaded way to overcome these issues. The algorithm establishes a receiving map M for every agent to store the messages they should get in the next turn. For each message m_j from Agent A_i, the algorithm first confirms the receivers $\mathcal{A}_r$, then add m_j into the receiving map M by every agent $A_k \in \mathcal{A}_r$. When the next turn begins, the algorithm will automatically push stored messages from M to corresponding agents. Please refer to Appendix B.4 for more detailed pseudo code.

3.2 Three Stages of `CMD`

As illustrated in Figure 3, `CMD` consists of three stages: group discussion, voting and the final decision stage. Typically, the final decision stage is reserved for instances of a tie and is otherwise unnecessary. Below is a breakdown of each stage involved in the `CMD` framework, and detailed introduction can be seen in Appendix B.3.

Stage 1: Group Discussion. During this stage, agents $\mathcal{A}$ are divided into groups $\mathcal{G} = \{G_k\}_{k=1}^{t}$ with an equal size. All agents are asked to solve task $\mathcal{Q}$ through discussion. For each agent $A_i \in G_k$, all answers and explanations from $A_i \in G_k$ in the last round are accessible where $A_j \in G_k$ and $j \neq i$. In contrast, they can only see the answers without explanations from agents in other groups. After $\mathcal{R}$ rounds of discussions, `CMD` moves to the voting stage.

Stage 2: Voting. When discussion reaches the maximum number of rounds, all agents $\mathcal{A}$ start voting. Each vote of agent in this discussion is treated equally, therefore, the result is determined by the majority decision. In the event of a tie, `CMD` transitions to stage 3. Otherwise, the discussion process concludes formally.

Stage 3: Final Decision. If a tie occurs, we introduce an extra agent S in the role of the secretary to make the final decision. Each proposed answer is accompanied by an explanation selected from agents with the same opinion and provided to the secretary for the final determination. To sum up, the final decision is made by either $\text{Vote}(\mathcal{A})$ or $S(V, O|\mathcal{Q})$ where $V = \{v_i\}^n$ and $O = \{o_i\}^n$, representing a set of viewpoints and a set of explanations respectively.

4 Experimental Setups

In the experiments, we contrast single-agent setup with four established multi-agent discussion frameworks: Debate, MAD, ReConcile, and `CMD` under various prompt conditions. These methods are introduced in previous Sections 2.2 and 3.

4.1 Implementation Details and Metrics

Our experiments are primarily implemented with three advanced LLMs, including ChatGPT-3.5 (OpenAI, 2022), Gemini Pro (Team et al., 2023) and Bard (Anil et al., 2023). In particular, we employ the gpt-35-turbo (0613) instance hosted on Azure OpenAI[1] for ChatGPT-3.5, while the chat-

bison-001 model represents Bard via PaLM2 architecture. Gemini Pro and Bard interfaces operate through Google MakerSuite API[2]. A uniform dialogue temperature of 0.25 is configured across LLMs on `CMD` to ensure consistency. For all multi-agent discussion frameworks, we set the maximum discussion round number to 3. Our evaluations use accuracy to measure performance across all tasks.

4.2 Downstream Tasks

The frameworks are implemented on a suite of reasoning tasks, including a commonsense reasoning task and two deductive reasoning tasks: (1) ECQA (Aggarwal et al., 2021): A QA dataset centered on commonsense knowledge, (2) GSM8K (Cobbe et al., 2021): A benchmark consists of math word problems, (3) FOLIO-wiki (Zhang et al., 2023b): A dataset adapted from FOLIO (Han et al., 2022) for both symbolic and natural language deductive reasoning. In alignment with constraints imposed by computational resources and following precedents set by earlier research (Du et al., 2023; Chen et al., 2023a; Liang et al., 2023), a subset of 100 instances from the test sets of both ECQA and GSM8K are selectively sampled. For an in-depth analysis, we choose to conduct a comprehensive experiments of all 460 cases within the curated version of the FOLIO-wiki dataset, which removes the flawed cases to ensure the result authenticity.

5 Experiments on Single LLM

In this section, we conduct our experiments using a single LLM, ChatGPT-3.5. To gain the initial insights, we provide an in-depth analysis of the FOLIO-wiki dataset, examining both single-agent settings and multi-agent discussions as detailed in Section 5.1. Afterward, we extend our experiments to two other datasets GSM8K and ECQA in Section 5.2. We also investigate common mistakes made by discussions through a case study in Section 5.3. Finally, we summarize our findings in Section 5.4.

5.1 Analysis of FOLIO-wiki Dataset

We begin by examining if multi-agent discussions are more effective than an agent using the strongest prompt on FOLIO-wiki dataset. Drawing from previous research (Wei et al., 2022; Ling et al., 2023) on crafting prompts for reasoning tasks, we divide the prompt into three parts: a detailed question description, which provides an in-depth background

Prompt Components			Multi-Agent Discussion (%)				Single Agent (%)
Q-Desc.	A-Desc.	Demo.	MAD (3)	Debate (3)	Debate (6)	CMD (6)	
✗	✗	✗	64.13	70.00	69.13	73.26	70.22
✓	✗	✗	74.13	75.65	76.30	74.13	73.26
✗	✓	✗	68.91	71.96	71.74	73.89	71.30
✓	✓	✗	71.96	70.22	70.00	71.09	73.91
✓	✓	✓	74.13	75.65	74.78	77.39	76.09

Table 1: Comparative performance of single-agent settings and multi-agent discussions on FOLIO-wiki using ChatGPT-3.5. Abbreviations are: detailed question descriptions (Q-Desc.), and answer format descriptions (A-Desc.), demonstrations (Demo.). Only the question itself is used as input when prompt components are disabled. The number next to the framework represents the number of agents.

of the task; an answer format description, which instructs how an agent should reply; and a task-specific demonstration, which shows an example of a question and answer pair. For this task, we meticulously craft a demonstration for input prompt. We start by labeling each premise. After that, we quote all the premises and relate them to every step in the reasoning process by using these labels, until the final step is reached. A labeled example is similar to the case in Table 5. We then test the performance of various combinations of these components for both single agents and discussions.

Single Agent. We conduct an evaluation of different prompt components with a single agent, and the results are outlined in Table 1. This table shows that for both single-agent settings and multi-agent discussions, the inclusion of a detailed question description or a task-specific demonstration enhances reasoning abilities on the FOLIO-wiki dataset. The detailed question description is helpful because the possible answers to judge the correctness of a given proposition—true, false, or unknown—require clarity. Without such clarity, agents often struggle to differentiate between what is 'false' and what is 'unknown'. Most notably, the addition of a demonstration contributes significantly to improved performance, highlighting its value as the most impactful component, in line with what prior studies have suggested (Min et al., 2022).

Multi-Agent Discussions. We assess the same prompt components within multi-agent discussion setups, and Table 1 reveals three key insights: (1) Demonstrations and detailed question description enhance multi-agent discussions. (2) Despite similar overall performance, most multi-agent discussions do not surpass the single agent when a demonstration is introduced. (3) CMD performs better than both single-agent setups and other multi-agent dis-

cussion frameworks on the FOLIO-wiki dataset. Below is further analysis of our findings.

A Strong Single Agent is Comparable to Discussion Frameworks. Analyzing experiments with single-agent settings and multi-agent discussions on the FOLIO-wiki dataset, we find that task-specific demonstrations significantly enhance a single agent's performance. Additionally, we establish that a well-supported agent can perform on par with discussion frameworks. Our analysis indicates that prompt engineering can boost reasoning performance in large language models, with demonstrations in both single-agent and multi-agent discussions pushing towards the upper bound of performance.

5.2 Evaluation on All Tasks

In Section 5.1, we discover that demonstrations play a pivotal role in enhancing performance on FOLIO-wiki dataset. With this insight in mind, we simplify our evaluation to two prompt scenarios: with (referred as direct) and without demonstrations (referred as demo). The results presented in Table 2 cover all tested reasoning tasks. The findings show: (1) With demonstrations, discussion frameworks and single-agent settings have comparable performance on average. This is consistent with our earlier observations. (2) Without demonstrations, CMD tend to surpass single-agent settings, both on average and in most individual tasks.

When Does Discussion Work Better? Most Multi-agent discussion frameworks, especially CMD, achieve better performance compared with single-agent settings when neither is supported by demonstrations. We believe this is because, during discussions, the input from other agents can introduce new perspectives, leading to a more thorough reasoning process. Therefore, this collaborative

5

Method	ECQA		GSM8K		FOLIO-wiki		Average	
	Direct	**Demo**	**Direct**	**Demo**	**Direct**	**Demo**	**Direct**	**Demo**
Single Agent	63.00	67.00	69.00	83.00	70.22	76.09	67.41	75.63
MAD (3 Agents)	55.00	58.00	74.00	78.00	61.25	74.13	63.42	70.04
Debate (3 Agents)	67.00	65.00	78.00	81.00	70.00	75.65	71.67	73.88
Debate (6 Agents)	65.00	64.00	74.00	78.00	69.13	74.78	69.38	72.26
`CMD` (6 Agents)	64.00	63.00	75.00	83.00	73.26	77.39	70.75	74.46

Table 2: Results for all tasks, with and without demonstration settings included. Using ChatGPT-3.5.

advantage makes multi-agent discussions a more effective option in scenarios lacking specific expert knowledge or detailed examples.

Why Does Discussion Frameworks Perform Differently on Tested Tasks? Table 2 indicates that MAD is the least effective among the frameworks tested. We suspect that this is because MAD incorporates a divergent thinking way, which asks agents to disagree with each other. This can sometimes hinder reasoning by introducing irrelevant information that complicates decision-making. We explore this further with an error analysis in Section 5.3. Additionally, we observed unexpected behaviors from other discussion frameworks under specific conditions or tasks. For example, Debate and `CMD` perform worse on ECQA dataset when demonstrations are introduced. We hypothesize that because ECQA demands more commonsense knowledge than purely analytical reasoning, the single-source interaction stemming from a single LLM might cause agents to overthink. Instead of clarifying misunderstandings, the increased dialogue may introduce complexity without addressing the underlying knowledge gaps. In contrast to ECQA, datasets like GSM8K and FOLIO-wiki place a greater emphasis on deductive reasoning abilities. In these cases, the discussion process between agents can be beneficial as it allows them to identify and address flaws in each other's reasoning through interaction.

5.3 Two Discussion Error Types: A Case Study

Our experiments show that multi-agent discussions can sometimes reach incorrect conclusions on questions that a single agent answers correctly. Figure 4 presents an example from the FOLIO-wiki dataset illustrating this point: a single agent provides the correct answer, but multi-agent discussions lead to an erroneous result. We identify two
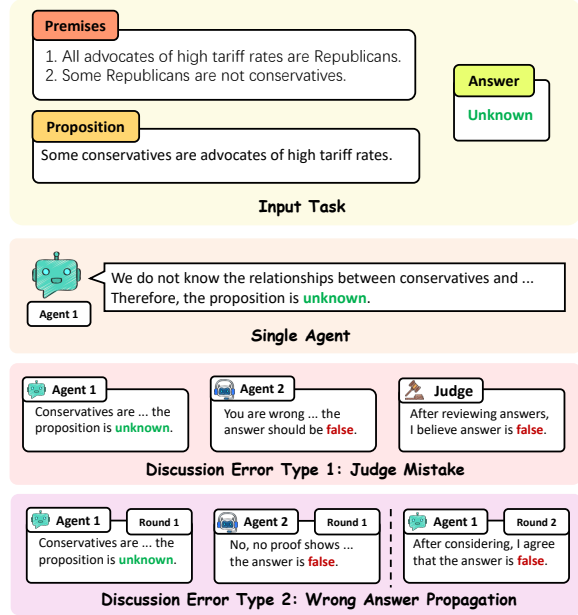

Figure 4: Two common types of errors that may occur in multi-agent discussions are judge mistake and wrong answer propagation. These issues can lead to circumstances where a multi-agent discussion reaches an incorrect conclusion, even if single agent can arrive at the correct one.

unique types of errors in multi-agent discussions: (1) Judge Mistake: This occurs in situations where an agent serves as a judge to decide on the final answer. If there are varying responses among agents, the judge might select the incorrect option as the final verdict. Frameworks like MAD and `CMD` are susceptible to this error, particularly when decisions are made during a tie. (2) Wrong Answer Propagation: This type of error happens when an agent, influenced by the input from others, deviates from its initial correct answer and adopts an incorrect consensus, spreading the mistake further in the discussion. This is the most common mistake the multi-agent discussion can make, even when most of their initial answers are correct.

5.4 Summary

In this section, we evaluate various prompt component combinations for both single-agent settings and multi-agent discussions. Our findings suggest that multi-agent discussions are on par with a single agent when both have access to demonstrations. However, in the absence of demonstrations, multi-agent discussions generally outperform a single agent, making them a better option in scenarios where expert knowledge or detailed examples are insufficient. We also highlight unusual outcomes and present a case study to identify two frequent errors in multi-agent discussions: Judge Mistake and Wrong Answer Propagation.

6 Experiments on Multiple LLMs

In this section, we expand our experimental scope from a single LLM to multiple LLMs for both single-agent settings and multi-agent discussion frameworks, which allows us to test the validity of our previous findings in multi-LLM scenarios. Specifically, we assess the performance of agents powered by three advanced LLMs: ChatGPT-3.5, Gemini Pro, and Bard. In Section 6.1, we compare the performance of three single-agent configurations, each using a different LLM, against multi-agent discussions leveraging all three LLMs. Our round-level analysis in Section 6.2 yields another insight: an agent powered by stronger LLM can enhance the performance of an agent powered by a less capable LLM.

6.1 Validate Findings on Multiple LLMs Scenarios

In this section, we evaluate the performance of three single-agent settings, each supported by a different LLM, and two multi-agent discussion frameworks, ReConcile and CMD, across all tasks. We utilize two types of prompt settings for each task: one with demonstrations and one without. For the CMD framework, we organize six agents into two groups of three, with each agent in a group powered by one of the LLMs: ChatGPT-3.5, Gemini Pro, or Bard. The results presented in Table 3 support our previous findings from Section 5 with some slight modifications.

A Strong Single Agent is Comparable to Discussion Frameworks. Based on experimental results from Table 3, we can find that discussion frameworks perform comparably to a single agent powered by Gemini Pro when both are provided

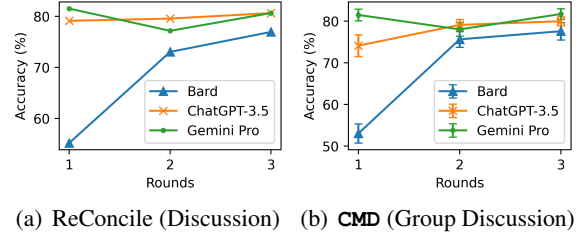(a) ReConcile (Discussion) (b) CMD (Group Discussion)

Figure 5: Round-level performance of each LLM in multi-agent discussions on FOLIO-wiki dataset.

with demonstrations. This consolidates our earlier conclusion that a 'strong' single agent—supported by both a well-designed prompt and a SOTA LLM—can rival the performance of a multi-agent discussion framework.

Discussion Frameworks Outperform Single Agents with No Demonstration. Table 3 reveals that, in multi-LLM scenarios, multi-agent discussions outperform single agents when demonstrations are not provided. This outcome aligns with our previous observations in single LLM settings. Furthermore, CMD and ReConcile demonstrate similar performance when they both have no access to demonstrations and they are both powered by same LLMs. This indicates that our findings are consistent on different multi-LLM multi-agent discussions frameworks.

6.2 Enhancing Agents in Weaker LLMs with Support from Stronger LLMs

As shown in Table 3, single agents using Bard show the least effectiveness in reasoning tasks such as those in the FOLIO-wiki dataset. However, multi-LLM multi-agent frameworks remain competitive. To understand how a less advanced LLM like Bard performs during multi-agent discussions, we further study the round-by-round performance of each LLM engaged in the discussions. Figure 5 demonstrates that agents with less capable LLMs like Bard and ChatGPT-3.5 gradually enhance their performance over consecutive rounds with the support of the more robust LLM, Gemini Pro. We infer that throughout the discussion, Gemini Pro assists in bridging the gaps in knowledge and reasoning for the less advanced LLMs, guiding towards a stronger line of reasoning. Notably, although there is a slight drop in the performance of Gemini Pro during the second round, it demonstrates resilience and recovers swiftly, largely maintaining its superior performance. Again, this finding is applicable to different multi-LLM discussions, specifically for

Method Category	LLM	ECQA		GSM8K		FOLIO-wiki		Average	
		Direct	Demo	Direct	Demo	Direct	Demo	Direct	Demo
Single Agent	Bard	66.00	65.00	47.00	54.00	70.00	71.96	61.00	63.65
	Gemini Pro	74.00	75.00	75.00	81.00	74.13	79.78	74.38	78.59
	ChatGPT-3.5	63.00	67.00	69.00	83.00	70.22	76.09	67.41	75.63
Discussion	ReConcile (Bard, Gemini, ChatGPT)	70.00	71.00	78.00	83.00	80.34	81.09	76.11	78.36
Group Discussion	**CMD** (Bard, Gemini, ChatGPT)	73.00	72.00	78.00	82.00	79.78	81.96	76.93	78.66

Table 3: Results from single-agent and CMD across multiple LLMs on all tasks evaluated with two types of prompts: with demonstrations and without.

both ReConcile and CMD.

7 Related Work

7.1 Prompting LLM for Reasoning

Recent researches have experienced great progresses in building powerful LLMs (Brown et al., 2020; OpenAI, 2022, 2023) or exploring the strategy of adopting LLMs over many downstream tasks via prompt enigineering.

By training with different knowledge textual sources and parameter size, various LLMs equipped with different reasoning capabilities are constructed, such as OPT (Zhang et al., 2022a), LLaMA (Touvron et al., 2023a,b), BLOOM (Scao et al., 2022), and PaLM (Chowdhery et al., 2022; Anil et al., 2023). Recently, Gemini Pro (Team et al., 2023) extends the capabilities of LLMs to the field of multi-modality.

Numerous advancements have been made in the field of improving reasoning abilities of LLMs with prompt engineering. Chain of Thought (CoT) (Wei et al., 2022; Kojima et al., 2022) is a linear problem-solving approach where each step builds upon the previous one. Fu et al. (2022) propose to apply CoT to multi-step reasoning tasks. To automate the CoT, Auto-CoT (Zhang et al., 2022b) constructs demonstrations by sampling diverse questions and generating reasoning chains. Active-Prompt (Diao et al., 2023) aims to select the most uncertain questions for task-specific annotations. Other prompt strategies designed to enhance reasoning in LLMs include the PS Prompt (Wang et al., 2023), which breaks tasks into subtasks, ToT (Yao et al., 2023a) which expands on the reasoning process by considering multiple paths of reasoning and self-evaluating choices, the effective GoT (Yao et al., 2023b), which frames thoughts as graphs, Natural Program (Ling et al., 2023) which helps to improve the deductive reasoning tasks, re-reading prompt (Xu et al., 2023b) which revisits the ques-

tion information embedded within input prompts.

7.2 Multi-agent Discussion for Reasoning with LLMs

Multi-agent discussion utilizes multiple LLMs as agents to collectively discuss and reason given problems in an interactive way. Abundant researches have explored how to improve the reasoning ability of single LLM, while multi-agent discussion among LLMs is still under exploration.

The Multi-Agent Debate framework, introduced by (Du et al., 2023), establishes a mechanism for symmetric discussions among agents. During the same period, the MAD (Multi-Agent Debate) framework (Liang et al., 2023) introduces an asymmetric mechanism design. It assigns different roles (debater and judge) asymmetrically. Other similar works include (Chan et al., 2023). Also, the ReConcile framework (Chen et al., 2023a) exemplifies an asymmetric discussion mechanism by involving different LLMs and using a weighted voting mechanism. To understand discussion more deeply, Zhang et al. (2023a) aim to explain such collaboration mechanism in a social psychology view.

Unlike these works, we aim to explore the potential effects of prompting contents over the discussion process by our defined multi-agent group discussion framework CMD.

8 Conclusion

In this paper, we re-examine the claim that multi-agent discussions are superior to a single agent in reasoning tasks by conducting systematic experiments. We introduce a novel framework CMD for a comprehensive and fair assessment. By conducting experiments over standard benchmarks, we find that (1) A single agent with a strong prompt and powered by a strong LLM achieves comparable performance with multi-LLM multi-agent discussions; (2) In the absence of demonstrations, multi-agent discussion frameworks outperform single agents on

most tasks; (3) When multiple LLMs are involved in multi-agent discussions, agents with stronger LLMs can enhance the performance of agents with weaker LLMs as discussion progresses.

9 Ethical Considerations

Our study employs publicly available datasets and LLMs accesses via official APIs, ensuring responsible and ethical use. Specifically, our ethical considerations can be summarized as follows:

Public Datasets. Datasets we use are designed for academic research. No personal data has been processed.

Licensed API Usage. Our application of LLMs complies with the API usage policies, maintaining fair use standards and respecting intellectual property.

Transparency. We provide detailed experimentation methods to allow for result reproduction and encourage transparent scientific practices.

10 Limitations

Our research offers comprehensive experiments to study the performance of a strong single agent and multi-agent discussions. However, several aspects highlighted below can be further refined and explored in future work.

Enhancing Agent Complexity. Currently, all discussion frameworks including `CMD` considers an LLM session as an AI Agent. This perspective simplifies the the concept of LLM-based AI Agent defined in the literature (Weng, 2023). By integrating more sophisticated techniques such as Tree-of-Thought (Yao et al., 2023a) or Cumulative Reasoning (Zhang et al., 2023b), or incorporating with external tools or knowledge bases, we could potentially improve the overall reasoning performance of multi-agent discussions.

Expanding Task Diversity. While our study mainly focuses on reasoning tasks for assessing both single-agent settings and multi-agent discussions, the adaptive nature of discussions allows for a broader types of applications. Future research could explore the use of agent discussions in diverse scenarios such as real-world strategic planning or the integration of agents into interactive gaming environments.

Experimenting with Additional LLMs. Due to computational and financial constraints, our investigation is limited to testing three LLMs—Bard, Gemini Pro, and ChatGPT-3.5. Expanding our analysis to include additional LLMs could provide a more extensive understanding of the capabilities and variances across different language models, offering valuable insights into the generalizability and scalability of our findings in multi-agent discussion frameworks.

References

Shourya Aggarwal, Divyanshu Mandowara, Vishwajeet Agrawal, Dinesh Khandelwal, Parag Singla, and Dinesh Garg. 2021. Explanations for commonsenseqa: New dataset and models. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 3050–3065.

Philip W Anderson. 1972. More is different: Broken symmetry and the nature of the hierarchical structure of science. *Science*, 177(4047):393–396.

Rohan Anil, Andrew M Dai, Orhan Firat, Melvin Johnson, Dmitry Lepikhin, Alexandre Passos, Siamak Shakeri, Emanuel Taropa, Paige Bailey, Zhifeng Chen, et al. 2023. Palm 2 technical report. *arXiv preprint arXiv:2305.10403*.

Maciej Besta, Nils Blach, Ales Kubicek, Robert Gerstenberger, Lukas Gianinazzi, Joanna Gajda, Tomasz Lehmann, Michal Podstawski, Hubert Niewiadomski, Piotr Nyczyk, et al. 2023. Graph of thoughts: Solving elaborate problems with large language models. *arXiv preprint arXiv:2308.09687*.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Chi-Min Chan, Weize Chen, Yusheng Su, Jianxuan Yu, Wei Xue, Shanghang Zhang, Jie Fu, and Zhiyuan Liu. 2023. Chateval: Towards better llm-based evaluators through multi-agent debate. *arXiv preprint arXiv:2308.07201*.

Justin Chih-Yao Chen, Swarnadeep Saha, and Mohit Bansal. 2023a. Reconcile: Round-table conference improves reasoning via consensus among diverse llms. *arXiv preprint arXiv:2309.13007*.

Xinyun Chen, Maxwell Lin, Nathanael Schärli, and Denny Zhou. 2023b. Teaching large language models to self-debug. *arXiv preprint arXiv:2304.05128*.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. 2022. Palm: Scaling language modeling with pathways. *arXiv preprint arXiv:2204.02311*.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

Constantinos Daskalakis and Seth Matthew Weinberg. 2012. Symmetries and optimal multi-dimensional mechanism design. In *Proceedings of the 13th ACM conference on Electronic commerce*, pages 370–387.

Shizhe Diao, Pengcheng Wang, Yong Lin, and Tong Zhang. 2023. Active prompting with chain-of-thought for large language models. *arXiv preprint arXiv:2302.12246*.

Yilun Du, Shuang Li, Antonio Torralba, Joshua B Tenenbaum, and Igor Mordatch. 2023. Improving factuality and reasoning in language models through multiagent debate. *arXiv preprint arXiv:2305.14325*.

Dheeru Dua, Shivanshu Gupta, Sameer Singh, and Matt Gardner. 2022. Successive prompting for decomposing complex questions. *arXiv preprint arXiv:2212.04092*.

Yao Fu, Hao Peng, Ashish Sabharwal, Peter Clark, and Tushar Khot. 2022. Complexity-based prompting for multi-step reasoning. *arXiv preprint arXiv:2210.00720*.

Simeng Han, Hailey Schoelkopf, Yilun Zhao, Zhenting Qi, Martin Riddell, Luke Benson, Lucy Sun, Ekaterina Zubova, Yujie Qiao, Matthew Burtell, et al. 2022. Folio: Natural language reasoning with first-order logic. *arXiv preprint arXiv:2209.00840*.

Shima Imani, Liang Du, and Harsh Shrivastava. 2023. Mathprompter: Mathematical reasoning using large language models. *arXiv preprint arXiv:2303.05398*.

Tushar Khot, Harsh Trivedi, Matthew Finlayson, Yao Fu, Kyle Richardson, Peter Clark, and Ashish Sabharwal. 2022. Decomposed prompting: A modular approach for solving complex tasks. *arXiv preprint arXiv:2210.02406*.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. Large language models are zero-shot reasoners. *Advances in neural information processing systems*, 35:22199–22213.

Jean-Jacques Laffont and David Martimort. 2000. Mechanism design with collusion and correlation. *Econometrica*, 68(2):309–342.

Yifei Li, Zeqi Lin, Shizhuo Zhang, Qiang Fu, Bei Chen, Jian-Guang Lou, and Weizhu Chen. 2022. On the advance of making language models better reasoners. *arXiv preprint arXiv:2206.02336*.

Tian Liang, Zhiwei He, Wenxiang Jiao, Xing Wang, Yan Wang, Rui Wang, Yujiu Yang, Zhaopeng Tu, and Shuming Shi. 2023. Encouraging divergent thinking in large language models through multi-agent debate. *arXiv preprint arXiv:2305.19118*.

Zhan Ling, Yunhao Fang, Xuanlin Li, Zhiao Huang, Mingu Lee, Roland Memisevic, and Hao Su. 2023. Deductive verification of chain-of-thought reasoning. *arXiv preprint arXiv:2306.03872*.

Ruibo Liu, Jason Wei, Shixiang Shane Gu, Te-Yen Wu, Soroush Vosoughi, Claire Cui, Denny Zhou, and Andrew M Dai. 2022. Mind's eye: Grounded language model reasoning through simulation. *arXiv preprint arXiv:2210.05359*.

Pan Lu, Baolin Peng, Hao Cheng, Michel Galley, Kai-Wei Chang, Ying Nian Wu, Song-Chun Zhu, and Jianfeng Gao. 2023. Chameleon: Plug-and-play compositional reasoning with large language models. *arXiv preprint arXiv:2304.09842*.

Aman Madaan, Niket Tandon, Peter Clark, and Yiming Yang. 2022. Memory-assisted prompt editing to improve gpt-3 after deployment. *arXiv preprint arXiv:2201.06009*.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, et al. 2023. Self-refine: Iterative refinement with self-feedback. *arXiv preprint arXiv:2303.17651*.

Sewon Min, Xinxi Lyu, Ari Holtzman, Mikel Artetxe, Mike Lewis, Hannaneh Hajishirzi, and Luke Zettlemoyer. 2022. Rethinking the role of demonstrations: What makes in-context learning work? *arXiv preprint arXiv:2202.12837*.

Marvin Minsky. 1988. *Society of mind*. Simon and Schuster.

OpenAI. 2022. Chatgpt. https://openai.com/blog/chatgpt.

OpenAI. 2023. Gpt-4 technical report.

Ofir Press, Muru Zhang, Sewon Min, Ludwig Schmidt, Noah A Smith, and Mike Lewis. 2022. Measuring and narrowing the compositionality gap in language models. *arXiv preprint arXiv:2210.03350*.

Teven Le Scao, Angela Fan, Christopher Akiki, Ellie Pavlick, Suzana Ilić, Daniel Hesslow, Roman Castagné, Alexandra Sasha Luccioni, François Yvon, Matthias Gallé, et al. 2022. Bloom: A 176b-parameter open-access multilingual language model. *arXiv preprint arXiv:2211.05100*.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik R Narasimhan, and Shunyu Yao. 2023. Reflexion: Language agents with verbal reinforcement learning. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Kristopher Tapp. 2021. *Symmetry*. Springer.

Gemini Team, Rohan Anil, Sebastian Borgeaud, Yonghui Wu, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M Dai, Anja Hauth, et al. 2023. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023a. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

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

Lei Wang, Wanyu Xu, Yihuai Lan, Zhiqiang Hu, Yunshi Lan, Roy Ka-Wei Lee, and Ee-Peng Lim. 2023. Plan-and-solve prompting: Improving zero-shot chain-of-thought reasoning by large language models. *arXiv preprint arXiv:2305.04091*.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2022. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*.

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

Lilian Weng. 2023. Llm-powered autonomous agents. *lilianweng.github.io*.

Yixuan Weng, Minjun Zhu, Shizhu He, Kang Liu, and Jun Zhao. 2022. Large language models are reasoners with self-verification. *arXiv preprint arXiv:2212.09561*.

Zhiheng Xi, Senjie Jin, Yuhao Zhou, Rui Zheng, Songyang Gao, Tao Gui, Qi Zhang, and Xuanjing Huang. 2023. Self-polish: Enhance reasoning in large language models via problem refinement. *arXiv preprint arXiv:2305.14497*.

Fangzhi Xu, Qika Lin, Jiawei Han, Tianzhe Zhao, Jun Liu, and Erik Cambria. 2023a. Are large language models really good logical reasoners? a comprehensive evaluation from deductive, inductive and abductive views. *arXiv preprint arXiv:2306.09841*.

Xiaohan Xu, Chongyang Tao, Tao Shen, Can Xu, Hongbo Xu, Guodong Long, and Jian-guang Lou. 2023b. Re-reading improves reasoning in language models. *arXiv preprint arXiv:2309.06275*.

Tianci Xue, Ziqi Wang, Zhenhailong Wang, Chi Han, Pengfei Yu, and Heng Ji. 2023. Rcot: Detecting and rectifying factual inconsistency in reasoning by reversing chain-of-thought. *arXiv preprint arXiv:2305.11499*.

Zhicheng Yang, Jinghui Qin, Jiaqi Chen, Liang Lin, and Xiaodan Liang. 2022. Logicsolver: Towards interpretable math word problem solving with logical prompt-enhanced learning. *arXiv preprint arXiv:2205.08232*.

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

Yao Yao, Zuchao Li, and Hai Zhao. 2023b. Beyond chain-of-thought, effective graph-of-thought reasoning in large language models. *arXiv preprint arXiv:2305.16582*.

Eric Zelikman, Yuhuai Wu, Jesse Mu, and Noah Goodman. 2022. Star: Bootstrapping reasoning with reasoning. *Advances in Neural Information Processing Systems*, 35:15476–15488.

Jintian Zhang, Xin Xu, and Shumin Deng. 2023a. Exploring collaboration mechanisms for llm agents: A social psychology view. *arXiv preprint arXiv:2310.02124*.

Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, et al. 2022a. Opt: Open pre-trained transformer language models. *arXiv e-prints*, pages arXiv–2205.

Yifan Zhang, Jingqin Yang, Yang Yuan, and Andrew Chi-Chih Yao. 2023b. Cumulative reasoning with large language models. *arXiv preprint arXiv:2308.04371*.

Zhuosheng Zhang, Aston Zhang, Mu Li, and Alex Smola. 2022b. Automatic chain of thought prompting in large language models. *arXiv preprint arXiv:2210.03493*.

A Discussion Engineering and Agent Symmetry

This section introduces a general framework to understand the discussion engineering of large language models. Let $\mathcal{L}$ be a Large Language Model (LLM) and x be the input text, the forward inference generates a response of surprising quality, which is written as

$$\hat{y} = \mathcal{L}(x). \tag{1}$$

Given the high price of obtaining one large language model, one essential research question is to *unlock the reasoning capability* of large language models so that they can perform better on various tasks. One of the key directions is prompt engineering, where the format and content of the input texts are decorated to improve performances for downstream tasks. Specifically, for a task $\mathcal{T}$, the original input x is decorated as a task-specific prompt input $p(x; \mathcal{T}, \mathcal{L})$. The function $p(\cdot; \mathcal{T}, \mathcal{L})$ is denoted as a prompt decorator for $\mathcal{T}$ and $\mathcal{L}$. Then, output $\hat{y}$ is generated by prompt engineering, which is written as

$$\hat{y} = \mathcal{L}(p(x; \mathcal{T}, \mathcal{L})). \tag{2}$$

Another way to improve the reasoning capability of $\mathcal{L}(x)$ is to leverage multiple times of inference of large language models, leading to a way larger space for **mechanism design**. The output $\hat{y}$ derived by mechanism $\mathcal{M}$ is considered as

$$\hat{y} = M\left(x; \{(\mathcal{L}_i, p_i(\cdot, \mathcal{T}, \mathcal{L}_i))\}_{i=1}^n\right), \tag{3}$$

where M is the mechanism pipeline, $\mathcal{L}_i$ is the i-th inference of LLM, and $p_i(\cdot, \mathcal{T}, \mathcal{L}_i)$ is the corresponding prompt decorator. A mechanism $\mathcal{M}$ is designed by jointly considering $\mathcal{M} = (M, \{(\mathcal{L}_i, p_i)\}_{i=1}^n)$, i.e., how to organize prompt decorated LLM inferences into the mechanism pipeline M.

The discussion engineering, characterized by the mechanism $\mathcal{M} = (M, \{(\mathcal{L}_i, p_i)\}_{i=1}^n)$, includes careful designs at three levels: (1) the (i-th) inference $\mathcal{L}_i$ of an LLM, (2) the prompt decorator p_i for the inference $\mathcal{L}_i$, and (3) the mechanism $M(\cdot, \{\mathcal{L}_i, p_i\})$ to organize the inferences of LLMs. We further introduce the symmetry of the multi-agent system of LLMs to demonstrate the complexity of $\mathcal{M}$. Under our framework, several examples are discussed.

A.1 Agent symmetry in discussion engineering

Symmetry and its breaking is a fundamental concept and widely investigated in science (Anderson, 1972). Symmetry also plays an important role in the context of mechanism design of multiple agents (Laffont and Martimort, 2000; Daskalakis and Weinberg, 2012). Here we introduce a formal description to justify the agent symmetry of the mechanism $\mathcal{M}$.

Definition 1 (Computational graph). *Given the mechanism $\mathcal{M} = (M, \{(\mathcal{L}_i, p_i)\}_{i=1}^n)$, let $G = (V \cup \{x, y\}, E)$ be the directed graph whose node set $V \cup \{x, y\}$ includes the inference operation of LLM and the computational input node and output nodes. x is the node to emit the input text, y is the node that takes the inputs from one or multiple inferences and emits the final output, and $v_i \in V$ is the inference node that describes the i-th LLM inference with $(\mathcal{L}_i, p_i)$. The directed edge in $(v_i, v_k) \in E$ describes the output of inference node v_i is then the input of inference node v_k.*

We note that the graph G contains all the necessary information to determine the mechanism $\mathcal{M}$.

Then we introduce how the LLM agents are associated with the computational graph. To make an LLM conversational session, the entire conversational history is always concatenated as part of the input of each inference call, resulting in many additional connections of the computational graph. The presence of agents allows us to conceptually assume the agents "know" the information in the conversation history, and then the complexity of the graph can be largely reduced. In this paper, we consider the computational graph with agents, so the connections that feed conversation history to the inference nodes are ignored for simplicity.

Let $\mathcal{A} = \{A_1, ..., A_m\}$ be the set of m discussion agents. The inference call of an agent also concatenates its conversation history, denoted as $A_j(x)$.

Definition 2 (Agent assignment). *Let $\mathcal{A} = \{A_1, ..., A_m\}$ be the set of m discussion agents. Each inference node $v_i \in V$ is assigned to an agent $A_j \in \mathcal{A}$. Let $[n]$ present integers from 1 to n, $\alpha : [n] \mapsto [m]$ is the assignment map that assigns the i-th inference to the $\alpha(i)$-th agent. Let $P \in \{0, 1\}^{n \times m}$ be the agent-inference assignment matrix, such that*

$$P_{ij} = \left\{ \begin{array}{ll} 1 & j = \alpha(i), \\ 0 & j \neq \alpha(i) \end{array} \right. . \tag{4}$$

The assignment matrix P is equivalent to the assignment map α.

Then a multi-agent discussion mechanism is defined by the triple $D = (\mathcal{M}, \mathcal{A}, \alpha) = (G, \mathcal{A}, \alpha)$, which is the central object of discussion engineering. Then, we can further introduce the concept of the coloring of the computational graph.

Definition 3 (Agent coloring of the computational graph). *Given the the multi-agent discussion $D = (G, \mathcal{A}, \alpha)$, then each inference node v_i is described by $(\mathcal{L}_i, p_i) = (A_{\alpha(i)}, p_i)$. Let $c_i^D = (A_{\alpha(i)}, p_i)$ be the color of v_i and $C^D(v_i) = c_i^D$ be the color mapping of nodes, and $\mathcal{C}^D = (V, E, C^D)$ be a colored computational graph.*

Then the agent symmetry is established by the permutation operation over the agent set $\mathcal{A}$. Then we can discuss the symmetry by the agent permutation. We begin with the concept of mechanism invariance.

Definition 4 (Mechanism invariance under the agent permutation). *Given the multi-agent discussion $D = (G, \mathcal{A}, \alpha)$ and a permutation mapping $\pi : [m] \mapsto [m]$, the new discussion $D_\pi = (G, \mathcal{A}, \pi \circ \alpha)$ is derived by applying π to the agents. We say a discussion D is invariant under π if and only if there is an isomorphism ϕ between two colored graphs $\mathcal{C}^D = (V, E, C^D)$ and $\mathcal{C}^{D_\pi} = (V, E, C^{D_\pi})$, such that*

(1) ϕ is a bijection,

(2) $\forall (v_i, v_j) \in E, (\phi(v_i), \phi(v_j)) \in E$,

(3) $\forall v \in V, C^D(v) = C^{D_\pi}(\phi(v))$.

We can also define the model invariance to justify the symmetry in the multi-model setting (Chen et al., 2023a).

Definition 5 (Model invariance under the agent permutation). *Given the agent $\mathcal{A} = \{A_1, ..., A_m\}$, and their underlying LLM $\{L_1, .., L_m\}$ (i.e., ChatGPT, GPT4, etc.), the permutation $\pi : [m] \mapsto [m]$ is invariant if for $i = 1, ..., m$ $L_i = L_{\pi(i)}$.*

Following the definition of invariance, the symmetry group of agents is naturally defined following the standard algebra (Tapp, 2021). The largest possible symmetry group for a discussion of m agents is the group S_m. However, the group S_m is not always the symmetry group of a given multi-agent discussion mechanism D, the reasons for not achieving the largest symmetry group can be due to the asymmetry in the mechanism and **asymmetry in models**, respectively. Furthermore, for asymmetric mechanisms, there are two major ways of symmetry breaking. Referring to the three conditions of establishing the isomorphism, the dissatisfaction of condition (2) implies the asymmetry in the computational graph or **asymmetry in the mechanism pipeline** M, and that of condition (3) implies the **asymmetry in prompt decorators**.

B   CMD: Conquer and Merge Discussion Framework

B.1   Motivation

The Debate framework proposes that an increased number of agents and discussion rounds will result in improved performance in multi-agent discussions (Du et al., 2023). Also, the ReConcile framework asserts that a greater number of discussion rounds leads to a higher level of consensus among agents, and the higher the consensus among agents, the more accurate the discussion outcomes become (Chen et al., 2023a). However, as the number of agents increases in a discussion, it leads to increased overhead in each

round of discussion: 1. Each agent has to read more viewpoints from others, resulting in a sharp rise in input tokens. 2. The increase in input tokens puts significant pressure on language models with context token limitations. Inspired by real-life group discussions, we propose a variant framework for Debate called CMD. Fig 3 shows the overview of our method.

B.2 Problem Definition

Assume that there are n agents $\mathcal{A} = \{A_i\}_{i=1}^{m}$ are discussing the given debate task $\mathcal{Q}$, and each agent is a session created from a LLM. Suppose that the maximum number of the discussion rounds is R, the current round is r, the current discussion level is L, and the current active agents set is $\mathcal{A}'$. For each agent $A_i \in \mathcal{A}'$, an answer it generates is $Ans_i = (v_i, e_i)$, where v_i denotes the viewpoint and e_i denotes the explanation. A debate history is $H^{(r)} = \bigcup_{i=1}^{t} Ans_i$ where $t = |\mathcal{A}'|$. $Ans_i^{(r)}$ is generated through the input prompt $A_i(\mathcal{Q}, Ans_i^{(r-1)}, O_i^{(r-1)})$, where $O_i^{(r)}$ indicates the opinions generated by $A_{j \neq i}$ in $(r-1)$-th round. The formal definition of $O_i^{(r)}$ is

$$O_i^{(r)} = \bigcup_{j=1, j \neq i}^{|H^{(r-1)}|} \left(v_i, e_i \cdot \mathbb{1}\{\text{Group}^{(L)}(A_i) = \text{Group}^{(L)}(A_j)\} \right). \tag{5}$$

This represents that A_i will receive all information from the group members while it can only receive viewpoints from agents that are not in the same group. Our goal is to obtain the final result a through an unweighted vote. In the event of a tie, either a can be made by a secretary S, or representatives from each group will proceed to the next level of discussion until the tie is resolved. Each discussion group has only one representative. Note that $\mathcal{Q}$ can be described differently for each A_i. For example, different agents may be asked to hold different views at first.

B.3 CMD Stages

Generally, there are three stages in CMD. In stage 1, the group map and all states will be initialized, then all the participants will generate their initial answers. Then, in stage 2, the participants will continue the discussion during the remaining rounds in groups. When the discussion round reaches the maximum number, it moves to stage 3. In this stage, all participants will vote to get the final answer.

- Stage 1 : Group Discussion Initialization. In this stage, $\mathcal{A}'$ will be initialized as all participants. All participating agents are initially assigned names in uppercase letters, and they are then grouped in sets of three. For getMaxLevel, if secretary mode is on, then it will be set to 1. Otherwise, it will be determined based on the current number of agents. For getGroupMap, the algorithm will automatically generate the groups for all levels. Higher-level groups are generated among the current representatives. All active agents will generate their first response Ans_i.

- Stage 2 : Multi Rounds Discussion. In this stage, the active agents will continue to discuss in the remaining rounds. Assume current round number is r, for each agent $A_i \in \mathcal{A}'$,
  1. The algorithm will start to update the opinions history $O_i^{(r-1)}$ from H. For each record h_j whose sender $A_j (j \neq i)$ stored in H, if A_j and A_i are in the same group, $O_i^{(r-1)}$ will record both the viewpoint and explanation from h_j; otherwise, $O_i^{(r-1)}$ will only record the viewpoint from h_j. After traversing all the records in H, O_i will first gather all opinions of agents from other groups, then $O_i^{(r-1)}$ will gather local group explanations based on group members opinions.

  2. To save up tokens, last round history H will be reset. Then, the new prompt $p_i^{(r)}$ will be generated based on $(\mathcal{Q}, Ans_i^{(r-1)}, O_i^{(r-1)})$. A_i will make a response $Ans_i = (v_i, e_i)$ when given $p_i^{(r)}$. At last, Ans_i will be appended to current round history H for the further use.

  3. Repeat 1. and 2. until the maximum number of discussion rounds is reached.

- Stage 3 : Vote for the Final Result. In this stage, all agents will vote based on their final viewpoints stored in the history $H^{(r)}$ to obtain the result a. If there is no tie, then a will be the final result, thus the discussion is over. If there is a tie and secretary mode is on, then the final result will be obtained

Algorithm 1 CMD: A Conquer and Merge Style Multi-Agents Discussion Framework

---

Require: Debate Task $\mathcal{Q}$, Maximum Discussion Rounds R, Agents $\mathcal{A} = \{A_i\}_{i=1}^n$, Secretary S
Ensure: Final Result a

1: **function** CMD($\mathcal{Q}, R, \mathcal{A}, S$)
2:     $\mathcal{A}' \leftarrow \mathcal{A}$ ▷ Initialize current active agents
3:     $Ng \leftarrow n/3$ ▷ Initialize group numbers, every 3 agents a group
4:     $r \leftarrow 0, L \leftarrow 0$ ▷ Initialize current round and current discussion level
5:     $L_{max} \leftarrow$ getMaxLevel($\mathcal{A}, S$)
6:     $M_g \leftarrow$ genGroupMap($\mathcal{A}, N_g, L_{max}$)   ▷ Assign agents as groups based on discussion levels
7:     $Ans_i \leftarrow \emptyset, O_i \leftarrow \emptyset, H \leftarrow []$ ▷ Initialize answer, others' opinions and history
8:     **while** $L \leq L_{max}$ **do**
9:         **while** $r \leq R$ **do**
10:             **if** $r > 0$ **then**   ▷ Stage 2
11:                 Update O_i from H
12:             **end if**
13:             $H \leftarrow []$
14:             **for each** $A_i \in \mathcal{A}'$ **do**
15:                 $Ans_i \leftarrow A_i(\mathcal{Q}, Ans_i, O_i)$ ▷ $Ans_i = (v_i, e_i)$
16:                 $H \leftarrow H + [Ans_i]$
17:             **end for**
18:             $r \leftarrow r + 1$
19:         **end while**
20:         $a \leftarrow$ AnswerVote(H)
21:         **if** $a \neq$ **Tie then break**
22:         **else if** S **is not None then**   ▷ Check if the secretary mode is on
23:             $a \leftarrow S(\mathcal{Q}, H)$; **break**
24:         **else**   ▷ Secretary mode is off, representative mode is on
25:             $L \leftarrow L + 1$
26:             $\mathcal{A}' \leftarrow$ currentActiveAgents(L, M_g)
27:         **end if**
28:         $r \leftarrow 0$ ▷ The higher-level discussion has commenced
29:     **end while**
30:     **return** a
31: **end function**

---

through S based on the viewpoints and explanations from all sides. If there is a tie and representative mode is on, then the discussion will move to the higher level, and deactivate agents that do not represent their groups. The representatives will be assigned to new groups, and return to Stage 2 for further discussion. If there is still a tie, new representatives will be designated for further discussion, until either the tie is resolved or only one agent is activated for the discussion.

B.4   Message-Passing Algorithm

Below is the detailed message-passing algorithm that synchronize all agents messages during the discussion process. This algorithm supports various discussion architectures.

Algorithm 2 MesSync: A Message-Passing Algorithm for Multi-Agents Communication

---

Require: Discussion Rule $\mathcal{R}$, Agents $\mathcal{A} = \{A_i\}_{i=1}^n$, Agent Attribute Table $\mathcal{T} = \{bot_i\}_{i=1}^n$, Agent Initial Prompt Messages $\mathcal{M} = \{p_j\}_{j=1}^t$

1: **function** MESSYNC($\mathcal{R}, \mathcal{A}, \mathcal{T}, \mathcal{M}$)
2: $\quad$ $Q_{msg} \leftarrow \mathcal{M}$ $\qquad\qquad\qquad\qquad\qquad\qquad$ ▷ Initialize messages storage queue
3: $\quad$ $Q_{send} \leftarrow []$ $\qquad\qquad\qquad\qquad\qquad\qquad$ ▷ Initialize messages to be sent queue
4: $\quad$ $S \leftarrow \mathcal{R}.S_{first}$ $\qquad\qquad\qquad\qquad\qquad\qquad$ ▷ Initialize the first speaker
5: $\quad$ $d \leftarrow 0$ $\qquad\qquad\qquad\qquad\qquad\qquad$ ▷ Initialize the discussion depth to 0
6: $\quad$ **while** $Q_{msg} \neq \emptyset$ **or not** $\mathcal{R}$.isOver **do**
7: $\quad\quad$ **if** Q_{msg} is $\emptyset$ **then**
8: $\quad\quad\quad$ $Q_{msg} \leftarrow$ pushSilenceMessage(d)
9: $\quad\quad$ **end if**
10: $\quad\quad$ $d \leftarrow$ peekDepth(Q_{msg}) $\qquad\qquad\qquad$ ▷ Peek the discussion depth of the first message
11: $\quad\quad$ $M_d \leftarrow$ popAllMessagesAtDepth(Q_{msg}, d)
12: $\quad\quad$ **for each** $A_i \in \mathcal{A}$ **do**
13: $\quad\quad\quad$ $m_i \leftarrow \mathcal{R}$.mergeCommonMessages($M_d, A_i$)
14: $\quad\quad\quad$ Q_{send}.push(m_i)
15: $\quad\quad$ **end for**
16: $\quad\quad$ $m' \leftarrow Q_{send}$.pop()
17: $\quad\quad$ isHold $\leftarrow m'$.hold $\qquad\qquad$ ▷ Signal to merge messages from this round with the next
18: $\quad\quad$ $H \leftarrow \{\}$ $\qquad\qquad\qquad$ ▷ Initialize the map of messages to hold in the current depth
19: $\quad\quad$ $d_{next} \leftarrow d + 2$ **if** isHold **else** $d + 1$
20: $\quad\quad$ **while** m' **and** $m'.d < d_{next}$ **do**
21: $\quad\quad\quad$ **if** m'.hold **then** $H[m'$.name$] \leftarrow m'$.content
22: $\quad\quad\quad$ **else**
23: $\quad\quad\quad\quad$ $S \leftarrow m'$.name
24: $\quad\quad\quad\quad$ $M_{raw} \leftarrow m'$.content $\cup H[S]$
25: $\quad\quad\quad\quad$ $t_{in} \leftarrow \mathcal{R}$.modifyRawInput($M_{raw}$)
26: $\quad\quad\quad\quad$ $t_{out} \leftarrow$ sendText2Bot($t_{in}, S, d, \mathcal{T}$)
27: $\quad\quad\quad\quad$ $t_{out} \leftarrow \mathcal{R}$.validateOutput($t_{in}, t_{out}, S, d$)
28: $\quad\quad\quad\quad$ $\mathcal{A}' \leftarrow \mathcal{R}$.getReceivers($S, d$) $\qquad\qquad$ ▷ Get the receivers of the new message
29: $\quad\quad\quad\quad$ **if** $\mathcal{A}' \neq \emptyset$ **then**
30: $\quad\quad\quad\quad\quad$ $m_{new} \leftarrow$ genMessage($t_{out}, S, \mathcal{A}', d_{next}$) $\qquad$ ▷ (content, sender, receivers, depth)
31: $\quad\quad\quad\quad\quad$ Q_{msg}.push(m_{new})
32: $\quad\quad\quad\quad$ **end if**
33: $\quad\quad\quad$ **end if**
34: $\quad\quad\quad$ $m' \leftarrow Q_{send}$.pop()
35: $\quad\quad$ **end while**
36: $\quad\quad$ **if** $\mathcal{R}$.isOver **then break**
37: $\quad\quad$ **end if**
38: $\quad$ **end while**
39: **end function**

C An `CMD` Example

In the upcoming demonstration, we'll walk you through the CMD framework from the viewpoint of agent **A** among six agents using ChatGPT-3.5.

C.1 Meta Prompt

System

You will be asked to answer a correctness verification task of a deductive reasoing proposition.

Use knowledge only comes from the given premises.

The suffix of your answer should be the your opinion of one of these answer type:

1. [Correct], 2. [Incorrect], 3. [Unknown].

Do not omit the brackets.

Table 4: Vanilla Prompt

C.2 Round 1 Answer

Assistant (ChatGPT-3.5) - Agent A

First let's write down all the premises with labels:

#1. Neocrepidodera Corpulentas are flea beetles or moths.

#2. The species Neocrepidodera Corpulenta is in the Chrysomelidae family.

#3. There are no moths within the Chrysomelidae family.

Next, let's answer the question step by step with reference to the question and reasoning process:

#4. (by #1, #2) Neocrepidodera Corpulenta is either a flea beetle or a moth, and it is in the Chrysomelidae family.

#5. (by #3, #4) Since there are no moths within the Chrysomelidae family, Neocrepidodera Corpulenta must be a flea beetle.

Final Step (by #5): Neocrepidodera Corpulenta is a flea beetle, so the proposition "There are no flea beetles within the Chrysomelidae family" is [Incorrect].

Table 5: The Answer of Agent **A** in Round 1

C.3 Middle System and User Prompts in Round 1

System

There are 2 groups of people discussing on the same topic. I will provide you the detailed opinions and reasoning steps from your group member and opinions from other group members. Use these opinions and your previous opinion as additional advice, note that they maybe wrong. Do not copy other's entire answer, modify the part you believe is wrong.

Other group members' opinions:

Three agents think the proposition is Incorrect.

Your group's opinions:

One agent thinks the proposition is Correct. Below is his answer:

First let's write down all the premises with labels:

. . .

Next, let's answer the question step by step with reference to the question and reasoning process:

#4. (by #1, #2) Neocrepidodera Corpulenta is either a flea beetle or a moth, and it is in the Chrysomelidae family.

#5. (by #3) There are no moths within the Chrysomelidae family.

#6. (by #4, #5) Neocrepidodera Corpulenta cannot be a moth because there are no moths in the Chrysomelidae family.

Final Step (by #6): The proposition "There are no flea beetles within the Chrysomelidae family" is [Correct] because Neocrepidodera Corpulenta, which is in the Chrysomelidae family, cannot be a moth according to premise #5.

One agent thinks the proposition is Incorrect. Below is his answer:

. . .

Next, let's answer the question step by step with reference to the question and reasoning process:

#4. (by #1, #2) Neocrepidodera Corpulenta is a species of Neocrepidodera Corpulentas, so it is either a flea beetle or a moth.

#5. (by #3, #4) Since there are no moths within the Chrysomelidae family, Neocrepidodera Corpulenta must be a flea beetle.

Final Step (by #5): The proposition states that there are no flea beetles within the Chrysomelidae family. However, from premise #2, we know that Neocrepidodera Corpulenta, which is a flea beetle, is in the Chrysomelidae family. Therefore, the proposition is [Incorrect].

User

Use the provided opinions and your previous answer as additional advice critically, please provide an updated answer. Make sure to state your opinion over proposition at the end of the response in the given format: [Correct], [Incorrect] or [Unknown], and such format should be only used to determine the correctness of the proposition.

Table 6: Middle System Prompt at the End of Round 1

C.4 Round 2 Answer

Assistant (ChatGPT-3.5) - Agent A

Based on the opinions provided by the other group members and my previous answer, I will critically evaluate the reasoning steps and provide an updated answer.

The first agent from my group argues that the proposition is correct. Their reasoning is as follows:

#4. (by #1, #2) Neocrepidodera Corpulenta is either a flea beetle or a moth, and it is in the Chrysomelidae family.

#5. (by #3) There are no moths within the Chrysomelidae family.

#6. (by #4, #5) Neocrepidodera Corpulenta cannot be a moth because there are no moths in the Chrysomelidae family.

Final Step (by #6): The proposition "There are no flea beetles within the Chrysomelidae family" is [Correct] because Neocrepidodera Corpulenta, which is in the Chrysomelidae family, cannot be a moth according to premise #5.

The second agent from my group argues that the proposition is incorrect. Their reasoning is as follows:

#4. (by #1, #2) Neocrepidodera Corpulenta is a species of Neocrepidodera Corpulentas, so it is either a flea beetle or a moth.

#5. (by #3, #4) Since there are no moths within the Chrysomelidae family, Neocrepidodera Corpulenta must be a flea beetle.

Final Step (by #5): The proposition states that there are no flea beetles within the Chrysomelidae family. However, from premise #2, we know that Neocrepidodera Corpulenta, which is a flea beetle, is in the Chrysomelidae family. Therefore, the proposition is [Incorrect].

After critically evaluating the reasoning steps, I agree with the second agent from my group that the proposition is incorrect. Neocrepidodera Corpulenta is a species of Neocrepidodera Corpulentas, which are either flea beetles or moths according to premise #1. Since Neocrepidodera Corpulenta is in the Chrysomelidae family according to premise #2, it cannot be concluded that there are no flea beetles within the Chrysomelidae family. Therefore, the proposition is [Incorrect].

Table 7: The Answer of Agent **A** in Round 2

C.5 Middle System Prompt at the End of Round 2

System

There are 2 groups of people discussing on the same topic. I will provide you the detailed opinions and reasoning steps from your group member and opinions from other group members. Use these opinions and your previous opinion as additional advice, note that they maybe wrong. Do not copy other's entire answer, modify the part you believe is wrong.

Other group members' opinions:

Three agents think the proposition is Incorrect.

Your group's opinions:

Two agents think the proposition is Incorrect. Below are their answers:

. . .

User

This part is the same user part shown in table 6.

Table 8: Middle System Prompt at the End of Round 2

C.6 Round 3 Answer

Assistant (ChatGPT-3.5) - Agent A

Based on the opinions provided by the other group members and my previous answer, here is an updated response:

First, let's write down all the premises with labels:

#1. Neocrepidodera Corpulentas are flea beetles or moths.

#2. The species Neocrepidodera Corpulenta is in the Chrysomelidae family.

#3. There are no moths within the Chrysomelidae family.

Next, let's answer the question step by step with reference to the question and reasoning process:

#4. (by #1, #2) Neocrepidodera Corpulenta is either a flea beetle or a moth, and it is in the Chrysomelidae family.

#5. (by #3) There are no moths within the Chrysomelidae family.

#6. (by #4, #5) Neocrepidodera Corpulenta cannot be a moth because there are no moths in the Chrysomelidae family.

Final Step (by #6): The proposition "There are no flea beetles within the Chrysomelidae family" contradicts the given premises, as Neocrepidodera Corpulenta, which is in the Chrysomelidae family, is a flea beetle. Therefore, the proposition is [Incorrect].

After critically evaluating the reasoning steps and considering the opinions provided by the other group members, I agree that the proposition is [Incorrect]. The given premises state that Neocrepidodera Corpulenta is in the Chrysomelidae family and that it is either a flea beetle or a moth. Since there are no moths within the Chrysomelidae family, it can be concluded that Neocrepidodera Corpulenta must be a flea beetle. Therefore, the proposition "There are no flea beetles within the Chrysomelidae family" contradicts the given premises.

Table 9: The Answer of Agent **A** in Round 3

D `CMD` Secretary - A Tie Case Solution

In the event of a tie between two groups, we suggest employing a secretary to resolve such cases. The complete chat transcript below in table 10 illustrates a scenario where the secretary provides the ultimate resolution.

System

6 agents are discussing the correctness of a proposition based on the given premises. The given premises are: · · · The proposition is: "IBM has an office in London or Zurich."

However, now there is a draw:

Three agents think the proposition is Correct. Below is one of their answers: · · ·

Three agents think the proposition is Unknown. Below is one of their answers: · · ·

User

Your task is to carefully determine which opinion is more plausible. Answer opinion types are: [Correct], [Incorrect], [Unknown]. You should give your response in the required format. You are forbidden to copy others' reasoning steps. You can only use the given premises and your reasoning steps to answer the question.

Here are the instructions how you organize your answer format:

First, let's write down all the premises with labels. The labels look like "#{premise_number}."

Next, let's answer the question step by step with reference to the question and reasoing process. There will be a prefix in your every reasoning step with the format "#{number} (by {list_of_premises_and_steps_used})". In your final step, you should come to your conclusion with the format "Final Step (by {list_of_premises_and_steps_used}):".

The suffix of your answer should be the answer type: [Correct], [Incorrect] or [Unknown].

Here are several hints to help you conclude the final answer:

1. Pay attention to the logic of quantifiers and distinguish the meanings of quantifiers such as "all" and "some".

2. When two quantifiers are connected as the premise of existence, the conclusion may not be established, that is, it is unknown.

Assistant (ChatGPT-3.5) - Secretary

First, let's write down all the premises with labels:

#1. Evangelos Eleftheriou is a Greek electrical engineer.

#2. Evangelos Eleftheriou worked for IBM in Zurich.

#3. If a company has employees working for them somewhere, then they have an office there.

#4. IBM is a company.

Next, let's answer the question step by step with reference to the question and reasoning process:

#5 (by #4, #3) Since IBM is a company, and according to premise #3, if a company has employees working for them somewhere, then they have an office there. Therefore, IBM has an office in Zurich.

Now, let's consider the opinions of the other group members:

- One agent thinks the proposition is Correct.

- Two agents think the proposition is Unknown.

Considering the opinions of the other group members, there is disagreement regarding the correctness of the proposition. However, based on the premises and the reasoning process, it can be concluded that IBM has an office in Zurich.

Final Step (by #5): Since IBM has an office in Zurich, the proposition "IBM has an office in London or Zurich" is [Correct].

Table 10: Secretary to solve a tie case.

E Extended Related Work

E.1 Large language models $\mathcal{L}$

With the advent of large language models (LLMs) like GPT-3 (Brown et al., 2020), chatGPT (OpenAI, 2022), GPT-4 (OpenAI, 2023), Llama series (Touvron et al., 2023a,b) and PaLM 2 (Anil et al., 2023), prompt engineering is a burgeoning field that focuses on optimizing the output of language models by crafting effective input prompts.

E.2 Prompt decorator $p(\cdot; \mathcal{T}, \mathcal{L})$ for reasoning

Numerous advancements have been made in the field of improving reasoning abilities of LLMs with prompt engineering. Chain of Thought (CoT) (Wei et al., 2022; Kojima et al., 2022) is a linear problem-solving approach where each step builds upon the previous one. Fu et al. (2022) propose to apply CoT to multi-step reasoning tasks. To automate the CoT, Auto-CoT (Zhang et al., 2022b) constructs demonstrations by sampling diverse questions and generating reasoning chains. Active-Prompt (Diao et al., 2023) aims to select the most uncertain questions for task-specific annotations. Other prompt strategies include PS prompt (Wang et al., 2023) which divides task into subtasks then solve them, effective GoT (Yao et al., 2023b) which models human thought processes as a graph rather than a chain, Natural Program (Ling et al., 2023) which helps to improve the deductive reasoning tasks, re-reading prompt (Xu et al., 2023b) which revisits the question information embedded within input prompts.

E.3 Mechanism $\mathcal{M}$ for reasoning

In addition to focusing on the design of prompts themselves, some works incorporate different mechanisms to assist language models in reasoning.

The design directions of these mechanisms include: 1. Breaking down the reasoning process into multiple stages, with possible verification at each stage. 2. Optimizing the reasoning process via ensemble methods. 3. Iteratively prompting the model for reflection and correction. 4. Utilizing external tools to aid in reasoning. These approaches aim to enhance the overall reasoning capabilities of language models by introducing additional support and guidance throughout the process.

Break Down the Reasoning Process into Multiple Stages. Self-Polish (Xi et al., 2023) make models to progressively refine given problems with multiple stages of prompts. Additionally, some works (Khot et al., 2022; Press et al., 2022; Dua et al., 2022; Zhang et al., 2023b) study over decomposing the tasks into smaller tasks, and use an individual prompt to solve each subtask.

Optimize the Reasoning Process via Ensemble Methods. Tree of Thoughts (ToT) (Yao et al., 2023a) expands on the reasoning process by considering multiple paths of reasoning and self-evaluating choices. Graph of Thoughts (GoT) (Besta et al., 2023) further advances this by modeling information as an arbitrary graph, enabling complex networks of thoughts. And some works propose to first sample reasoning pathsvthen vote for the best one, including self-consistency (Wang et al., 2022) and step-aware verifier (Li et al., 2022). Discussion Engineering is also one category of ensemble methods. We leave it to section 7 for further introduction.

Iteratively Prompt the Model for Reflection and Correction. These works (Zelikman et al., 2022; Weng et al., 2022; Shinn et al., 2023; Madaan et al., 2023; Chen et al., 2023b; Xue et al., 2023) are mainly based on iteratively asking model to find the mistakes or inconsistencies among previous reasoning steps or the knowledge, then solve them one by one.

Utilize External Tools or Knowledge to Aid in Reasoning. Mind's Eye (Liu et al., 2022) proposes to use physical simulator to help models reason in the physical world. There are also other works (Lu et al., 2023; Imani et al., 2023) will equip models with tools to solve problems. Further more, some works like MemPrompt (Madaan et al., 2022) and LogicSolver (Yang et al., 2022) use external knowledge to assist LLMs with reasoning.