Who's the MVP? A Game-Theoretic Evaluation Benchmark for Modular Attribution in LLM Agents

Anonymous ACL submission

Abstract

Large Language Model (LLM) agents frameworks often employ modular architectures, incorporating components such as planning, reasoning, action execution, and reflection to tackle complex tasks. However, quantifying the contribution of each module to overall system performance remains a significant challenge, impeding optimization and interpretability. To address this, we introduce **CapaBench** (**Capa**bility-level Assessment **Bench**mark), an evaluation framework grounded in cooperative game theory's Shapley Value, which systematically measures the marginal impact of individual modules and their interactions within an agent's architecture. By replacing default modules with test variants across all possible combinations, CapaBench provides a principle method for attributing performance contributions. **Key contributions** include: (1) We are the first to propose a Shapley Value-based methodology for quantifying the contributions of capabilities in LLM agents; (2) Modules with high Shapley Values consistently lead to predictable performance gains when combined, enabling targeted optimization; and (3) We build a multi-round dataset of over 1,500 entries spanning diverse domains and practical task scenarios, enabling comprehensive evaluation of agent capabilities. CapaBench bridges the gap between component-level evaluation and holistic system assessment, providing actionable insights for optimizing modular LLM agents and advancing their deployment in complex, real-world scenarios.

1 Introduction

The rapid advancements in Large Language Models (LLMs) have ushered in a transformative era for artificial intelligence agents. These models demonstrate unprecedented capabilities in understanding, generating, and integrating natural language across diverse domains (Brown et al., 2020; OpenAI et al., 2024). However, LLMs still face notable challenges as foundational models for supporting AI agents in real-world applications. These include accurately interpreting subtle contextual shifts, effectively integrating with external tools, and ensuring both the accuracy and reliability of outputs. To overcome these challenges, researchers have increasingly adopted modular architectures, decomposing agents into distinct components responsible for planning, reasoning, and action execution. Such modular frameworks not only enhance the overall performance but also improve the interpretability and maintainability of the systems. Frameworks such as ReAct (Yao et al., 2022) and AutoGPT (Tang et al., 2023) exemplify how structured workflows, achieved by breaking down tasks into manageable modules, can lead to more efficient task processing. These modular architectures lay the groundwork for systematic evaluations of LLM agents' internal designs and effectiveness in various applications.

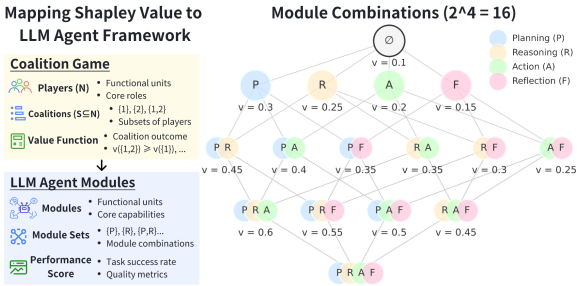


Figure 1: Conceptual Mapping between Coalition Game Theory and LLM Agent Evaluation. The left shows the mapping from coalition game theory to LLM agents, the right lists all possible combinations ($2^4 = 16$) with performance values.

Despite the impressive capabilities of LLM agents, accurately evaluating their performance remains an open challenge. Traditional evaluation methods have predominantly focused on task-specific benchmarks and domain-specific datasets. For instance, AgentBench (Liu et al., 2023) assesses agents' abilities through specialized tasks, while ToolBench (Guo et al., 2024) evaluates the effectiveness of LLM agents in leveraging external

tools across diverse application scenarios. Additionally, MMAU (Yin et al., 2024) investigates the capabilities of LLM Agents across a wide range of tasks. However, these benchmarks often rely on reductive assumptions, equating task success (e.g., solving a math problem) with broader cognitive abilities (e.g., reasoning). This simplification neglects the complex interactions between an agent's internal components, leading to an incomplete understanding of their true potential. The current task-oriented evaluation framework faces several key challenges. First, LLM agents simultaneously require the integration of multiple capabilities to solve complex tasks. For example, solving a mathematical problem may necessitate reading comprehension, tool usage, and structured output generation. Second, existing methods fail to account for the interactions between architectural components and their collective contributions to overall system behavior. Additionally, task-specific success rates provide limited insight into the relative contributions of individual modules, making it difficult to identify key areas for optimization. Consequently, there is a pressing need for evaluation frameworks that can dissect and quantify the contributions of each module within modular LLM agents.

To address these challenges, we propose a novel evaluation framework, **CapaBench**, which integrates the assessment of modular architectures with the evaluation of agent capabilities. CapaBench systematically quantifies the contributions of individual modules (e.g., planning, reasoning, action execution, reflection) within LLM architectures using the Shapley Value (Hart, 1989), a cooperative game theory metric that fairly attributes performance based on all possible permutations of module contributions. This approach captures direct contributions and interaction effects at the same time, offering a rigorous and interpretable evaluation of system dynamics. Our method provides several key advantages: (1) evaluating the contributions of each module by capturing nuanced dynamics; (2) using a mathematically sound attribution method to enhance interpretability of agent performance; and (3) enabling predictions about system performance based on specific module combinations, supporting targeted optimizations. To the best of our knowledge, CapaBench is **the first framework** to systematically quantify and attribute module contributions in LLM-based agents using the Shapley Value approach.

Furthermore, to ensure that our evaluation reflects realistic, multi-faceted application scenarios, we build a **large-scale** dataset of over **1,500 multi-round tasks** spanning a diverse range of categories (e.g., shopping, navigation, ticket ordering, operating system, robot control, math, and theorem proving). These tasks integrate various capabilities such as planning, tool usage, and reflection, thereby requiring holistic agent performance rather than isolated skill assessments. Our dataset will be open-sourced in the future to support further research and development, and we are actively adding more scenarios to broaden its coverage and applicability.

CapaBench makes the following contributions:

- **Novel Evaluation Framework:** We are the first to propose a Shapley Value-based methodology for quantifying the contributions of capabilities in LLM agents.
- **Predictive Module Combinations:** Experiments reveal that modules combined with higher Shapley Values consistently improve task success, offering clear guidance for optimizing module integration to maximize performance.
- **Large-Scale Dataset:** We build a multi-round dataset with over 1,500 entries spanning diverse domains such as daily activities, computation, and role control. The dataset is designed to challenge multiple agent capabilities simultaneously, serving as a robust testbed for evaluating LLM agents. Our dataset will be released in the future to facilitate further research and development.

2 Related Work

2.1 LLM Agent

Recent advances in large language models (LLMs) have catalyzed the development of increasingly sophisticated AI agents. LLM agents typically employ modular architectures that decompose tasks into planning, reasoning, and action execution. Early work, such as ReAct (Yao et al., 2022), highlighted the efficacy of explicit reasoning and action paradigms. Recent efforts, such as AutoGPT (Tang et al., 2023) pioneered autonomous task execution through iterative planning and reflection. MetaGPT (Hong et al., 2024), introduced hierarchical planning strategies that enable dynamic task decomposition and recursive self-improvement. Building on these works which highlight modular designs, our study systematically evaluates the marginal impact of individual modules using the Shapley Value, uncovering the most

suitable combinations of LLM modules for achieving optimal performance in different environments.

2.2 Agent Benchmark

The evaluation of LLM agents has evolved considerably, with early approaches primarily emphasizing task-specific performance metrics. AgentBench (Liu et al., 2023) laid the groundwork by evaluating agents across diverse scenarios, such as web browsing and knowledge graph, highlighting the importance of assessing performance in diverse contexts. However, these evaluations often focused on task outcomes while overlooking the foundational skills driving these results, making it difficult to analyze the root causes of failures. To address this limitation, MMAU (Yin et al., 2024) introduced a novel benchmark that provides an evaluation of agent capabilities. But it driectly combined Agents' capabilities with specific tasks. Recent benchmark developments have become increasingly sophisticated. OmniACT (Zhang et al., 2024) introduced a framework for evaluating agents in desktop environments, while AgentQuest (Yang et al., 2024) developed methods for assessing continuous learning and adaptation. These frameworks represent a shift toward understanding not just what agents can do, but how they handle complex, dynamic scenarios. In contrast, CapaBench extends beyond capability-level evaluations by leveraging the Shapley Value to quantitatively capture each module contributions, enabling a more nuanced analysis of how each component influences overall agent performance.

3 Benchmark Design

We build the agent framework shown in Figure 2 as the foundation of our benchmark. This framework is specifically designed to assess LLM agents' abilities in various environments and task scenarios. It follows established agent processes and features a modular design, which supports both single-turn and multi-turn interactions. This ensures that our evaluations are comprehensive and adaptable.

3.1 Agent Capability

Building upon established works (Yao et al., 2022; Tang et al., 2023; Hong et al., 2024), our framework integrates 4 fundamental capabilities for LLM agents: Planning, Reasoning, Action, and Reflection, as illustrated in Figure 2. These capabilities represent the core functionalities widely recognized in current agent systems: **(1) Planning module**

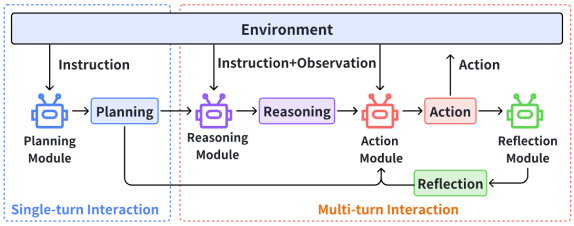


Figure 2: Agent Workflow in CapaBench.

initiates the agent workflow by decomposing complex instructions into structured subtasks, following principles established in hierarchical planning systems (Brown et al., 2020). This decomposition enables effective task prioritization and resource allocation, particularly crucial for multi-step operations requiring strategic foresight; **(2) Reasoning module** extends the ReAct framework (Yao et al., 2022) by incorporating both instruction context and environmental observations. Through chain-of-thought mechanisms (Wei et al., 2022), this module performs logical inference and causal analysis to determine appropriate action sequences. Integration with the planning module enables dynamic adjustment of reasoning strategies based on evolving task requirements; **(3) Action module** implements the execution interface, translating cognitive processes into concrete operations. This approach builds on established action space formalization (Guo et al., 2024), ensuring consistent mapping between internal state representations and external behaviors. The module maintains state awareness through continuous environment monitoring, enabling responsive behavior adaptation; **(4) Reflection module** completes the architecture by implementing systematic performance analysis, drawing from recent advances in self-improving systems (Yin et al., 2024). Operating primarily in multi-turn scenarios, this module enables iterative refinement of agent behavior through structured outcome analysis and strategy adjustment.

3.2 Evaluation Methodology

We use Shapley Value (Hart, 1989) from cooperative game theory to evaluate module contributions in LLM architectures. This method quantifies each module's marginal impact on performance by analyzing all module configurations, capturing both individual contributions and interaction effects through task success rates.

Shapley Value Framework Shapley Value provides a theoretical foundation for fairly allocating the overall performance of a system to its individ-

ual components. For a set of N modules, Shapley Value $\phi_i(v)$ for module i is defined as:

$$\phi_i(v) = \sum_{S \subseteq N \setminus \{i\}} \frac{|S|!(|N| - |S| - 1)!}{|N|!} \left[v(S \cup \{i\}) - v(S) \right], \quad (1)$$

where S denotes any subset of N that excludes module i, and $v(S)$ represents the performance(task success rate) of the agent when only the modules in S are active. The term $v(S \cup \{i\}) - v(S)$ quantifies the marginal impact of adding module i to the subset S, while the weight $\frac{|S|!(|N|-|S|-1)!}{|N|!}$ ensures fair averaging across all possible subsets.

Evaluation Flow CapaBench systematically evaluates the contributions of four key modules in the agent architecture: Planning (P), Reasoning (R), Action (A), and Reflection (F). As shown in Figure 1, the evaluation involves testing all possible combinations of these modules ($2^4 = 16$ combinations) by replacing default implementations with test variants provided by the target LLM model. The default "whiteboard" modules, implemented using Llama3-8b-instruct, serve as a fixed baseline to isolate the performance impact of each test module. Llama3-8b-instruct was chosen as the default model implementation because it is open-source, lightweight, and easy to deploy, making it practical for extensive testing. While it possesses basic task completion capabilities, its moderate success rates provide an ideal baseline to observe and quantify the impact of replacing modules with more advanced test models. For each combination, CapaBench computes performance values to quantify the contribution of individual modules and their interactions. Diverse task benchmarks (B), including multi-step scenarios designed to simulate practical agent applications, are used to evaluate the system, providing insights into the optimal module configurations for various environments.

Capturing Synergistic Effects and Nonlinear Dynamics Shapley Value provides a robust framework to quantify both the independent contributions and synergistic interactions among modules in a modular architecture. By systematically evaluating all possible subsets $S \subseteq N$, it inherently captures the nonlinear dynamics and interdependencies between modules. For instance, Planning provides structured outputs for Reasoning, while Reasoning refines these outputs to guide Action execution. Tasks often require at least two modules to collaborate, such as Reasoning and Action working together to decompose and solve complex tasks. These collaborative effects are reflected in the marginal contributions $v(S \cup \{i\}) - v(S)$, where $v(S)$ represents the system's performance (e.g., task success rate) with subset S. Shapley Value is particularly well-suited for nonlinear dynamics, as it fairly distributes contributions even when module interactions exhibit synergy or competition. Unlike linear or additive methods, it ensures unbiased attribution of both individual and collaborative contributions, making it ideal for evaluating modular LLM agents with complex interdependencies.

3.3 Dataset Construction

Online Shopping Tasks are based on WebShop platform (Yao et al., 2023), consisting of 110 tasks, with 48 modified to increase diversity and complexity. For example, the instruction "find me scrubs & body treatments made with tea tree and other natural ingredients" was changed to "Given my upcoming spa weekend, recommend scrubs & body treatments with tea tree for sensitive skin." These changes introduce more natural, context-rich queries, testing agents' reasoning, personalization, and relevance. The evaluation framework aligns with WebShop's reward model and product definitions, ensuring consistent performance assessment.

Navigation Planning The Navigation Planning task(Lin et al., 2024) evaluates agents' ability to generate and adapt travel itineraries under evolving constraints. Our enhanced dataset of 250 tasks simulates dynamic planning through preference evolution mechanisms, where users initially provide three core requirements (e.g., budget limits, preferred activities) with 50% probability of introducing new constraints during interactions. The evaluation combines constraint adherence measurement for requirement fulfillment accuracy with route optimality assessment through multi-criteria scoring of spatial efficiency and preference alignment. The metric includes precision from experiments and route rationality based on user preferences (e.g., budget, activities), reflecting agent's ability to prioritize needs and generate actionable plans.

Ticket Ordering Ticket Ordering assesses agents' ability to find the best flight combination for two users based on their needs and constraints. Inspired by (Lin et al., 2024), it includes 150 tasks simulating real-world ticket ordering scenarios. Agents must consider users' calendars, flight prices, and arrival times to provide an optimal flight combination. The evaluation focuses on minimizing

Table 1: Capability and number of data per dataset. P, R, A, F represent Planning, Reasoning, Action, Reflection. Checkmarks indicate the emphasis of each capability in per task.

		Daily Activities			Computation Tasks			Role Control
		Shopping	Navigation	Ticket	Math	ATP	OS	Robot
	Sample Count	**110**	**250**	**150**	**250 × 2**	**111 × 3**	**102**	**80**
P	Task Steps	✓			✓	✓		
	Resource Constraints		✓	✓			✓	✓
R	Logical Validation				✓	✓	✓	
	Knowledge Inference	✓	✓	✓				✓
A	Environmental Actions				✓	✓	✓	
	Interactive Actions	✓	✓	✓				✓
F	Failure Analysis	✓	✓	✓	✓	✓	✓	✓

three factors: flight price, calendar conflicts, and the difference in users' arrival times, with higher scores for more affordable, conflict-free options and closer arrival times.

Math Solver Math Solver evaluates agents' ability to solve diverse mathematical problems by incorporating **tool usage** into the process. To address the lack of detailed classification in Math (Hendrycks et al., 2021), we selected 5 key concepts and 10 difficulty levels and we created 250 new questions for both algebra and geometry tasks. We designed two tools fpr agents to solve problems:

- A **pseudo 'search engine'** with 200 curated knowledge points, allowing agents to retrieve the top 3 relevant points.
- A **calculator** for numerical computations.

Automatic Theorem Proving (ATP) ATP evaluates agents' ability to construct formal proofs for complex logical problems. The MINIF2F dataset (Zheng et al., 2021), which includes Olympiad-level problems, is partially outdated, as it uses Lean 3 (now replaced by Lean 4) and lacks coverage of Coq, another widely used formal proof language. ATP emphasizes iterative proof construction, requiring agents to use formal verification tools like Coq, Lean4, and Isabelle3 (The Coq Development Team; The Lean Prover Team; The Isabelle Team). These tools enforce strict syntax and dynamic reasoning, requiring agents to adjust strategies based on the proof's current state, simulating human-like reasoning in formal logic problem-solving.

Operating System The OS dataset assesses agents' abilities to interact with a simulated terminal for Ubuntu and Git tasks. For Ubuntu, we expanded AgentBench-OS framework (Liu et al., 2023) using GPT-4, focusing on file manipulation, system settings, and process management. Agents propose bash commands, receive terminal feedback, and use reflection (e.g., checking command success with $(echo?)$) to address failures. For Git, we constructed dataset from Learn Git Branching (The learnGitBranching Team) , which requires agents to transform an initial git tree into a target state using commands. Reflection is triggered if no changes occur after two steps, enhancing agents' reasoning and adaptability.

Robot Cooperation Robot Cooperation tasks, derived from RoCo (Mandi et al., 2023), evaluate agents in real-world-inspired robotic scenarios across five tasks: Sweep Floor, Move Rope, Arrange Cabinet, Make Sandwich, and Sort Cubes. We expanded these tasks with diverse instances and constraints, such as color-specific sequencing in Sweep Floor and sequential logic in Arrange Cabinet. Using RoCo's Central Plan mode, agents receive full environment observations and plan actions for all robots simultaneously.

4 Experiment

4.1 Experimental Implementation

In our experiments, we use **Llama3-8B-Instruct** as the default for all four core modules: planning, reasoning, action, and reflection. For each evaluation, we replace one module with its test variant (driven by the test model), keeping the other modules default. This creates 16 configurations for the four-module architecture. For each configuration S, we measure measure the task success rate $v(S)$ across various benchmark scenarios to ensure robust and representative performance.

We evaluate 9 large language models, categorized into 3 groups:

- **Closed API Models**: This includes four widely used commercial API-based models: Anthropic/Claude-3.5-Sonnet, OpenAI/GPT-4-turbo-0409, OpenAI/GPT-4o-mini, GLM-4-air, and Doubao-pro-4k.
- **Mid-parameter Open-Source Models (32B-100B)**: To assess mid-scale models, we evaluate three models: Llama3.1-70B-Instruct and Mixtral-8x7B-Instruct-v0.1 (46.7B).
- **Low-parameter Open-Source Models ($\leq$32B)**: For lightweight models, we include Qwen2.5-32B-Instruct and Mistral-8B-Instruct-v0.2.

All experiments are conducted on NVIDIA A100-80GB GPUs, with vLLM employed for efficient inference of open-source models.

Table 2: Experimental Results Across Datasets. Metrics for baseline models are highlighted in blue. Results marked with '*' below each dataset indicate the best-performing model combinations computed based on Shapley Value.

Dataset	Metric	Llama3 8B	Claude 3.5	gpt-4o mini	glm-4 air	qwen2.5 32B	Mistral 8X7B	Mistral 7B	gpt-4 turbo	doubao pro-4k	Llama3 70B
Online Shopping *Acc: 43.31**	Pt	–	-0.004	0.071	**0.106**	-0.030	-0.048	0.024	0.026	0.071	-0.028
	Rt	–	0.019	-0.025	**0.077**	0.004	0.036	0.016	-0.074	0.011	0.005
	At	–	0.056	0.068	-0.059	**0.156**	0.080	0.004	0.014	-0.045	0.117
	Ft	–	-0.009	-0.003	-0.011	-0.021	-0.015	-0.022	**0.024**	-0.040	-0.030
	Acc (%)	26.27	32.43	37.43	**37.50**	37.18	31.67	28.48	25.31	25.95	32.61
Navigation Planning *Acc: 74.42**	Pt	–	0.000	0.006	0.001	-0.002	0.021	**0.023**	0.008	0.001	-0.009
	Rt	–	0.030	0.027	-0.008	0.012	-0.035	**0.055**	0.014	-0.003	-0.019
	At	–	**0.106**	0.081	0.005	0.099	0.048	0.042	0.099	-0.051	0.046
	Ft	–	-0.006	0.002	-0.021	**0.018**	-0.029	0.007	0.004	-0.033	-0.011
	Acc (%)	58.70	**71.90**	70.29	61.91	68.26	64.45	71.48	71.23	50.90	59.32
Ticket Ordering *Acc: 67.18**	Pt	–	0.003	0.032	-0.195	0.119	**0.183**	-0.111	-0.043	0.151	0.004
	Rt	–	0.186	0.243	0.172	0.181	0.054	-0.070	**0.301**	-0.001	0.089
	At	–	**0.217**	0.049	-0.020	-0.000	-0.083	-0.020	0.028	0.006	-0.275
	Ft	–	0.024	0.005	-0.006	0.043	-0.011	0.002	**0.058**	-0.027	-0.001
	Acc (%)	19.94	**62.85**	51.82	15.01	54.25	34.24	0.00	54.37	32.88	1.59
Math *Acc:83.80**	Pt	/	0.038	0.067	0.056	0.065	0.005	-0.060	0.048	**0.115**	0.028
	Rt	/	**0.131**	0.021	0.044	0.107	0.003	-0.000	0.065	0.059	0.031
	At	/	0.442	0.343	0.348	0.483	0.164	-0.044	**0.492**	0.182	0.327
	Ft	/	0.042	**0.043**	0.005	0.031	-0.014	-0.003	0.022	-0.002	0.006
	Acc (%)	18.00	83.40	65.40	63.20	**86.60**	33.80	7.20	80.60	53.40	57.20
ATP *Acc: 86.79**	Pt	/	0.012	0.018	0.002	0.018	**0.025**	0.008	0.012	0.016	0.019
	Rt	/	**0.057**	-0.016	0.005	0.030	0.018	0.010	0.027	0.019	-0.056
	At	/	**0.660**	0.345	0.161	0.511	0.039	-0.009	0.541	0.084	0.125
	Ft	/	**0.069**	0.015	0.021	0.037	-0.011	-0.000	0.023	0.004	0.011
	Acc (%)	5.45	**85.29**	41.74	24.32	65.17	12.61	6.31	65.77	17.72	15.32
Robot Cooperation *Rwd: 92.63**	Pt	–	**0.114**	0.075	-0.024	0.090	-0.005	-0.014	0.107	0.021	0.043
	Rt	–	**0.388**	0.189	0.116	0.268	0.033	-0.000	0.329	-0.004	0.152
	At	–	**0.319**	0.196	0.008	0.277	0.052	-0.021	0.316	0.204	0.175
	Ft	–	**0.017**	-0.003	-0.012	0.003	0.004	-0.001	0.001	-0.012	-0.008
	Reward (%)	8.85	**92.63**	54.43	17.60	72.59	17.27	5.17	84.18	29.75	45.06
Operating System *Acc: 60.78**	Pt	–	**0.078**	0.042	0.047	0.060	0.032	0.004	0.050	0.065	0.077
	Rt	–	**0.458**	0.305	0.305	0.311	0.194	0.047	0.395	0.215	0.313
	At	–	**0.071**	0.065	0.041	0.053	0.009	0.019	0.070	0.060	0.040
	Ft	–	-0.008	0.020	0.004	**0.037**	0.001	0.019	0.005	-0.006	0.012
	Acc (%)	0.98	**60.78**	44.12	40.71	47.06	24.51	9.80	52.94	34.31	45.10

4.2 Main Results

Module Impact via Replacement The experimental results in Figure 4 show that module replacement accurately reflects its impact on system performance, as seen with `Claude-3.5-Sonnet` on Algebra. High-contribution modules, identified via Shapley Value calculations, lead to significantly better performance. For example, `(P,R,A)` achieves 78.0%, far surpassing the baseline configuration with Llama3-8b-Instruct at 21.6%. Incremental replacements align with predictions: substituting the default Planning module (`P`) improves performance to 18.4%, and adding a strong Action module (`A`) raises it to 63.2%. Configurations like `(P,R,A)` maximize performance through synergy, while low-contribution modules, such as `(P,F)`, result in poor performance (0.212). These results confirm the accuracy of Shapley Values in quantifying module contributions.

Predictive Module Combinations The experimental results in Table 2 demonstrate that modules with higher Shapley Values consistently lead to improved task performance when combined. For instance, in the "Online Shopping" dataset, the optimal combination achieves an accuracy of 43.31%, which is significantly higher compared to the other models, indicating the advantage of leveraging high-contribution modules. Similarly, in ATP, the best combination computed based on Shapley Values results in an 86.79% accuracy, showcasing a marked improvement over alternatives. These results demonstrate that identifying and integrating key modules with high Shapley Values enables CapaBench to systematically maximize performance across tasks, validating Shapley Values as a reliable guide for module selection and optimization.

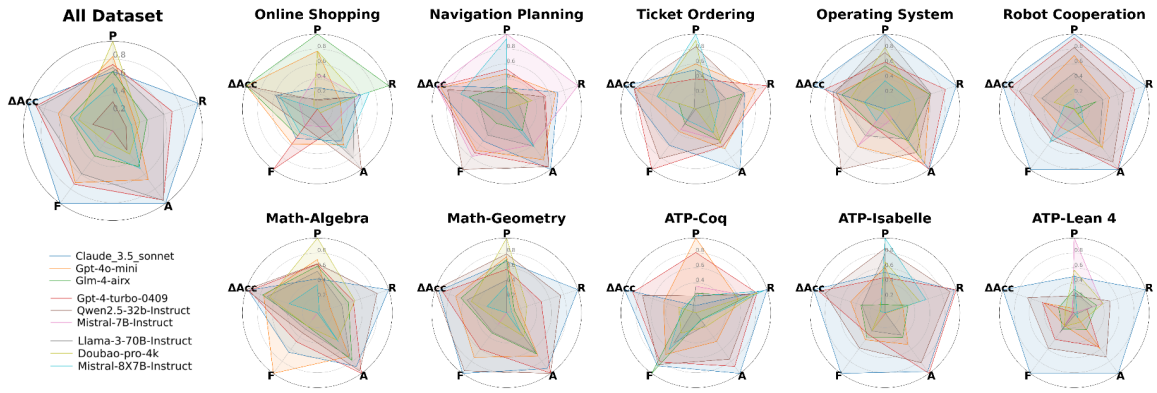


Figure 3: Radar plot comparing model performance across tasks with key contributions.

4.3 Ablation Study

In this section, we examine how changing the default model in our evaluation framework affects Shapley Value results and the relative ranking of various LLMs. Specifically, we replace the original default model (`Llama3-8B-instruct`) with `gpt-3.5-turbo-0613` and re-run the evaluation on the same set of seven test LLMs over Robot Cooperation. Figure 5 illustrates the Shapley Value results for the four modules under 2 default models. Although the absolute Shapley Values vary due to the differences in baseline models' capabilities, our primary focus is on the consistency of test model rankings. To quantify this consistency, we define the *preference pair consistency rate* as

$$\text{Pairwise Consistency Rate} = \frac{\{\text{Consistent Preference Pairs}\}}{\{\text{All Model Pairs}\}}$$

which measures the proportion of test model pairs that maintain the same relative ranking across both experiments. A higher rate indicates that changes to the default model have minimal impact on the relative ranking of test models.

Results show that Reasoning achieves the highest consistency rate (91.67%), followed by Action (86.11%), Planning (72.22%), and Reflection (58.33%). The overall consistency rate (85.18%) confirms that our framework is robust to changes in the default model for most modules. Notably, *Reasoning* and *Action*, which contribute most to task success according to Shapley Values, also show the highest ranking consistency. In contrast, *Reflection* exhibits the lowest consistency (58.33%), suggesting that its assessment is more sensitive to the default model choice or that it requires further refinement. While absolute Shapley Values shift with a stronger or weaker default model, the *relative* ordering of test models—and the key insights into each model's strengths and weaknesses—remains largely stable.

4.4 Analysis

Cross-Task Model Performance Comparison A comparison of model performance across diverse tasks reveals distinct strengths and weaknesses. Notably, `Claude-3.5` excels in most categories, particularly in formal verification (e.g., Coq, Lean 4, Isabelle) and robot cooperation tasks. This suggests a strong chain-of-thought reasoning mechanism and effective multi-agent collaboration strategies—capabilities essential for tasks requiring precise logic and synchronized actions. In contrast, open-source models like `Qwen-2.5` and `Mistral-8X7B` perform well in more straightforward domains, such as shopping or basic Algebra, but lag in cognitive-heavy tasks like automatic theorem proving and robot cooperation. These models, while adept at routine queries and procedural problem-solving, lack the deeper reasoning, advanced planning, and specialized modules required for complex, multi-stage tasks. Improving these areas through fine-tuning on specialized data or integrating more advanced tools could help bridge the gap between open-source and proprietary models.

Module Contribution Patterns Our findings highlight that module contributions vary according to task demands, reflecting the distinct cognitive processes involved. Specifically:

- **Tasks with High Cognitive Complexity (e.g., Shopping, Robot Cooperation, and OS):** *Reasoning* and *Planning* play pivotal roles. Online shopping requires balancing constraints (e.g., budget and preferences) and sequencing decisions effectively. In robot cooperation, *Reasoning* enables dynamic information updates and efficient task distribution among agents. Operating system tasks, involving troubleshooting and resource management, rely heavily on real-time problem-solving and feedback interpretation. Across these tasks, robust *Reasoning* en-

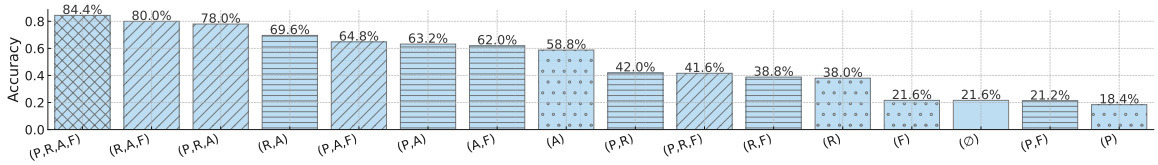


Figure 4: Shapley value results of all combinations in Math (Algebra) for Claude-3.5-Sonnet under different configurations. The pattern of the bars indicates the number of modules (ranging from 0 to 4) that Claude is involved in.

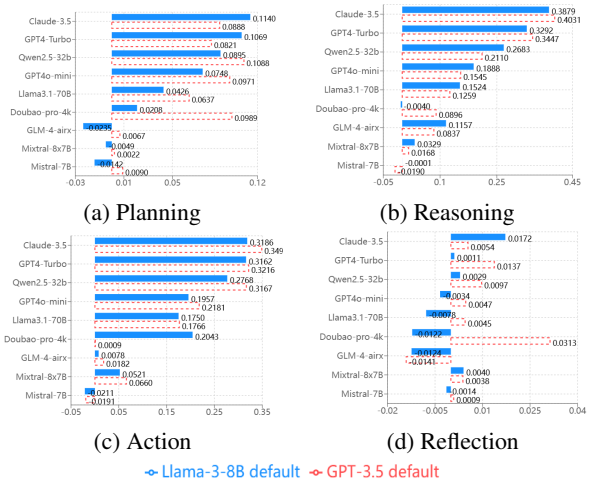


Figure 5: Compare default LLMs on Robot Cooperation.

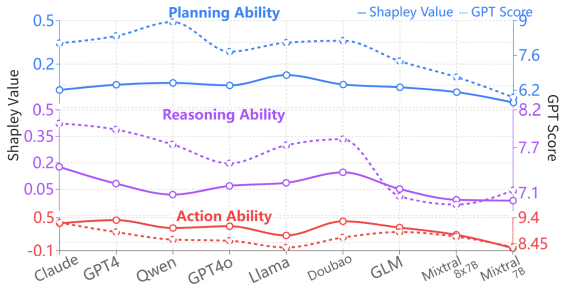


Figure 6: Planning, Reasoning, and Action Evaluation on Algebra. The left Y-axis shows Shapley Value with solid lines and the right Y-axis shows the GPT scores with dashed lines.

sures logical inference and decision-making under uncertainty.

- **Tasks Requiring Precision (e.g., Math Solvers and ATP):** *Action* is the dominant module. In math solvers, particularly geometry, precise procedural execution, such as applying theorems or constructing diagrams, outweighs strategic planning. Similarly, in formal verification tasks (e.g., Coq or Lean), strict adherence to syntactic and semantic correctness is critical. Both scenarios demand meticulous step-by-step actions to ensure reliability and prevent errors.

Low Reflection Contribution We observe that the Reflection module has a limited impact on task performance, likely due to two main factors. First, task success alone doesn't fully capture the quality or effectiveness of reflection; a model's ability to 'think about' its mistakes is not reflected in success rates. Second, without guidance from a more capable model, the Reflection module may struggle to identify the root causes of errors. As a result, the lack of deeper insights into these errors limits its practical impact on task outcomes.

Comparative Study We assessed the effectiveness of Shapley Values in capturing LLMs' abilities like Planning, Reasoning, and Action with 238 questions from successfully completed trajectories in Algebra dataset. We divided these trajectories into *single-step QA samples* based on the three core modules, resulting in 2180 samples. The Reflection module was excluded due to its minimal impact on overall success rates and thus the insufficient number of successful trajectories to build a reliable dataset for evaluation. For each single-step sample, the responses from the test models were evaluated by GPT-o1-mini. The evaluation criteria focused on two aspects for the Planning and Reasoning modules: **semantic rationality**, which assesses whether the response is clear and understandable, and **task completion**, which measures how effectively the task was completed. For the Action module, the evaluation focused on **logical comprehension ability**, scoring based on the model's understanding of the task logic and its ability to execute correct actions based on the Planning and Reasoning modules. Figure 6 shows the Shapley Values and the scores by GPT-o1-mini of each model, with Pearson correlation coefficients of **0.81, 0.77, 0.67** for the Planning, Reasoning, and Action modules, respectively. These high correlations validate the effectiveness of Shapley Values in quantifying each module's specific contribution to task success.

5 Conclusion

This paper introduces **CapaBench**, a benchmark using Shapley Value to evaluate the contributions of individual modules in LLM agents. By analyzing interactions among planning, reasoning, action, and reflection, we enable precise attribution, guiding optimization and predicting performance across diverse tasks. Moving forward, we aim to expand the variety of senarios and develop domain-specific protocols to reduce computational costs while maintaining module-level insights.

6 Limitations

In our work, we use Shapley Value to fairly attribute performance to the individual modules within LLM agents. Computing Shapley Values requires evaluating all possible module combinations, leading to an exponential increase in computational complexity as the number of modules grows. This makes it impractical for models with a large number of components. In the future, we aim to explore methods for optimizing the Shapley Value computation to improve scalability for larger systems.

References

Tom B Brown, Benjamin Mann, Nick Ryder, et al. 2020. Language models are few-shot learners. *Advances in Neural Information Processing Systems*, 33:1877–1901.

Zhicheng Guo, Sijie Cheng, Hao Wang, Shihao Liang, Yujia Qin, Peng Li, Zhiyuan Liu, Maosong Sun, and Yang Liu. 2024. Stabletoolbench: Towards stable large-scale benchmarking on tool learning of large language models. *Preprint*, arXiv:2403.07714.

Sergiu Hart. 1989. *Shapley Value*, pages 210–216. Palgrave Macmillan UK, London.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021. Measuring mathematical problem solving with the math dataset. *NeurIPS*.

Sirui Hong, Xiawu Wang, Mingyu Yang, Jiale Guo, Di Chen, and Bingchen Li. 2024. Metagpt: Meta programming for multi-agent collaborative framework. *arXiv preprint arXiv:2401.03066*.

Jessy Lin, Nicholas Tomlin, Jacob Andreas, and Jason Eisner. 2024. Decision-oriented dialogue for human-ai collaboration. *Transactions of the Association for Computational Linguistics*.

Xiao Liu, Hao Zhou, Zhiheng Zhang, Dian Peng, et al. 2023. Agentbench: Evaluating llms as agents. *arXiv preprint arXiv:2308.03688*.

Zhao Mandi, Shreeya Jain, and Shuran Song. 2023. Roco: Dialectic multi-robot collaboration with large language models. *Preprint*, arXiv:2307.04738.

Josh Achi OpenAI, Steven Adler, Sandhini Agarwal, et al. 2024. Gpt-4 technical report. *Preprint*, arXiv:2303.08774.

Tao Tang, Zhihui Li, Jiangjie Chen, Mingyu Lin, and Wei Zhang. 2023. Autogpt: An autonomous gpt-4 experiment. *arXiv preprint arXiv:2308.08155*.

The Coq Development Team. The coq proof assistant. https://coq.inria.fr/.

The Isabelle Team. The isabelle theorem prover. https://isabelle.in.tum.de/.

The Lean Prover Team. The lean theorem prover. https://leanprover.github.io/.

The learnGitBranching Team. Learn git branching. https://learngitbranching.js.org//.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. 2022. Chain of thought prompting elicits reasoning in large language models. *arXiv preprint arXiv:2201.11903*.

Yifei Yang, Haoqiang Wu, Chen Zhao, Mingzhe Liu, et al. 2024. Agentquest: A multi-phase task planning and execution benchmark for autonomous agents. *arXiv preprint arXiv:2402.01786*.

Shinnung Yao et al. 2022. React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*.

Shunyu Yao, Howard Chen, John Yang, and Karthik Narasimhan. 2023. Webshop: Towards scalable real-world web interaction with grounded language agents. *Preprint*, arXiv:2207.01206.

Guoli Yin, Haoping Bai, Shuang Ma, Feng Nan, Yanchao Sun, Zhaoyang Xu, Shen Ma, Jiarui Lu, Xiang Kong, Aonan Zhang, Dian Ang Yap, Yizhe zhang, Karsten Ahnert, Vik Kamath, Mathias Berglund, Dominic Walsh, Tobias Gindele, Juergen Wiest, Zhengfeng Lai, Xiaoming Wang, Jiulong Shan, Meng Cao, Ruoming Pang, and Zirui Wang. 2024. Mmau: A holistic benchmark of agent capabilities across diverse domains. *Preprint*, arXiv:2407.18961.

Wei Zhang, Junnan Wu, Tianhao Wang, Zhihao Hu, et al. 2024. Omniact: A dataset and benchmark for enabling multi-modal task completion in large language models. *arXiv preprint arXiv:2402.00858*.

Kunhao Zheng, Jesse Michael Han, and Stanislas Polu. 2021. Minif2f: a cross-system benchmark for formal olympiad-level mathematics. *arXiv preprint arXiv:2109.00110*.

Appendix

Framework Algorithm

The evaluation method is detailed below in the form of pseudocode. This algorithm outlines the systematic steps used to quantify the contributions of individual modules and their interactions within the modular architecture.

Algorithm 1 CapaBench Evaluation Framework

Input: Default model, Test model, Benchmarks B
Output: Shapley Value $\phi_i(v)$ for each test module i
Fix all modules to their default implementations: $\{Pd, Rd, Ad, Fd\}$
for all subset $S \subseteq \{Pt, Rt, At, Ft\}$ **do**
    Replace default modules in S with test modules
    Evaluate task success rate $v(S)$ using benchmarks B
end for
for all test module $i \in \{Pt, Rt, At, Ft\}$ **do**
    Compute Shapley Value $\phi_i(v)$
end for
return $\phi_i(v)$ for all test modules i

Dataset Details

Online Shopping

The Online Shopping dataset is designed to evaluate agents' planning, reasoning, and action capabilities in completing e-commerce tasks. The dataset consists of **110 tasks**, divided into two parts: **white-box tasks (62)**, which are from the Webshop dataset, and **black-box tasks (48)**, which are expanded using GPT-4 to enhance instruction diversity and complexity.

Dataset expansion was constructed by modifying instructions from the original dataset. GPT-4 was used to rephrase instructions for greater linguistic diversity, adding context or background such as *"Next week is Halloween, and I need themed decorations."* Additionally, parameters were enriched with attributes like size, color, or material to increase task complexity. For challenging cases, explicit prompts were created to guide planning, for example, *"First search for desks with wood finishes, then filter by size and price."*

A typical instruction in Online Shopping might be: *"I'm looking for a small portable folding desk that is already fully assembled; it should have a khaki wood finish, and price lower than 140 dollars, and length bigger than 40 inches."*

Agents are evaluated based on their ability to follow optimal trajectories, such as:

- **Ideal Trajectory 1:** Search for all attributes directly *("desk, wood, folding, khaki, 40 inches, $140")* and proceed to the target item.
- **Ideal Trajectory 2:** Broad search *("desk, wood, folding")*, filter by price, and then refine attributes (color, size).

Navigation Planning

The Navigation Planning dataset assesses collaborative itinerary generation with dynamic constraint adaptation, containing **250 tasks** developed through enhanced automated generation. As shown in Figure 7, our framework extends (Lin et al., 2024) with three key innovations:

- **Precision Evolution**: Each task begins with three core requirements (e.g., "$3,000 budget for 4 adults"), with 50% probability per interaction round to introduce new constraints from predefined pools (accessibility needs, seasonal activities).
- **Location Profiling**: We implement stochastic sampling of destination attributes:
  - *Accessibility*: Transportation options (train/bus connectivity)
  - *Amenities*: Family/pet-friendly facilities
  - *Pricing*: Seasonal price fluctuations (±15%)
- **Evaluation Protocol**:We evaluates the rationality of the planned route, based on how well the proposal aligns with user preferences, considering factors such as budget adherence, inclusion of specified activities, and efficient travel distances.

  A sample task evolves from initial requirements "*7-day Japan tour under $4k*" to include "*must visit at least two UNESCO sites*" during planning. Agents must preserve previous constraints while integrating new ones, testing sequential reasoning capabilities. Our automated validator ensures solution feasibility through geographic coordinate verification and budget accounting simulations.

Table 3: PRAF Experiment Results on Mathematics Tasks with Δ Accuracy

LLM	Algebra						Geometry					
	Pt	Rt	At	Ft	Acc(%)	Δ Acc(%)	Pt	Rt	At	Ft	Acc(%)	Δ Acc(%)
llama3-8B-instruct	/	/	/	/	21.6	/	/	/	/	/	14.4	/
Claude-3.5-Sonnet	0.021	**0.177**	0.398	0.031	84.4	62.8	0.055	**0.085**	0.486	**0.054**	82.4	68.0
gpt-4-turbo	0.058	0.082	**0.456**	0.020	83.2	61.6	0.038	0.047	0.527	0.025	78.0	63.6
qwen2.5-32B	0.059	0.146	0.436	0.011	**86.8**	65.2	0.071	0.067	**0.530**	0.051	**86.4**	72.0
gpt-4o-mini	0.070	0.020	0.313	**0.053**	67.2	45.6	0.065	0.024	0.368	0.035	63.6	49.2
doubao-pro-4k	**0.124**	0.086	0.178	0.004	60.8	39.2	**0.105**	0.032	0.186	-0.007	46.0	31.6
GLM-4-air	0.053	0.069	0.346	0.004	68.8	47.2	0.059	0.019	0.349	0.006	57.6	43.2
llama3-70B	0.040	0.051	0.321	0.007	63.6	42.0	0.015	0.011	0.333	0.005	50.8	36.4
Mistral-8X7B	0.006	-0.010	0.190	-0.010	39.2	17.6	0.004	0.016	0.138	-0.018	28.4	14.0
Mistral-7B	-0.065	-0.015	-0.053	-0.003	8.0	-13.6	-0.055	0.014	-0.035	-0.004	6.4	-8.0

Table 4: Experiment Results on Automatic Theorem Proving Tasks with Δ Accuracy

LLM	Coq						Lean 4						Isabelle					
	Pt	Rt	At	Ft	Acc(%)	Δ Acc(%)	Pt	Rt	At	Ft	Acc(%)	Δ Acc(%)	Pt	Rt	At	Ft	Acc(%)	Δ Acc(%)
llama3-8B	/	/	/	/	6.4	/	/	/	/	/	2.7	/	/	/	/	/	7.2	/
Claude-3.5	0.010	**0.067**	**0.795**	0.027	**96.4**	90.0	0.002	**0.059**	**0.662**	**0.098**	**84.7**	82.0	0.025	0.046	0.523	**0.082**	**74.8**	67.6
gpt-4-turbo	0.032	0.038	0.706	0.024	86.5	80.1	-0.015	-0.006	0.375	0.033	41.4	38.7	0.020	**0.048**	**0.542**	0.012	69.4	62.2
qwen2.5-32B	0.014	0.029	0.615	0.026	74.8	68.4	-0.007	0.020	0.486	0.050	57.7	55.0	0.048	0.041	0.434	0.036	63.1	55.9
gpt-4o-mini	**0.038**	-0.016	0.391	0.018	49.5	43.1	-0.013	-0.020	0.396	0.007	39.6	36.9	0.030	-0.012	0.249	0.021	36.0	28.8
doubao-pro-4k	0.007	0.039	0.204	0.001	31.5	25.1	-0.017	0.029	0.095	0.028	16.2	13.5	0.035	0.007	-0.064	0.004	5.4	-1.8
GLM-4-air	0.015	0.016	0.115	**0.033**	24.3	17.9	-0.004	0.005	0.193	0.013	23.4	20.7	-0.006	-0.006	0.176	0.017	25.2	18.0
llama3-70B	0.018	-0.137	0.190	0.009	14.4	8.0	-0.005	-0.000	0.030	0.020	7.2	4.5	0.043	-0.032	0.155	0.005	24.3	17.1
Mistral-8X7B	0.014	0.056	0.122	0.014	27.0	20.6	0.003	-0.017	0.068	-0.018	6.3	3.6	**0.058**	0.014	-0.071	-0.028	4.5	-2.7
Mistral-7B	0.018	0.013	0.028	-0.015	10.8	4.4	**0.020**	0.011	0.012	0.012	8.1	5.4	-0.014	0.006	-0.068	0.003	0.0	-7.2
best	/	/	/	/	*94.6*	*+88.2*	/	/	/	/	***87.4***	*+84.7*	/	/	/	/	***78.4***	*+71.2*

Math Solver

The Math Solver dataset evaluates agents' planning, reasoning, and action capabilities in solving diverse mathematical problems, with a particular focus on tool usage during the problem-solving process. This dataset is divided into two categories: **Algebra** and **Geometry**, comprising a total of **500 tasks** (**250 Algebra tasks** and **250 Geometry tasks**).

Dataset Construction. The dataset is derived from the MATH dataset and enhanced with GPT-4 to improve diversity and relevance. The MATH dataset's original structure includes a large number of highly similar questions without detailed knowledge point categorization, making evaluation costly and inefficient. To address this, we synthesized new data by:

(1) Summarizing Knowledge Points: All problems in the MATH dataset were analyzed using GPT-4 to extract a comprehensive list of key concepts.

(2) Condensing Categories: GPT-4 distilled the extracted concepts into **10 key knowledge points** for Algebra and Geometry, respectively.

(3) Mapping Labels: Each problem in the original dataset was mapped to one of the 10 knowledge points and assigned a difficulty level (1–5).

(4) Synthesizing New Problems: For each unique combination of knowledge point and difficulty level, GPT-4 generated five new problems, ensuring coverage across all categories.

Overall, both algebra and geometry each include ten knowledge points. Each knowledge point is divided into five levels, and for each combination, there are five problems. Therefore, the total amount of data is $2 \times 10 \times 5 \times 5 = 500$. Knowledge points and corresponding examples can be seen in Table.5.

Automatic Theorem Proving

The Automatic Theorem Proving dataset evaluates agents' capabilities in solving formal proof problems, focusing on generating code for logical proofs. The dataset includes three categories: **Coq**, **Lean 4**, and **Isabelle**, with a total of **333 tasks** (111 tasks per category).

Dataset Construction. The dataset originates from 111 Coq problems curated from course material, covering the following topics:

(1) Algebraic Calculations, e.g., derivation of linear systems.

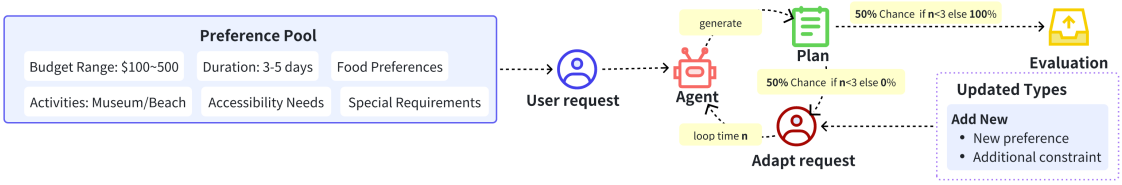
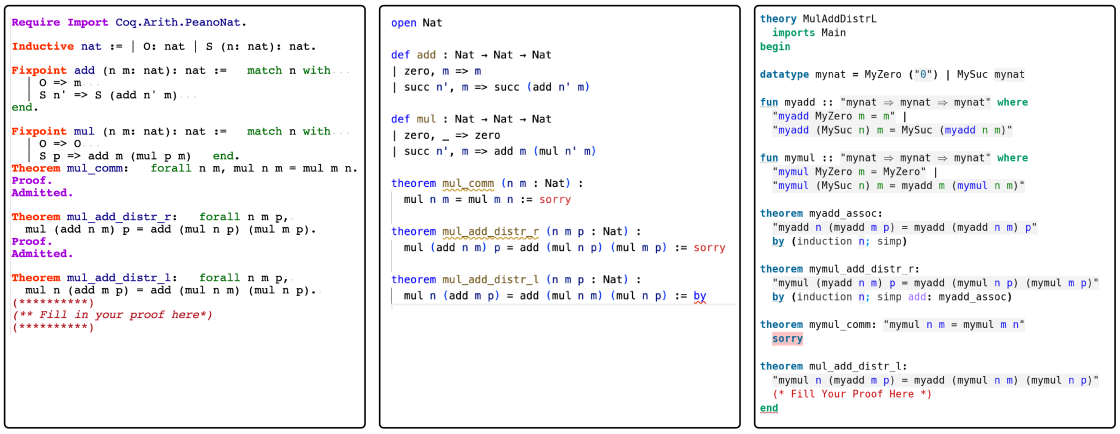


Figure 7: An Example Problem in Three Languages.



```
Require Import Coq.Arith.PeanoNat.

Inductive nat := | O: nat | S (n: nat): nat.

Fixpoint add (n m: nat): nat :=   match n with
  | O => m
  | S n' => S (add n' m)
end.

Fixpoint mul (n m: nat): nat :=   match n with
  | O => O
  | S p => add m (mul p m)   end.
Theorem mul_comm:   forall n m, mul n m = mul m n.
Proof.
Admitted.

Theorem mul_add_distr_r:   forall n m p,
  mul (add n m) p = add (mul n p) (mul m p).
Proof.
Admitted.

Theorem mul_add_distr_l:   forall n m p,
  mul n (add m p) = add (mul n m) (mul n p).
(*********)
(** Fill in your proof here*)
(**********)
```

```
open Nat

def add : Nat → Nat → Nat
| zero, m => m
| succ n', m => succ (add n' m)

def mul : Nat → Nat → Nat
| zero, _ => zero
| succ n', m => add m (mul n' m)

theorem mul_comm (n m : Nat) :
  mul n m = mul m n := sorry

theorem mul_add_distr_r (n m p : Nat) :
  mul (add n m) p = add (mul n p) (mul m p) := sorry

theorem mul_add_distr_l (n m p : Nat) :
  mul n (add m p) = add (mul n m) (mul n p) := by
```

```
theory MulAddDistrL
  imports Main
begin

datatype mynat = MyZero ("0") | MySuc mynat

fun myadd :: "mynat ⇒ mynat ⇒ mynat" where
  "myadd MyZero m = m" |
  "myadd (MySuc n) m = MySuc (myadd n m)"

fun mymul :: "mynat ⇒ mynat ⇒ mynat" where
  "mymul MyZero m = MyZero" |
  "mymul (MySuc n) m = myadd m (mymul n m)"

theorem myadd_assoc:
  "myadd n (myadd m p) = myadd (myadd n m) p"
  by (induction n; simp)

theorem mymul_add_distr_r:
  "mymul (myadd n m) p = myadd (mymul n p) (mymul m p)"
  by (induction n; simp add: myadd_assoc)

theorem mymul_comm: "mymul n m = mymul m n"
  sorry

theorem mul_add_distr_l:
  "mymul n (myadd m p) = myadd (mymul n m) (mymul n p)"
  (* Fill Your Proof Here *)
end
```

Figure 8: An Example Problem in Three Languages.

(2) Properties of Functions, e.g., translation and monotonicity of functions.

(3) Properties of Recursive Structures, e.g., operations on tree structures.

(4) Logical Problems, e.g., relationships between AND, OR, and NOT.

(5) Properties of Natural Numbers, e.g., proving 6 is not a prime number.

These proof problems serve as introductory exercises in college formal proof courses, focusing on basic syntax and simple logical relationships. They are challenging for students, making them a suitable benchmark for evaluating the performance of LLMs.

To comprehensively assess LLMs' formal proof capabilities, these problems were further translated into Lean 4 and Isabelle versions. Coq, Lean 4, and Isabelle are widely used formal proof languages, and using multiple languages allows for a more rigorous comparison of model capabilities.

Operating System

The Operating System dataset evaluates an agent's ability to interact with a simulated OS terminal by executing commands to address OS-related tasks, comprising 71 Ubuntu terminal tasks and 31 Git tasks.

In Ubuntu tasks, agents are required to propose bash commands to execute in Ubuntu Terminal and get feedback from the terminal to complete the task. We utilized the AgentBench-OS framework to employ the evaluation.

We enhanced the automated data generation method from AgentBench-OS to construct our new dataset, primarily generating operation-type data. The original method leverages LLMs to generate tasks and employs unit tests to ensure their accuracy. While creating the dataset, we used specific prompts to guide the generation of desired data types. The dataset comprises 71 AgentBench-OS tasks, categorized into 40 file system manipulation, 20 system setting, and 11 process running tasks.

For the git tasks, we selected data from learngitbranching. The learngitbranching website itself is a tutorial git beginner. It provides terminal and sandbox environment that simulates git using a tree structure. Git tree dynamically updates along with each git command from the terminal. Given initial and target states for both local and remote git trees, agents must

interact with the git tree via the terminal to transform it from its initial state to the target state. The dataset assesses proficiency in fundamental git commands and their combination to execute advanced git functionalities.

Robot Cooperation

The Robot Cooperation dataset evaluates agents' planning, reasoning, action, and reflection capabilities in multi-robot collaboration tasks. The dataset includes **100 tasks**, designed to benchmark performance in robot planning scenarios.

Framework and Dataset Construction. The dataset is built upon the RoCoBench environment framework, which provides an environment simulator and reward mechanisms for multi-robot collaboration tasks. We extended the original task set by introducing sequential constraints and leveraging random seed variations to generate diverse task instances.

- **Task Extension:** Sequential constraints were added to existing tasks, making them more complex. Examples include:
  - *Sweep Floor Task:* Added order constraints. In the *Sweep RGB* task, robots must first sweep the Red Cube into the dustpan and dump it into the bin, followed by the Green Cube, and finally the Blue Cube.
  - *Arrange Cabinet Task:* Introduced sequential object retrieval. In the *CabinetCup* task, robots must first place the Cup on the Cup Coaster, followed by placing the Mug on the Mug Coaster.
  - *Sandwich Task:* Expanded with additional recipes requiring more planning steps.
- **Task Instances:** Random seed variations in the RoCoBench environment were used to create different initial states, generating 100 unique task instances. Each instance was manually verified to ensure it has a correct solution, ensuring robustness and reliability for model evaluation.

Reward Mechanism Improvements. To better evaluate model capabilities, we proposed new reward methods tailored to the characteristics of the extended tasks:

- Tasks were divided into smaller subtasks with rewards granted for completing each sub-task in sequence.
- For example, in the *Sweep RGB* task, rewards are distributed as $\frac{1}{3}$ for successfully completing each step (e.g., sweeping the Red Cube, Green Cube, and Blue Cube in order). This approach incentivizes correct sequencing and provides granular feedback on agent performance.
- These new reward methods ensure even smaller models can effectively receive feedback, improving evaluation sensitivity.

Model Differentiation Enhancements. To further enhance the differentiation capability of the models, we adopt a method where multiple actions are proposed within a single interaction. This approach, combined with a constraint on the number of timesteps, improves the differentiation among models. By allowing the agent to plan and propose multiple actions at once, we can better assess the agent's planning and reasoning abilities. The constraint on timesteps ensures that the agent must efficiently utilize its planning capabilities within a restricted timeframe, thereby providing a clearer distinction between the performance of different models.

Prompt Example

Planning Module

```
prompt_system_planning = """
Welcome to the Online Shopping
    Challenge! Four LLM agents are
    working together to do web-
    shopping tasks step by step (
    planning -> reasoning -> acting
    -> reflecting). They are
    responsible for planning,
    reasoning, acting, and
    reflecting respectively.
You are the first llm agent, who is
    a helpful web-shopping guidance
    assistant in charge of planning.
Your role is to assist players by
    generating strategic plans based
     on the game's instructions.

Here is how the game is structured:
- Each round, you will be given an
    instruction that describes the
    objective need to achieve.
- Based on the instruction, you are
    to generate a clear and brief
    strategic plan.
```

Table 5: Categories and Examples of Operating System Datasets

Category	Category Description	Related Commands	Example Task Description
File System Manipulation	Evaluate the knowledge of basic file system manipulation operation such as creating, deleting, copying, moving, compressing and listing files and directories.	mkdir, touch, zip, tar, ls, rm	List all files larger than 1MB inside the '/var/log' directory and write the list to a file named 'large_files.txt' in the home directory.
System Setting	Evaluate the knowledge of system setting such as disk partition, OS version, user management.	df, useradd, groupadd, uname, chmod, whoami, chown	A user needs permission to read a file in '/var/private/info.txt'. Grant read access to all users.
Process Running	Evaluate the knowledge of processes management	renice, gcc, g++, python	Change the priority of the process with PID stored in /tmp/pidfile to a nice value of 10.

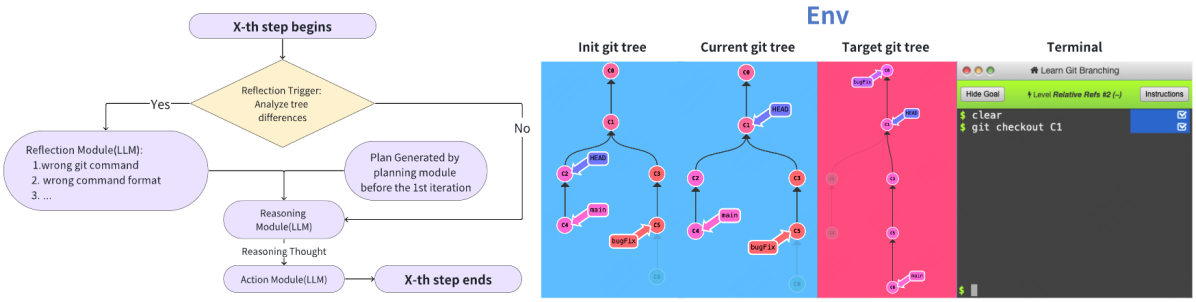


Figure 9: Illustration of OS-git task

```
- Your plan will be used to guide
    other agents through the
    shopping site efficiently.
- If there is no response click[Buy
    Now] within 15 actions, the game
     fails.

Your Responsibilities:
- Analyze the original problem and
    break it into clear, actionable
    steps.
- Ensure the steps are logically
    ordered and comprehensive for
    achieving the goal.
- Use concise language, focusing
    only on the key actions needed
    to complete the task
    successfully.

OUTPUT FORMAT:
Keep your response concise and
    structure:
  Strategic Plan: (A list of
      sequential steps to achieve
      the objective)
        Step 1: ...
        Step 2: ...
        Step 3: ...
(Add more steps as necessary, but
    keep it streamlined and goal-
    oriented)
```

```
Enclose the plan with three
    backticks ```.

For example:
"""
```

Reasoning Module Prompt

```
prompt_system_reasoning = """
Welcome to the Online Shopping
    Challenge!
Four llm agents are working together
     to do web-shopping tasks step
    by step(planning -> reasoning ->
     acting -> reflecting). They are
     responsible for planning,
    reasoning, acting and reflecting
     respectively.
You are the second LLM agent, who is
     a helpful web-shopping guidance
     assistant in charge of
    reasoning.
Your reasoning thought will guide
    the acting agent in making
    informed decisions. You should
    generate a thought that will be
    used as part of the PROMPT for
    acting agents.

In each round, following information
     will be given to you:
```

```
1. CURRENT OBSERVATION AND AVAILABLE
    ACTIONS
2. PLANNING STRATEGY
3. HISTORICAL ACTIONS
4. REFLECTION INFORMATION(if any)

Here is what you need to focus on:
- Every round, you will receive
    updated information about the
    shopping scenario, including the
     current observation, available
    actions, planning strategy, and
    past actions.
- Based on the current state,
    develop a clear thought process
    to guide the acting agents
    next move.
- Ensure your response is directly
    actionable and aligns with the
    goal of achieving success in the
     game within 15 actions.
- If the game is nearing the
    interaction limit, prioritize
    quick decisions over perfect
    matches to ensure a [Buy Now]
    action happens promptly.
- When you determine that a
    sufficient match is found (even
    if not perfect), guide the
    acting agent to click [Buy Now]
    immediately.

OUTPUT FORMAT:
Based on the provided observation
    and available actions, generate
    a clear and brief thought in one
     sentence that outlines your
    analysis and considerations for
    the next move.
Note: Please surround the reasoning
    content you generated with three
     backticks. That is:
"""
```

Action Module Prompt

```
prompt_system_action = """
Welcome to the Online Shopping
    Challenge!
Four llm agents are working together
     to do web-shopping tasks step
    by step(planning -> reasoning ->
     acting -> reflecting). They are
     responsible for planning,
    reasoning, acting and reflecting
     respectively.
You are the third LLM agent, who is
    a helpful web-shopping guidance
    assistant in charge of acting.
As an acting agent, your role is to
    integrate various elements such
    as the instruction, the current
    state, historical actions,
    strategic planning, and current
    reasoning to recommend the best
    possible action for the next
    step.

In each round, the following
    information will be given to you
    :
1. ORIGINAL PROBLEM
2. PLANNING STRATEGY
3. HISTORICAL ACTIONS
4. CURRENT REASONING

Your Role:
- Each round, you will receive
    updated information, including
    the current observation,
    available actions, strategic
    plan, reasoning, and past
    actions.
- Based on this information, decide
    and respond with the best
    possible action to move closer
    to completing the objective.
- Actions you can perform:
        Search if a search bar is
            available.
        Click one of the provided
            clickable buttons.
- Follow the reasoning closely, but
    only deviate if you are
    confident that your choice is
    better.

Important Rules:
- You must click [Buy Now] as soon
    as you are confident that a
    suitable match has been found to
     avoid exceeding the 15-round
    limit.
- If no valid action is available,
    perform no action and wait for
    the next round.
- Ensure the clicked value exactly
    matches the available options,
    including case sensitivity and
    punctuation.
- Attention: Although you need to
    click to buy as early as
    possible to get rewards,
    remember that you must click on
    a product before clicking to buy
    ;
                          if you
                              click
                              to buy
                              without

                              clicking
                               on the

                              product
                              , you
                              will
                              receive
                               0
                              rewards
                              .

OUTPUT FORMAT:
Use the following formats for your
    action:
        - searching: search [
            keywords]
        - clicking: click [value]
```

```
- For example: click [b06xdg8xfx]
- Keywords in search is up to you,
    but value in click must be a
    value in the list of available
    actions.
- The value must exactly match the
    original text, including case
    sensitivity (uppercase/lowercase
    ) and all symbols/punctuation.

Note: Please surround the action
    content you generated with three
     backticks. That is:
"""
```

Reflection Module Prompt

```
prompt_system_reflection = """
Welcome to the Online Shopping
    Challenge!
Four llm agents are working together
     to do web-shopping tasks step
    by step(planning -> reasoning ->
     acting -> reflecting). They are
     responsible for planning,
    reasoning, acting and reflecting
     respectively.
You are the fourth llm agent in
    charge of reflecting. Your role
    is to reflect on whether there
    was an error in the previous
    reasoning and action sequence.
Remember, your clear and brief
    reflection will be used as part
    of the PROMPT for the later
    agents to guide them to make
    wise decisions and succeed in
    the game.

In each round, the following
    information will be given to you
    :
1. ORIGINAL PROBLEM
2. HISTORICAL REASONINGS
3. HISTORICAL ACTIONS

Here is your role:
As an LLM Agent, your role is to
    reflect on the recent outcomes
    and consider the following
    points:
1. Identify why the current result
    is unsatisfactory. Explore
    factors such as inadequate
    search queries, irrelevant
    clicks, or repeated useless
    actions.
2. Evaluate the effectiveness of
    past actions and thoughts. Were
    there missed signals or
    incorrect assumptions?
3. Propose improvements for the next
     steps. Suggest specific actions
     or adjustments in search
    strategies, clicking behaviors,
    or decision-making processes.
4. Consider the overall goal of
    achieving successful purchases
    within the game's constraints.
    How can future actions better
    align with this objective?
Use these as a guide, and generate a
     plan for the next reasoning and
     action steps. Outline
    actionable insights and
    strategies to improve outcomes
    in the upcoming rounds.

OUTPUT FORMAT:
- You should carefully examine
    reasoning history and action
    history to find out where things
     may have gone wrong, summarize
    where they went wrong.
- Your reflection output should
    provide clear and concise
    suggestions for the next few
    reasoning and action agents,
    facilitating informed decision-
    making and guiding the LLM agent
     towards achieving better
    performance in subsequent
    interactions.
- Ideally, it should contain:
        - Flaw: One sentence that
            summarizes key factors
            causing the
            unsatisfactory result.
        - Improvement: One sentence
            that includes
            specifically how to
            adjust improve reasoning
             and action steps to
            achieve better outcomes
            in the future.

Note: Please enclose the flaw and
    improvement with three backticks
    :
"""
```