Attribution-Based Analysis and Optimization of Modular Agentic Workflows

Yingxuan Yang[1] **Bo Huang**[1] **Siyuan Qi**[1] **Chao Feng**[1] **Haoyi Hu**[1]
Yuxuan Zhu[2] **Jinbo Hu**[1] **Haoran Zhao**[1] **Ziyi He**[3] **Xiao Liu**[4] **Zongyu Wang**[4]
Muning Wen[1] **Lin Qiu**[4] **Xuezhi Cao**[4] **Xunliang Cai**[4] **Yong Yu**[1] **Weinan Zhang**[1*]
[1]Shanghai Jiao Tong University [2]University of Chicago [3]University of Toronto [4]Meituan
{zoeyyx, wnzhang}@sjtu.edu.cn

Abstract

Agentic workflows solve complex tasks by orchestrating modular components (e.g., planning, reasoning, action, reflection) built on top of LLM backbones. A practical but underexplored question is *model allocation*: given a fixed workflow decomposition and a pool of candidate LLMs, which components should be upgraded (and with which models) to upgrade task performance, and how can we attribute gains to individual upgrades and their interactions? We present **ShapleyFlow**, a cooperative game theoretic framework that models component upgrades as players and evaluates component coalitions to compute Shapley values. This yields interaction-aware attribution and supports Shapley-guided configuration recommendation for model allocation under a fixed workflow structure. We further introduce **CapaBench**, a benchmark of 1,500+ tasks across seven domains (shopping, navigation, ticketing, mathematics, operating systems, robotic coordination, and automated theorem proving). Across 9 representative LLMs and all 2^4 upgrade coalitions in a 4-component workflow, ShapleyFlow provides (i) principled, interaction-aware attribution for modular workflows and (ii) actionable model-allocation recommendations that improve over strong single-model baselines.

Introduction

Large Language Models (LLMs) (Brown et al., 2020; OpenAI et al., 2024) have driven the rise of agentic workflows, where complex tasks are decomposed into orchestrated sequences of specialized components (Sapkota et al., 2025; Acharya et al., 2025). Frameworks such as ReAct (Yao et al., 2023b) and AutoGPT (Gravitas, 2023) show that structured orchestration can improve both performance and interpretability. These workflows typically combine planning, reasoning, action execution, and reflection (Wei et al., 2022; Shinn et al., 2023; Renze and Guven, 2024) to address diverse challenges including code generation, web navigation, and autonomous control (Zhou et al., 2024; Yang et al., 2024b; Chai et al., 2025).

Despite their widespread adoption, a key challenge remains: *how can we attribute performance to individual workflow components and use this understanding to make principled design decisions?* Existing evaluation methods (Liu et al., 2023; Chiang et al., 2024; Guo et al., 2024; Yin et al., 2024) primarily report end-to-end outcomes and provide limited visibility into the internal dynamics of multi-component workflows. This black-box evaluation obscures interdependencies between components and offers little actionable guidance for workflow design.

Current practice has three concrete limitations. First, end-to-end success rates do not isolate the **marginal contribution** of each component under different configurations. Second, standard one-factor-at-a-time ablations fail to capture **interaction effects** and can miss beneficial combinations because component utility is often non-additive. Third, end-to-end success alone does not tell us where to invest stronger models or more compute within the workflow, leaving it unclear which components to upgrade first.

To address these challenges, we introduce **ShapleyFlow**, a cooperative-game-theoretic framework for *interaction-aware attribution* in modular agentic workflows. Specifically, ShapleyFlow computes Shapley values (Shapley, 1952; Hart, 1989; Cohen et al., 2007; Lundberg and Lee, 2017; Ghorbani and Zou, 2019) over all component coalitions to quantify both (i) each component's marginal contribution and (ii) non-additive interaction effects among components. This enables principled, fine-grained analysis of *which components matter* and *which combinations work well* for different task

*Corresponding author.

Findings of the Association for Computational Linguistics: ACL 2026, pages 7247–7264
July 2-7, 2026

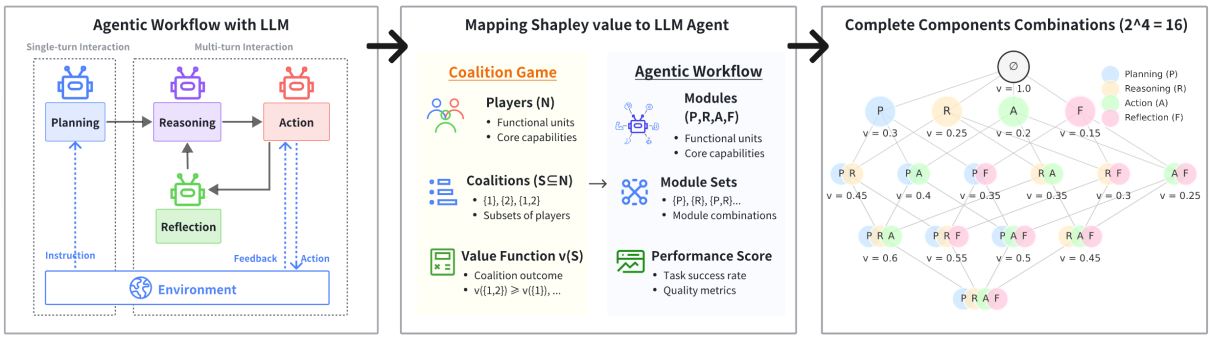


Figure 1: ShapleyFlow for interaction-aware attribution and configuration recommendation in modular agentic workflows. The left panel shows a four-component workflow (Planning, Reasoning, Action, Reflection). The middle panel illustrates our cooperative-game formulation, where component *upgrades* are modeled as players and coalition outcomes map to task performance. The right panel shows exhaustive evaluation over all $2^4 = 16$ upgrade coalitions, enabling Shapley-value attribution and model-allocation guidance.

types.

In this paper, we study *model allocation* in modular agentic workflows: given a fixed workflow structure with multiple functional components (e.g., Planning, Reasoning, Action, and Reflection)(Brown et al., 2020; Yao et al., 2023b; Wei et al., 2022; Gravitas, 2023; Shinn et al., 2023; Hong et al., 2023; Guo et al., 2024; Yin et al., 2024) and a pool of candidate LLM backbones, which components should be upgraded, and with which models, to most effectively improve task performance? Crucially, this problem is inherently *interaction-dependent*: the benefit of upgrading a component often depends on which other components are upgraded alongside it, making additive assumptions and one-factor-at-a-time analyses insufficient.

We formalize this problem as an *upgrade coalition game*, where workflow components act as players and each coalition corresponds to a concrete model assignment obtained by upgrading a subset of components from a shared baseline. Within this formulation, Shapley values provide a principled mechanism to attribute performance gains to individual upgrades while accounting for all interaction patterns among components. Unlike prior attribution approaches that focus on input features, neural submodules, or isolated ablations, our formulation directly targets *model allocation decisions* in agentic workflows and grounds attribution in executable end-to-end configurations.

To our knowledge, ShapleyFlow is among the first to cast *model allocation in modular agentic workflows* as a cooperative game over *component upgrades*: (i) upgrades are treated as players tied to executable end-to-end configurations, (ii) exhaustive coalition evaluation is used to faithfully capture non-additive interactions, and (iii) the resulting Shapley attributions are directly leveraged for *configuration recommendation* within a fixed workflow structure. ShapleyFlow does not propose a new Shapley estimator. Instead, our contribution is the formulation and executable evaluation protocol that instantiates Shapley-value attribution over workflow component upgrades the object of interest for real-world model-allocation decisions.

To empirically study this setting, we introduce **CapaBench**, a benchmark of 1,500+ tasks spanning seven diverse domains. Across 9 representative LLM backbones and exhaustive evaluation of all 2^4 upgrade configurations, ShapleyFlow yields interaction-aware component attributions, reveals systematic cross-domain interaction patterns, and supports Shapley-guided configuration recommendations that are strongest within the evaluated model pool.

We summarize our contributions as follows:

- **Upgrade-coalition formulation for model allocation.** We formalize model assignment in agentic workflows as an *upgrade coalition game*, where players are *component upgrade decisions* and each coalition corresponds to an *executable end-to-end configuration* under a fixed workflow.
- **Executable Shapley attribution protocol.** We provide a practical evaluation protocol that computes Shapley attributions over component upgrades, enabling principled analysis of marginal contributions and non-additive interactions for modular workflows.
- **Benchmark and validated configuration recommendation.** We introduce *CapaBench* (1,500+ tasks across 7 domains) and show that Shapley attributions serve as effective signals

for *configuration recommendation* within a fixed workflow.

Related Work

Agentic Workflow Systems

Agentic workflows have shifted AI system design from monolithic models to orchestrated multi-component architectures. ReAct (Yao et al., 2023b) demonstrated the effectiveness of structured reasoning–action cycles, and AutoGPT (Gravitas, 2023) advanced this paradigm through iterative planning, tool use, and self-reflection. MetaGPT (Hong et al., 2023) further refined workflow orchestration with hierarchical planning and role-based coordination. These frameworks establish effective workflow patterns but primarily emphasize *construction* and *execution*, providing limited tools for *attribution* and principled *model allocation* across components. Recent work explores automated agentic system optimization, including prompt- and topology-level improvements for multi-agent systems (Zhou et al., 2025; Hu et al., 2025). These approaches focus on end-to-end performance and do not provide component-level attribution or interaction analysis.

Workflow Analysis and Evaluation

Evaluation frameworks for agentic systems have evolved from task-centric benchmarks to broader capability assessments. AgentBench (Liu et al., 2023) evaluates diverse scenarios but largely reports end-to-end success without isolating component-level effects. MMAU (Yin et al., 2024) evaluates multiple skills yet still maps capabilities to tasks in a way that makes component-level attribution difficult. Recent benchmarks such as OmniACT (Zhang et al., 2024) and AgentQuest (Yang et al., 2024a) expand coverage but remain largely black-box with respect to workflow internals. Recent benchmarks evaluate agentic systems in more realistic and long-horizon settings and agentic workflow generation (Chan et al., 2025; Yang et al., 2025; Qiao et al., 2025). They primarily report end-to-end results and offer limited insight into component-level contributions. ShapleyFlow addresses this gap by attributing performance to component upgrades and quantifying interaction effects, enabling actionable guidance for model allocation and configuration recommendation.

Agentic Workflow Analysis Framework

We introduce **ShapleyFlow**, an executable protocol for interaction-aware attribution and configuration recommendation in modular agentic workflows. The Shapley value, introduced by (Shapley, 1952; Hart, 1989; Cohen et al., 2007; Lundberg and Lee, 2017; Ghorbani and Zou, 2019), provides a solution concept for cooperative games that fairly distributes the rewards among players according to their marginal contributions. ShapleyFlow instantiates cooperative-game attribution over *component upgrade decisions*, the relevant decision unit for *model allocation* under a fixed workflow decomposition.

Model allocation as a two-level problem. We study model allocation under a fixed workflow decomposition at two levels. **(1) Upgrade attribution.** For each candidate model $m \in \mathcal{M}$, we define an *upgrade-coalition game* relative to a shared baseline assignment: a coalition $S \subseteq N$ corresponds to an executable workflow where components in S are upgraded to use m while $N \setminus S$ remains at baseline. We evaluate all $2^{|N|}$ coalitions and compute exact Shapley values $\{\phi_i^{(m)}\}_{i \in N}$, which quantify the interaction-aware gain of upgrading component i *to model* m across all upgrade contexts. **(2) Allocation over a pool.** We then use the per-model Shapley profiles $\{\phi_i^{(m)}\}$ as a practical signal to recommend a mixed assignment from $\mathcal{M}$ (potentially choosing different models for different components). This second step is a *recommendation heuristic* built on the upgrade-attribution protocol, rather than a new Shapley estimator or a direct solution to a single multi-choice cooperative game.

Game-Theoretic Formulation: Upgrade Attribution

Players and executable coalitions. Let N denote the set of workflow components. We fix a workflow structure, component interfaces, and orchestration logic. A *baseline assignment* $\emptyset$ uses a baseline model for all components. An *upgrade coalition* $S \subseteq N$ is defined as an *executable* workflow configuration in which components in S are upgraded to a target implementation (or a chosen model), while the remaining components $N \setminus S$ stay at baseline. Thus, different coalitions differ only by *which components are upgraded*.

Algorithm 1 ShapleyFlow: Two-Level Protocol

Input: workflow components N; tasks T; baseline assignment b; model pool $\mathcal{M}$
Output: Shapley profiles $\{\phi_i^{(m)}\}$ and mixed assignment $\hat{a}$
for all $m \in \mathcal{M}$ **do**
  // Level 1: per-model upgrade attribution
  for all $S \subseteq N$ **do**
    Instantiate executable config $a_{m,S}$: use m on S, baseline b on $N \setminus S$
    $\mathrm{Perf}_m(S) \leftarrow \frac{1}{|T|} \sum_{t \in T} \mathbb{I}[\mathrm{success}(t, a_{m,S})]$
  end for
  $v_m(S) \leftarrow \mathrm{Perf}_m(S) - \mathrm{Perf}_m(\emptyset) \qquad (v_m(\emptyset) = 0)$
  Compute $\{\phi_i^{(m)}\}_{i \in N}$ from $\{v_m(S)\}_{S \subseteq N}$ via Eq. (3)
end for
// Level 2: pool-level recommendation (heuristic)
$m_i^\star \leftarrow \arg\max_{m \in \mathcal{M}} \phi_i^{(m)} \quad \forall i \in N$
$\hat{a}(i) \leftarrow m_i^\star \quad \forall i \in N$
return $\{\phi_i^{(m)}\}$ and $\hat{a}$

Performance and characteristic function. For a task set T, $u(t, S) \in [0, 1]$ denote the task-level utility of executing coalition S on task t. We define the end-to-end performance of coalition S as

$$\mathrm{Perf}(S) = \frac{1}{|T|} \sum_{t \in T} u(t, S). \tag{1}$$

where $\mathrm{success}(t, S)$ indicates whether the workflow configured by coalition S completes task t. We define the cooperative game $G = (N, v)$ using the *gain over baseline*:

$$v(S) = \mathrm{Perf}(S) - \mathrm{Perf}(\emptyset), \tag{2}$$

so that $v(\emptyset) = 0$ by construction. This choice ensures that attributions quantify *improvements relative to the baseline workflow*.

Shapley attribution over component upgrades. For the game $G = (N, v)$, the Shapley value for component $i \in N$ is

$$\phi_i(v) = \sum_{S \subseteq N \setminus \{i\}} \frac{|S|!(|N| - |S| - 1)!}{|N|!} \Big(v(S \cup \{i\}) - v(S) \Big). \tag{3}$$

The term $v(S \cup \{i\}) - v(S)$ is the marginal gain of upgrading component i in the context of coalition S, and the combinatorial weight averages this marginal gain over all possible upgrade orders. Under our definition of $v(S)$, $\phi_i(v)$ can be interpreted as the *expected performance gain over baseline* attributable to upgrading component i, while accounting for non-additive interactions with other upgraded components.

Evaluation Protocol

Algorithm 1 summarizes our two-level protocol. For each target model $m \in \mathcal{M}$, we exhaustively evaluate all upgrade coalitions $S \subseteq N$ to obtain $v_m(S)$ and compute per-model Shapley profiles $\phi_i^{(m)}$ (Level 1). We then construct a mixed assignment $\hat{a}$ using $\phi_i^{(m)}$ as a recommendation signal (Level 2), and report its end-to-end performance.

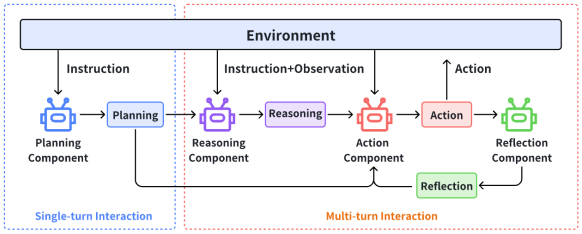


Figure 2: A 4-component agentic workflow with Planning (P), Reasoning (R), Action (A), and Reflection (F).

Workflow Instantiation

To demonstrate applicability, we instantiate ShapleyFlow on a four-component workflow: Planning (P), Reasoning (R), Action (A), and Reflection (F) (Brown et al., 2020; Yao et al., 2023b; Wei et al., 2022; Gravitas, 2023; Shinn et al., 2023; Hong et al., 2023; Guo et al., 2024; Yin et al., 2024).These components are fundamental for task decomposition, logical inference, action execution, and iterative improvement, following established patterns in agentic systems literature. While real-world systems may implement these functions through more specialized sub-components, we provide a representative case that balances analytical tractability with comprehensive coverage of key agentic capabilities.

For a new task, the workflow operates as follows:

- **Planning (P)**: initiates the process by decomposing the task into subtasks and analyzing required resources/conditions. Planning operates in single-turn mode, establishing the initial task structure.
- **Reasoning (R)**: receives the task, current observations, and planning output to analyze the next step progression.
- **Action (A)**: takes planning context and current reasoning analysis to generate properly formatted actions.
- **Reflection (F)**: is triggered based on task-specific conditions to analyze encountered problems, failure causes, and improvement suggestions. This feedback is integrated into subsequent reasoning prompts.

As illustrated in Figure 2, for each task, Planning operates in single-turn mode to propose a

Table 1: CapaBench Dataset Statistics

Category	**Shopping**		**Navigation**	**Ticket**	**Math**		**ATP**			**RobotCoop**	**OS**
Subcategory	Black	White	None	None	Algebra	Geometry	Coq	Lean4	Isabelle	None	None
Count	48	62	250	150	250	250	111	111	111	100	102

high-level plan, while Reasoning and Action follow a multi-turn interaction pattern (consistent with ReAct-style agents), and Reflection is triggered by task-specific conditions to provide feedback used in subsequent turns. This fixed interface ensures that all coalitions differ only by component implementation, making performance comparisons across coalitions semantically meaningful.

Benchmark Construction

To study *model allocation* at the component level, we construct **CapaBench** (**Capa**bility-level Assessment **Bench**mark), a benchmark of 1,500+ agentic workflows spanning seven domains. Table 1 reports the distribution of tasks across categories. Implementation details are provided in the supplementary material. We start from established task *scenarios* and systematically *redesign* them to reflect agent-specific requirements (multi-step decomposition, tool-mediated execution, and failure-aware recovery). Our construction follows three principles. **(1) Component integration.** Every task is designed such that solving it requires meaningful coordination across P/R/A/F rather than a single-shot response. **(2) Agent-realistic complexity.** We introduce dynamic constraints, multi-turn feedback, and tool/environment interactions to mirror practical agent deployments. **(3) Capability coverage.** We ensure tasks collectively span the core capability spectrum targeted by modular workflows, including planning under constraints, multi-step reasoning, precise action formatting/execution, and reflection-based error recovery.

CapaBench includes seven task types grouped into three super-categories:

- **Daily Activities**: Online Shopping (Yao et al., 2023a), Navigation Planning (Lin et al., 2024), Ticket Ordering (Lin et al., 2024).
- **Computation**: Mathematical Solver (Hendrycks et al., 2021), Automatic Theorem Proving (ATP) (Zheng et al., 2021; The Coq Development Team, 2024; The Lean Prover Team, 2024; The Isabelle Team, 2024), Operating System interaction (OS) (Liu et al., 2023; The learnGitBranching Team).
- **Role Control**: Multi-agent Robot Coordination (RobotCoop) (Mandi et al., 2023).

Across categories, we explicitly instantiate *agent-centric* mechanisms that are typically absent from static QA benchmarks. In **Math**, tasks require tool-mediated solving: the agent must decide when to invoke search/compute tools and verify intermediate results. In **ATP**, success depends on producing *executable* proof scripts (e.g., Coq/Lean/Isabelle), making action precision and iterative debugging essential. In interactive environments (**Shopping**, **OS**, **RobotCoop**), agents must reach a target state via valid environment actions under step budgets. For each task, success is determined by the underlying environment or verifier (e.g., purchase completion, constraint satisfaction, proof-checking pass/fail, OS command outcomes, or goal-state achievement). Reference solutions are manually reviewed to ensure correctness and consistency with the success criteria.

Experiments

Experimental Setup

We apply ShapleyFlow to analyze component contributions across 1,500+ agentic workflows using systematic configuration testing. For each workflow, we employ **Llama3-8B-Instruct** as the baseline implementation for all components (Planning, Reasoning, Action, Reflection), then systematically replace components with target implementations to evaluate all $2^4 = 16$ possible configurations. This approach isolates the contribution of each component upgrade while maintaining consistent baseline conditions.

The choice of **Llama3-8B-Instruct** as the default model implementation is motivated by three factors: (1) it is open-source and easy to deploy at scale, making it practical for large-scale experiments; (2) its lightweight architecture ensures efficient evaluation of thousands of workflows; (3) its moderate task success rates create a balanced baseline, allowing the performance impact of replacing components with more advanced models to be clearly observed and quantified.

We evaluate 9 classic LLMs across 3 categories:

- **API Models**: Claude-3.5-Sonnet (Anthropic, 2024), GPT-4-turbo (OpenAI, 2023), GPT-4o-mini (OpenAI, 2024), GLM-4-air (THUDM,

Table 2: Results Across Datasets. Metrics for baseline models are highlighted in blue. The evaluation covers 9 models across 7 tasks. Results marked with '*' below each dataset indicate the best-performing 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**	P	–	-0.004	0.071	**0.106**	-0.030	-0.048	0.024	0.026	0.071	-0.028
	R	–	0.019	-0.025	**0.077**	0.004	0.036	0.016	-0.074	0.011	0.005
	A	–	0.056	0.068	-0.059	**0.156**	0.080	0.004	0.014	-0.045	0.117
	F	–	-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**	P	–	0.000	0.006	0.001	-0.002	0.021	**0.023**	0.008	0.001	-0.009
	R	–	0.030	0.027	-0.008	0.012	-0.035	**0.055**	0.014	-0.003	-0.019
	A	–	**0.106**	0.081	0.005	0.099	0.048	0.042	0.099	-0.051	0.046
	F	–	-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**	P	–	0.003	0.032	-0.195	0.119	**0.183**	-0.111	-0.043	0.151	0.004
	R	–	0.186	0.243	0.172	0.181	0.054	-0.070	**0.301**	-0.001	0.089
	A	–	**0.217**	0.049	-0.020	-0.000	-0.083	-0.020	0.028	0.006	-0.275
	F	–	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**	P	/	0.038	0.067	0.056	0.065	0.005	-0.060	0.048	**0.115**	0.028
	R	/	**0.131**	0.021	0.044	0.107	0.003	-0.000	0.065	0.059	0.031
	A	/	0.442	0.343	0.348	0.483	0.164	-0.044	**0.492**	0.182	0.327
	F	/	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**	P	/	0.012	0.018	0.002	0.018	**0.025**	0.008	0.012	0.016	0.019
	R	/	**0.057**	-0.016	0.005	0.030	0.018	0.010	0.027	0.019	-0.056
	A	/	**0.660**	0.345	0.161	0.511	0.039	-0.009	0.541	0.084	0.125
	F	/	**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**	P	–	**0.114**	0.075	-0.024	0.090	-0.005	-0.014	0.107	0.021	0.043
	R	–	**0.388**	0.189	0.116	0.268	0.033	-0.000	0.329	-0.004	0.152
	A	–	**0.319**	0.196	0.008	0.277	0.052	-0.021	0.316	0.204	0.175
	F	–	**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**	P	–	**0.078**	0.042	0.047	0.060	0.032	0.004	0.050	0.065	0.077
	R	–	**0.458**	0.305	0.305	0.311	0.194	0.047	0.395	0.215	0.313
	A	–	**0.071**	0.065	0.041	0.053	0.009	0.019	0.070	0.060	0.040
	F	–	-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

2024), Doubao-pro-4k (AI, 2024a).

- **Mid-parameter Open-Source Models(40B-100B)**: Llama3.1-70B-Instruct (AI, 2024b), Mixtral-8x7B-Instruct (Jiang et al., 2024).
- **Low-parameter Open-Source Models($\leq$32B)**: Qwen2.5-32B-Instruct (Qwen et al., 2025), Mistral-8B-Instruct (AI, 2023).

All experiments use consistent settings (temperature=0.0, max tokens=2048, top-p=1.0) on NVIDIA A100-80GB GPUs with vLLM for efficient inference. The evaluation metric is task success rate, measuring the proportion of successfully completed tasks across all tasks.

Component Contribution Attribution

Table 2 presents comprehensive component contribution analysis across workflow categories. The Planning (P), Reasoning (R), Action (A), and Reflection (F) rows report per-model Shapley values $\phi_i^{(m)}$, which quantify the expected gain attributable to upgrading component i *to model* m, averaged over all upgrade contexts under the upgrade-coalition game induced by m.

Configuration Recommendation for Workflow

We use Shapley values as a practical signal for *model allocation* under a fixed workflow. Given a model pool, we construct a recommended configuration by assigning each component the model that achieves the highest Shapley value for that component (computed with respect to a shared baseline). We mark the resulting configurations with '*' in Table 2. In our four-component setting, we can explicitly evaluate the recommendation $\hat{a}$ and compare it against the best configuration *within the evaluated search space* (e.g., per-model upgrade coalitions and selected mixed assignments). Empirically, the Shapley-guided assignments consistently reach the strongest in our experiments.

Component Impact Validation

Figure 3 shows that Shapley attributions are well aligned with observed configuration performance trends, highlighting how high-contribution components tend to appear in high-performing configurations. Studying Claude-3.5-Sonnet on Math, we observe clear correspondence between component contributions and configuration performance. High-Shapley configurations like (P,R,A) achieve 78.0% success rate, dramatically outperforming

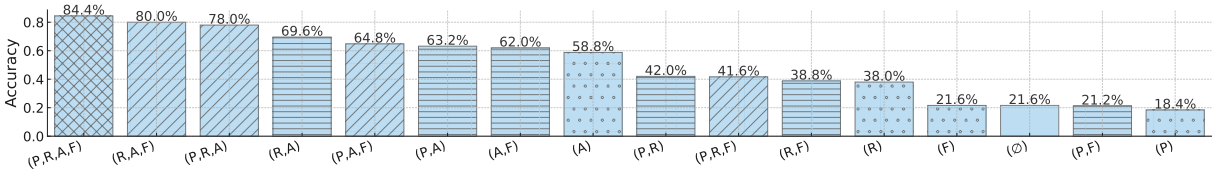
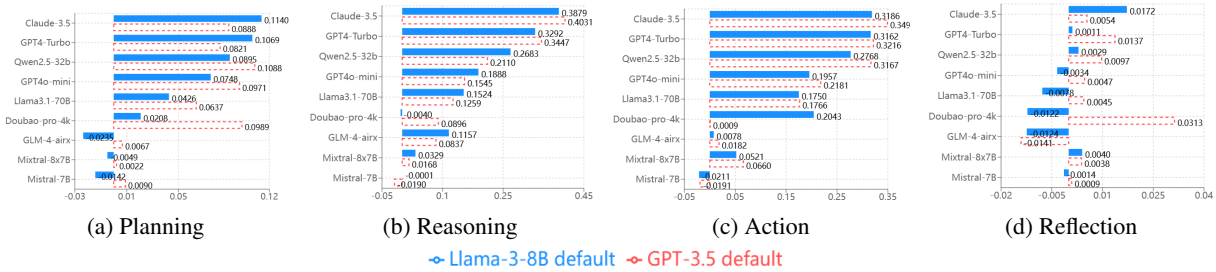


Figure 3: Results of all combinations in Math (Algebra) for Claude-3.5-Sonnet under different configurations. The pattern of the bars indicates the number of components (ranging from 0 to 4) that Claude is involved in.



(a) Planning (b) Reasoning (c) Action (d) Reflection

Figure 4: Comparison of Shapley values under different default models.

the 21.6% baseline. Incremental improvements align with predictions: adding Planning alone improves performance to 18.4%, while (P,A) reaches 63.2%. Low configurations like (P,F) yield poor performance (21.2%), confirming Shapley values accurately quantify component importance.

Analysis

Cross-Domain Optimization Patterns Figure 5 demonstrates distinct optimization strategies across workflow categories. Computation-intensive workflows (Math, ATP-Coq) benefit most from Action component upgrades, while interactive workflows (OS, RobotCoop) show strongest gains from Reasoning-Planning combinations. This systematic analysis enables domain-specific optimization strategies, guiding practitioners toward the most effective component investments for their use cases.

Component Specialization Patterns Our comprehensive analysis shows systematic component specialization across workflow types:

- **Reasoning-Dominant Workflows**: Interactive scenarios (Ticket, RobotCoop, OS) show highest Reasoning contributions. These workflows require dynamic decision-making, constraint balancing, and real-time adaptation. Strong Reasoning components enable effective uncertainty handling and logical inference under evolving conditions.
- **Action-Dominant Workflows**: Precision-critical tasks (Shopping, Math, ATP) prioritize Action components. These workflows demand exact procedural execution, syntactic correctness, and systematic verification. Robust Action components ensure reliable step-by-step execution without errors.
- **Reflection Component Analysis**: Across most workflow types, Reflection shows consistently lower Shapley value, indicating limited impact on task performance. This pattern likely stems from two factors: (1) success rates inadequately capture reflection quality, a model's ability to analyze its own mistakes doesn't directly translate to improved outcomes; (2) without guidance from stronger models, Reflection struggles to identify error root causes, limiting practical effectiveness in driving performance improvements.

Quantitative Analysis of Component Synergy ShapleyFlow enables quantitative analysis of *pairwise interaction effects* between components. In computation-intensive tasks (Math, ATP), the strongest positive synergies consistently occur between Reasoning and Action, indicating that execution gains depend on the availability of strong reasoning. In contrast, interactive environments (Robot Cooperation, Operating System) exhibit higher synergies between Planning and Reasoning, reflecting the need for coordinated anticipation and decision-making. Across most domains, interactions involving Reflection are weak or inconsistent, consistent with its low Shapley values under success-rate metrics. Overall, these results confirm that component utilities are non-additive and highlight the importance of interaction-aware allocation beyond standard ablations.

Framework Robustness Analysis We validate framework robustness by testing sensitivity to

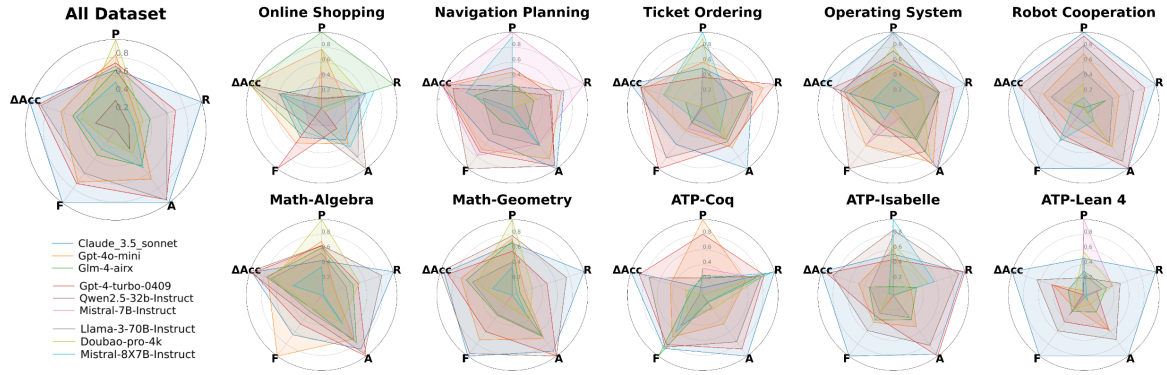


Figure 5: Radar plot comparing model performance across tasks with key contributions.

baseline model selection. Replacing Llama3-8B-Instruct with GPT-3.5-Turbo as the baseline model, we re-analyze component contributions across Robot Cooperation workflows. Figure 4 shows absolute Shapley values shift with baseline strength, but relative component rankings remain highly consistent. We use the pairwise consistency metric to quantify ranking stability:

$$\text{Consistency Rate} = \frac{\text{Consistent Model Pair Rankings}}{\text{Total Model Pairs}} \quad (4)$$

Results show strong consistency: Reasoning (91.67%), Action (86.11%), Planning (72.22%), with overall rate of 85.18%. This demonstrates that ShapleyFlow provides reliable contribution attribution regardless of baseline choice, ensuring robust optimization recommendations.

Attribution Consistency Validation To validate that ShapleyFlow's contribution attribution aligns with traditional evaluation methods, we conduct a consistency study by applying both ShapleyFlow and GPT-o1-mini based LLM-as-judge on successful Algebra workflow trajectories. We evaluate 2,180 single-step workflow samples, where GPT-o1-mini assesses semantic rationality and task completion for Planning/Reasoning components, and logical comprehension for Action components. Figure 6 demonstrates strong correlation between Shapley values and independent GPT assessment scores across different components: Planning (0.81), Reasoning (0.77), Action (0.67). The consistent ranking patterns across diverse methods confirm that ShapleyFlow's contribution attribution captures component-specific capabilities that align with existing automated evaluation approaches.

Comparison with Ablation Study One-factor ablation is sample-efficient ($n+1$ runs) but can be misleading for agentic workflows because the effect of upgrading a component depends on which

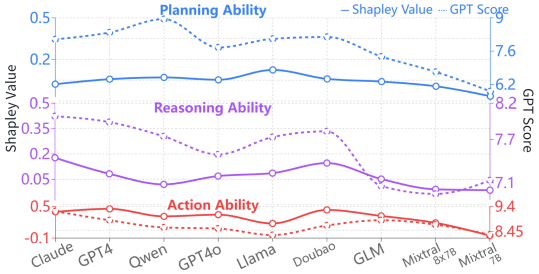


Figure 6: Planning, Reasoning, Action results on Algebra. The left Y-axis shows Shapley values with solid lines and the right shows GPT scores with dashed lines.

other components are upgraded alongside it. In ablation, the "contribution" of a component is measured in a single fixed context (e.g., holding all other components at baseline), so the conclusion can change if we choose a different reference configuration. This is especially salient for components such as Reflection, whose utility often depends on whether Action has produced informative observations and whether Reasoning has formed a coherent hypothesis to debug. ShapleyFlow pays the extra cost of evaluating all $2^{|N|}$ coalitions (16 vs. 5 for $|N| = 4$) and attributes gains by averaging a component's marginal improvement over *all* interaction contexts induced by other upgrades, yielding attribution that is not an artifact of a particular ablation path and that faithfully captures non-additive interactions.

Conclusion

We present **ShapleyFlow**, a game theoretic approach for attributing and optimizing agentic workflows. ShapleyFlow computes Shapley values over component coalitions to quantify both marginal contributions and interaction effects among planning, reasoning, action, and reflection. Across 9 LLMs and 1,500+ tasks in 7 domains, our analysis yields actionable configuration recommendations and empirical guidelines for allocating model capacity across workflow components.

Limitations

Scalability. Exact Shapley-value attribution requires evaluating all $2^{|N|}$ upgrade coalitions, which can be computationally prohibitive for workflows with many components. While our study uses a four-component decomposition to enable exhaustive evaluation, scaling ShapleyFlow to more fine-grained architectures may require (i) restricting attribution to task-relevant or budget-relevant subsets of components, and/or (ii) adopting approximation techniques such as KernelSHAP (Lundberg and Lee, 2017), SVARM (Kolpaczki et al., 2024), and other sampling-based estimators.

Attribution choice. We adopt the Shapley values due to its axiomatic guarantees and its ability to capture non-additive effects through coalition averaging. However, other cooperative-game attributions such as the Banzhaf value (Dragan, 1996) may yield different trade-offs between interpretability and computational cost, and interaction-focused extensions (e.g., Shapley interactions (Muschalik et al., 2024)) could provide a more expressive view of higher-order component synergies. Exploring these alternatives may improve attribution fidelity and support more scalable configuration guidance for modular agentic workflows.

Pool-level recommendation. Our pool-level mixed assignment $\hat{a}$ is a heuristic built on per-model Shapley profiles and does not provide optimality guarantees over the full combinatorial space $\mathcal{M}^{|N|}$. It is intended as a practical decision signal; stronger search/optimization strategies (e.g., constrained search over $\mathcal{M}^{|N|}$) are complementary.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (62322603).

References

Deepak Bhaskar Acharya, Karthigeyan Kuppan, and B. Divya. 2025. Agentic ai: Autonomous intelligence for complex goals—a comprehensive survey. *IEEE Access*, 13:18912–18936.

Doubao AI. 2024a. Doubao-pro-4k. Accessed from Doubao AI's GitHub repository.

Meta AI. 2024b. Llama 3.1 70b instruct. Accessed from HuggingFace repository.

Mistral AI. 2023. Mistral 8b instruct.

Anthropic. 2024. Claude 3.5 sonnet. Accessed from Anthropic's official website.

Tom B Brown, Benjamin Mann, Nick Ryder, and 1 others. 2020. Language models are few-shot learners. *Advances in Neural Information Processing Systems*, 33:1877–1901.

Huacan Chai, Zijie Cao, Maolin Ran, Yingxuan Yang, Jianghao Lin, Xin Peng, Hairui Wang, Renjie Ding, Ziyu Wan, Muning Wen, Weiwen Liu, Weinan Zhang, Fei Huang, and Ying Wen. 2025. Parl-mt: Learning to call functions in multi-turn conversation with progress awareness. *Preprint*, arXiv:2509.23206.

Jun Shern Chan, Neil Chowdhury, Oliver Jaffe, James Aung, Dane Sherburn, Evan Mays, Giulio Starace, Kevin Liu, Leon Maksin, Tejal Patwardhan, Aleksander Madry, and Lilian Weng. 2025. MLE-bench: Evaluating machine learning agents on machine learning engineering. In *The Thirteenth International Conference on Learning Representations*.

Wei-Lin Chiang, Lianmin Zheng, Ying Sheng, Anastasios Nikolas Angelopoulos, Tianle Li, Dacheng Li, Banghua Zhu, Hao Zhang, Michael Jordan, Joseph E. Gonzalez, and Ion Stoica. 2024. Chatbot arena: An open platform for evaluating LLMs by human preference. In *Forty-first International Conference on Machine Learning*.

Shay Cohen, Gideon Dror, and Eytan Ruppin. 2007. Feature selection via coalitional game theory. *Neural Comput.*, 19(7):1939–1961.

Irinel Dragan. 1996. New mathematical properties of the banzhaf value. *European Journal of Operational Research*, 95(2):451–463.

Amirata Ghorbani and James Zou. 2019. Data shapley: Equitable valuation of data for machine learning. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 2242–2251. PMLR.

Significant Gravitas. 2023. Autogpt: An open-source platform for autonomous ai agents. https://github.com/Significant-Gravitas/AutoGPT. Accessed: 2025-11-04.

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 Zheng, Jonathan P. Chen, Yuheng Cheng, Ceyao Zhang, Zili Wang, Steven Ka Shing Yau, Zi Hen Lin, Liyang Zhou, Chenyu Ran, Lingfeng Xiao, and Chenglin Wu. 2023. Metagpt:

Meta programming for multi-agent collaborative framework. *ArXiv*, abs/2308.00352.

Shengran Hu, Cong Lu, and Jeff Clune. 2025. Automated design of agentic systems. In *The Thirteenth International Conference on Learning Representations*.

Albert Q. Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, Gianna Lengyel, Guillaume Bour, Guillaume Lample, Lélio Renard Lavaud, Lucile Saulnier, Marie-Anne Lachaux, Pierre Stock, Sandeep Subramanian, Sophia Yang, and 7 others. 2024. Mixtral of experts. *Preprint*, arXiv:2401.04088.

Patrick Kolpaczki, Viktor Bengs, Maximilian Muschalik, and Eyke Hüllermeier. 2024. Approximating the shapley value without marginal contributions. *Preprint*, arXiv:2302.00736.

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, and 1 others. 2023. Agentbench: Evaluating llms as agents. *arXiv preprint arXiv:2308.03688*.

Scott Lundberg and Su-In Lee. 2017. A unified approach to interpreting model predictions. *Preprint*, arXiv:1705.07874.

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

Maximilian Muschalik, Hubert Baniecki, Fabian Fumagalli, Patrick Kolpaczki, Barbara Hammer, and Eyke Hüllermeier. 2024. shapiq: Shapley interactions for machine learning. *Preprint*, arXiv:2410.01649.

OpenAI. 2023. Gpt-4 turbo.

OpenAI. 2024. Gpt-4o mini. Accessed from OpenAI's developer resources.

Josh Achi OpenAI, Steven Adler, Sandhini Agarwal, and 1 others. 2024. Gpt-4 technical report. *Preprint*, arXiv:2303.08774.

Shuofei Qiao, Runnan Fang, Zhisong Qiu, Xiaobin Wang, Ningyu Zhang, Yong Jiang, Pengjun Xie, Fei Huang, and Huajun Chen. 2025. Benchmarking agentic workflow generation. In *The Thirteenth International Conference on Learning Representations*.

Qwen, :, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, and 25 others. 2025. Qwen2.5 technical report. *Preprint*, arXiv:2412.15115.

Matthew Renze and Erhan Guven. 2024. The benefits of a concise chain of thought on problem-solving in large language models. In *2024 2nd International Conference on Foundation and Large Language Models (FLLM)*, page 476–483. IEEE.

Ranjan Sapkota, Konstantinos I Roumeliotis, and Manoj Karkee. 2025. Ai agents vs. agentic ai: A conceptual taxonomy, applications and challenge. *arXiv preprint arXiv:2505.10468*.

Lloyd S. Shapley. 1952. *A Value for N-Person Games*. RAND Corporation, Santa Monica, CA.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. 2023. Reflexion: language agents with verbal reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 36, pages 8634–8652. Curran Associates, Inc.

The Coq Development Team. 2024. The coq proof assistant. `https://coq.inria.fr/`.

The Isabelle Team. 2024. The isabelle theorem prover. `https://isabelle.in.tum.de/`.

The Lean Prover Team. 2024. The lean theorem prover. `https://leanprover.github.io/`.

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

THUDM. 2024. Glm-4 air. `https://github.com/THUDM/GLM-4`. Accessed: 2025-07-31.

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*.

John Yang, Carlos E Jimenez, Alex L Zhang, Kilian Lieret, Joyce Yang, Xindi Wu, Ori Press, Niklas Muennighoff, Gabriel Synnaeve, Karthik R Narasimhan, Diyi Yang, Sida Wang, and Ofir Press. 2025. SWE-bench multimodal: Do AI systems generalize to visual software domains? In *The Thirteenth International Conference on Learning Representations*.

Yifei Yang, Haoqiang Wu, Chen Zhao, Mingzhe Liu, and 1 others. 2024a. Agentquest: A multi-phase task planning and execution benchmark for autonomous agents. *arXiv preprint arXiv:2402.01786*.

Yingxuan Yang, Huayi Wang, Muning Wen, Xiaoyun Mo, Qiuying Peng, Jun Wang, and Weinan Zhang. 2024b. P3: A policy-driven, pace-adaptive, and diversity-promoted framework for data pruning in llm training. *Preprint*, arXiv:2408.05541.

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

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2023b. React: Synergizing reasoning and acting in language models. *Preprint*, arXiv:2210.03629.

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, and 5 others. 2024. Mmau: A holistic benchmark of agent capabilities across diverse domains. *Preprint*, arXiv:2407.18961.

Wei Zhang, Junnan Wu, Tianhao Wang, Zhihao Hu, and 1 others. 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*.

Han Zhou, Xingchen Wan, Ruoxi Sun, Hamid Palangi, Shariq Iqbal, Ivan Vulić, Anna Korhonen, and Sercan Ö. Arık. 2025. Multi-agent design: Optimizing agents with better prompts and topologies. *Preprint*, arXiv:2502.02533.

Ruiwen Zhou, Yingxuan Yang, Muning Wen, Ying Wen, Wenhao Wang, Chunling Xi, Guoqiang Xu, Yong Yu, and Weinan Zhang. 2024. Trad: Enhancing llm agents with step-wise thought retrieval and aligned decision. *Preprint*, arXiv:2403.06221.

Appendix for Dataset Details

Our benchmark provides a carefully designed evaluation system that tests the key abilities needed for effective agent systems. Instead of simply using existing environments, we have thoughtfully redesigned and improved seven task categories to create a well-rounded capability assessment framework. These tasks evaluate four core agent abilities: (1) **planning** (including breaking down tasks step by step and handling resource limits), (2) **reasoning** (such as logical thinking and using background knowledge), (3) **action** (like changing the environment or making choices through interaction), and (4) **reflection** (mainly analyzing failures and recovery). The chosen tasks, covering **Daily Activities** (Shopping, Navigation, Ticket), **Computation** (Math, ATP, OS), and **Role Control** (Robot), together offer a broad and realistic test of agent performance across different situations.

Table 3: 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	✓	✓	✓	✓	✓	✓	✓

Importantly, while drawing inspiration from established environments, we have substantially extended each task with novel challenges, enhanced complexity, and rigorous evaluation metrics that specifically target modular agent capabilities. This comprehensive construction ensures our benchmark provides meaningful cross-cutting insights about agent capabilities rather than just environment-specific performance. Our benchmark encompasses over 1500 diverse task instances across all categories, providing robust statistical foundations for our Shapley Value-based attribution methodology. We ensure the quality and diversity of our tasks through manual or rule-based construction, with each item **reviewed by human annotators** to ensure it is accurate, clear, and relevant to agent capabilities.

Table 4: 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

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 evaluate 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.

Ticket Ordering.

The Ticket Ordering task evaluates the ability of agents to collaboratively provide the best flight combinations for two users. The dataset consists of 150 tasks, which are designed to benchmark the performance of different agents in ticket ordering.

Table 5: 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*

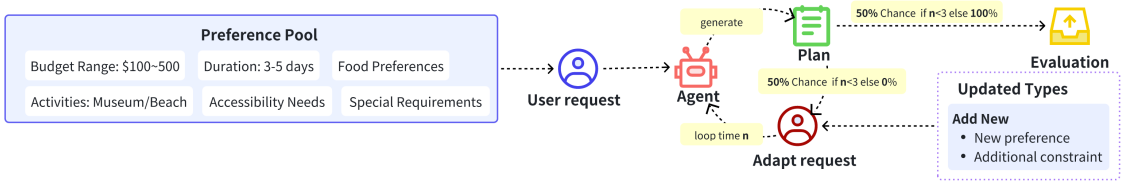


Figure 7: An Example of Navigation Planning.

Inspired by the framework presented by (Lin et al., 2024), we build our evaluation framework based on their structure. Specifically, we use the provided code to generate the dataset, which includes two users' calendars. The tasks are created by combining the users' calendar data, and agents are then asked to provide flight recommendations based on this information.

Table 2 shows the experimental results for the Ticket Ordering task. The baseline model achieves an accuracy of 19.94%. `Claude-3.5-Sonnet` achieves the highest accuracy of 62.85%, improving by +42.91%. `gpt-4-turbo-0409` follows with an accuracy of 54.37%, improving by +34.43%. The accuracy range, from 0.0% (`Mistral-7B-Instruct`) to 62.85%, highlights the dataset's ability to differentiate models based on their performance.

The dataset emphasizes Reasoning and Action capabilities, as seen in the high `R` and `A` Shapley values for top models like `Claude-3.5-Sonnet`, `gpt-4-turbo-0409`, and `qwen2.5-32b-Instruct`. Models with stronger Reasoning and Action abilities show significant accuracy improvements, whereas those with lower values for these modules, such as `Mistral-7B-Instruct`, experience considerable performance deficits.

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:

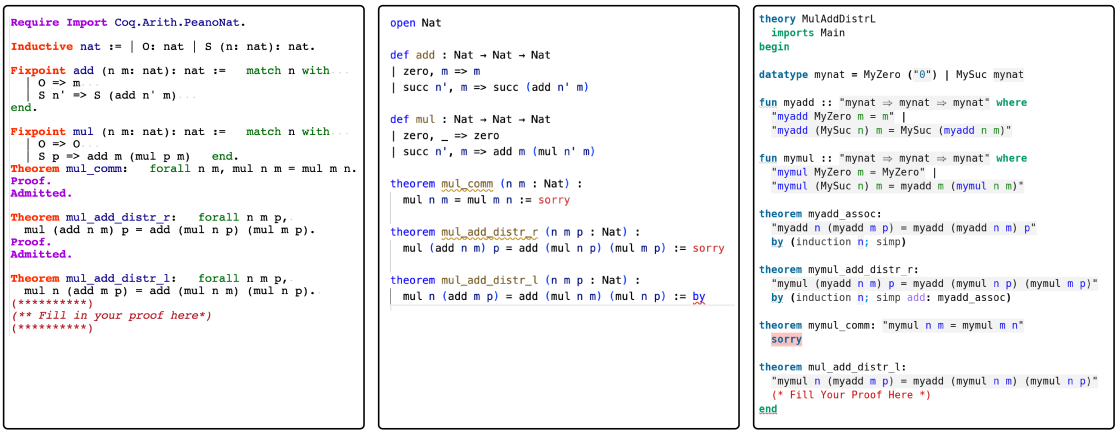
(1) Summarizing Knowledge Points: All problems in the MATH dataset were analyzed using GPT-4 to extract a comprehensive list of key concepts.

(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



```
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 of Automatic Theorem Proving.

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.6.

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.

(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

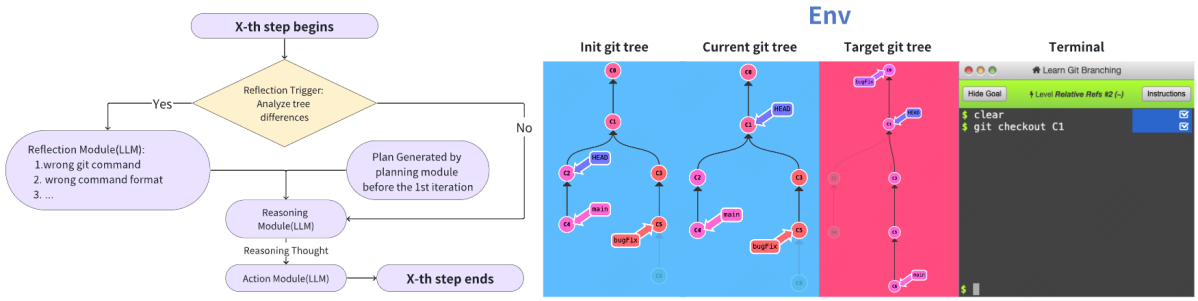


Figure 9: Illustration of OS-git task.

Table 6: 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.

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 sub-tasks 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

```
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.
- 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:
"""
```

```
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 agent's 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:
"""
```

```
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:
"""
```

```
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:
"""
```