ORGANIZING COLLABORATION AROUND CONTRIBUTIONS: AGENTIC INTELLECTUAL PROPERTY FOR LONG-HORIZON MULTI-AGENT SYSTEMS

Anonymous authors
Paper under double-blind review

ABSTRACT

Long-horizon tool use requires more than generating and combining candidate responses: agents must identify which contributions add grounded information to subsequent decisions. We introduce Agentic Intellectual Property (AIP), a framework that organizes multi-agent collaboration around attributable contributions rather than primarily around agent identities, specialized roles, or communication topology. At each environment state, agents contribute Action, Evidence, or Synthesis, enabling collaboration through executable proposals, supporting or opposing reasons, and new decision-relevant conclusions. Independent peer reviews assess contribution novelty, usefulness, and validity, assign credit, and govern admission to a shared ledger. An independent Final Judge uses the current state and retained contributions to generate the sole action committed to the environment. This design encourages useful exploration while seeking to limit the propagation of redundant or unsupported content, using the same backbone across all roles and no counterfactual environment branches. Evaluations on BFCL v4 multi-turn and ToolHop show that AIP achieves the highest overall accuracy among all evaluated single-agent and multi-agent baselines with both Qwen3-4B-Instruct and Llama-3.1-8B-Instruct. AIP reaches 17.5% and 40.1% accuracy with Qwen, and 14.3% and 11.0% with Llama, on the two benchmarks, respectively, while recording fewer average tool errors per task than multiple competing architectures. These results support contribution representation, evaluation, and selective propagation as a complementary design axis for improving long-horizon multi-agent collaboration.

1 INTRODUCTION

Long-horizon tool use requires language agents to connect observations with later actions, rather than solve each decision in isolation (Patil et al., 2025; Ye et al., 2025). Existing approaches strengthen reasoning through response-level consensus (Wang et al., 2023), multi-agent debate (Liang et al., 2024), layered aggregation (Wang et al., 2025), and structured interaction graphs (Qian et al., 2025). These mechanisms organize how responses are generated and combined, but agreement or additional communication does not necessarily supply missing evidence. Figure 1 illustrates this distinction: our Self-Consistency and Mixture-of-Agents adaptations submit orders at an unobserved price, despite unanimous candidates or repeated aggregation. AIP instead retains a quote-query contribution, executes it, and uses the resulting observation in a later user turn. This example motivates a complementary question: beyond who participates and how agents communicate, which contributions deserve to influence subsequent decisions?

We draw inspiration from the concept of intellectual property (IP) in human society. IP recognizes intellectual creations and their creators, with protection and recognition intended to encourage further innovation (World Intellectual Property Organization, 2020). We borrow this emphasis on attribution and recognition, rather than legal exclusivity, and introduce Agentic Intellectual Property (AIP). The central idea is to make an agent's contribution—not merely its participation—the object of evaluation, credit, and subsequent use. Novelty is defined relative to the current task state and visible contributions: a grounded correction, a missing reason, or a new decision-relevant conclu-

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

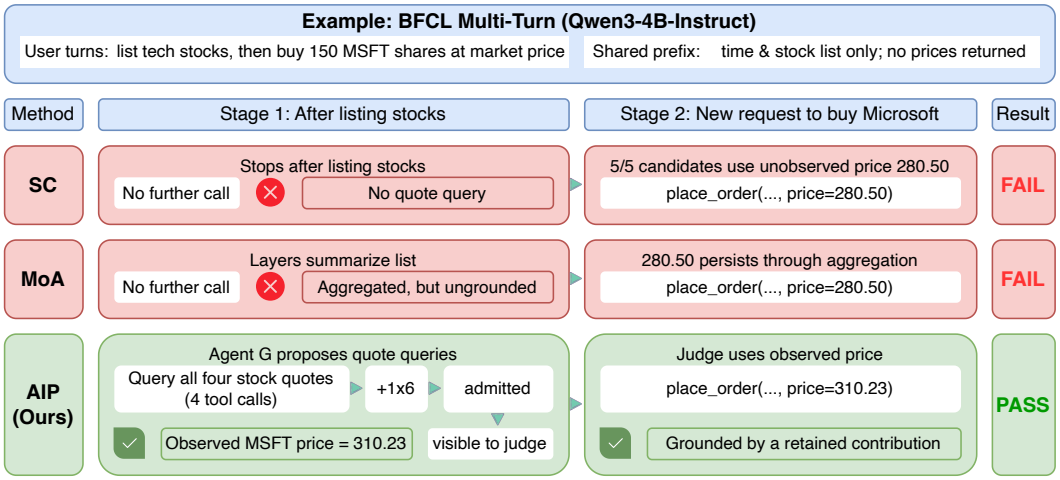

Figure 1: **Two-stage trajectory comparison on BFCL v4 multi-turn.** SC, MoA, and AIP use Qwen3-4B-Instruct-2507 on task `multi_turn_base_128`. Following a shared stock-list observation, SC and MoA later submit orders using an unobserved price. In AIP, the Judge executes the quote-query batch proposed by an admitted contribution, and the resulting observations supply the price used in the next user turn. PASS and FAIL denote official whole-task outcomes. Intervening steps are omitted.

sion can be valuable, whereas rephrasing an existing proposal need not earn additional credit. The objective is therefore useful exploration, not difference for its own sake.

AIP represents contributions as ACTION, EVIDENCE, or SYNTHESIS. At each state, agents extend an initial proposal through peer-reviewed additions or abstain. Review-derived credit governs admission to a shared ledger, and an independent Final Judge generates the sole executed response. Actual observations inform subsequent states, using the same backbone throughout.

Our main contributions are as follows:

- A contribution-centered framework inspired by attribution and recognition in human intellectual property.
- A protocol combining typed contributions, peer-assigned credit, selective propagation, and independent action generation.
- Evaluation on BFCL v4 multi-turn and ToolHop with two backbones, achieving the highest accuracy in all four settings and lower tool-error means in multiple comparisons.

2 Related Work

Multi-agent collaboration. LLM-based collaboration has been organized through specialized workflows, debate, aggregation, and search. MetaGPT (Hong et al., 2024) coordinates specialized roles through standardized workflows and structured intermediate artifacts, while MAD (Liang et al., 2024) uses opposing arguments and adjudication to refine answers. MoA (Wang et al., 2025) aggregates responses across successive layers, and MacNet (Qian et al., 2025) supports iterative artifact refinement through directed acyclic interaction graphs. FoA (Klein et al., 2025) distributes exploration across agents and coordinates search through value-guided resampling. AIP builds on intermediate-output exchange but makes an attributable contribution the unit of recognition and admission. It evaluates the task-relative novelty and grounded usefulness of additional Action, Evidence, and Synthesis contributions before exposing them to subsequent deliberation, providing a complementary design axis to role assignments and communication topology.

Tool use and feedback-based reasoning. Self-Consistency (Wang et al., 2023) selects answers by aggregating diverse sampled reasoning paths, while ReAct (Yao et al., 2023) interleaves reasoning

and actions to incorporate external observations. Reflexion (Shinn et al., 2023) retains feedback-based reflections in episodic memory for subsequent trials, whereas CRITIC (Gou et al., 2024) verifies and revises initial outputs through tool-interactive feedback. These approaches provide mechanisms for using alternative reasoning paths, observations, and critique to improve decisions. AIP instead uses peer assessment to govern which contributions become available during collaboration. Peers assess unexecuted contributions using the current state and available observations, and only the Final Judge commits an action. In particular, Evidence and Synthesis provide advisory content within a decision state rather than post-execution reflection memory. This focus on contribution admission complements answer selection and feedback-driven revision without requiring model-weight updates or trial execution of candidate actions.

3 METHOD

AIP separates contribution generation, contribution admission, and environment execution. As illustrated in Figure 2, agents propose attributable contributions, peers evaluate their added value, and admitted contributions inform an independent Final Judge. The protocol uses one deliberation round ($R = 1$) at each decision state, with seven contributor identities ($N = 7$) and a separate Final Judge. All generation roles share the same backbone parameters θ; their responsibilities are specified through prompts rather than separately trained models.

3.1 DECISION STATES AND TYPED CONTRIBUTIONS

Let $S_t = (H_t, \mathcal{T}_t)$ denote the current decision state: H_t contains received user requests, committed responses, and actual tool observations, and $\mathcal{T}_t$ contains the available schemas. Each contribution records its identifier, author, type, content, and applicable relation to a visible action.

ACTION. Propose a complete, executable next tool-call action grounded in the current state and available tool schema.

EVIDENCE. Provide a grounded, decision-relevant reason to support or oppose an ACTION.

SYNTHESIS. Integrate the current state and visible contributions into a new, decision-relevant conclusion that is not already stated; do not merely restate or summarize existing content.

On ToolHop, ACTION may contain a tool-call batch or a final answer. EVIDENCE and SYNTHESIS remain advisory. Novelty is task-relative, and admission establishes neither truth nor execution. These are contribution types, not fixed agent roles.

3.2 SEQUENTIAL CONTRIBUTION GENERATION

We index the seven contributor identities by $i \in \{0, \ldots, 6\}$, corresponding to Agents A–G. Agent A first generates an ACTION anchor b_t using the base task interface and low-temperature action-generation prompt, without the contribution-specific instructions. Agents B–G review the anchor, which remains available as a reference throughout deliberation regardless of its credit. The Final Judge may retain or replace it.

The remaining agents contribute sequentially. Let $\mathcal{L}_{t,i}$ be the ordered ledger of admitted non-anchor contributions after slot i. Starting from an empty ledger, each agent conditions on the current state, anchor, and previously admitted contributions:

$$\mathcal{L}_{t,0} = \varnothing, \qquad c_{t,i} \sim \pi_\theta^{\mathrm{C}}\big(\cdot \mid S_t, b_t, \mathcal{L}_{t,i-1}\big), \quad i = 1, \ldots, 6. \tag{1}$$

Here, π_θ^{C} denotes the contribution-prompted backbone. Each slot produces one ACTION, EVIDENCE, SYNTHESIS, or explicit `ABSTAIN`, with no type quota or replacement generation. An ACTION may be independent of earlier proposals; EVIDENCE targets a currently visible ACTION. Each candidate is reviewed and its admission resolved before the next contributor is invoked. A valid abstention receives credit zero, is not admitted, and skips its six peer reviews. Thus, contributors can refrain from manufacturing differences when no grounded, decision-relevant addition is available.

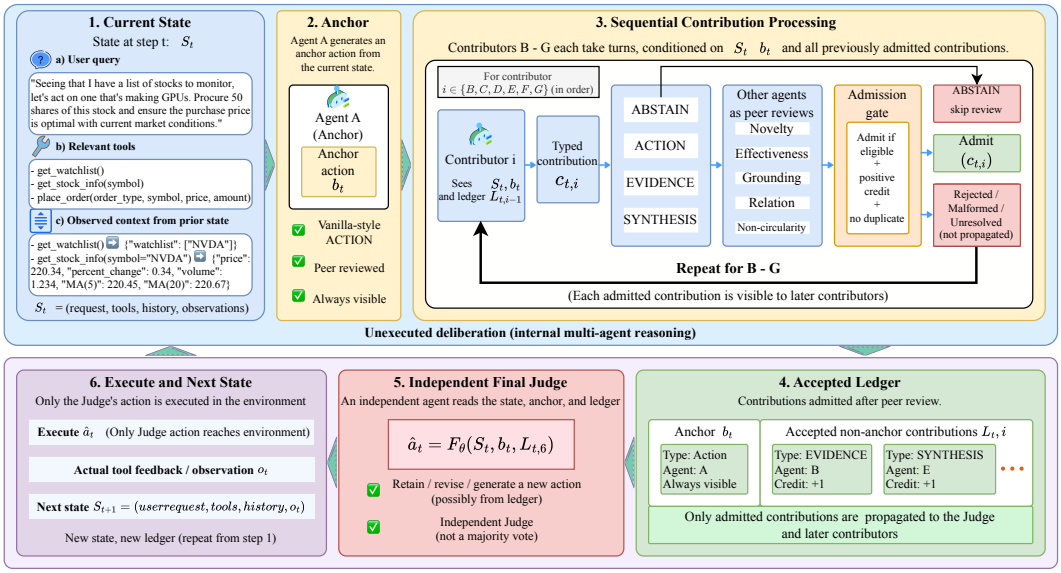

Figure 2: **Overview of Agentic Intellectual Property (AIP).** At each decision state, an initial ACTION anchor is followed by sequential ACTION, EVIDENCE, or SYNTHESIS contributions, with optional abstention. The anchor and non-abstaining candidates receive independent peer reviews. Eligible positive-credit contributions enter the shared ledger with their attribution, while the anchor remains visible throughout. An independent Final Judge generates the next outward action from the current state, anchor, and admitted contributions. Only the Judge's action is submitted to the environment, and the resulting observations inform subsequent decisions.

3.3 Peer Review and Selective Admission

Every non-abstaining candidate, including the anchor, receives six separately generated reviews from the other contributor identities. Reviewers assess the candidate against the current state and contributions visible before it; they do not observe one another's votes. Each reviewer provides five binary judgments: novelty, effectiveness, grounding, relation validity, and non-circularity. Relation and contribution-boundary requirements are applied according to the contribution type. Each review includes a brief rationale and assesses the candidate before execution.

Index $i = 0$ denotes the anchor in the review notation. For a successfully parsed review from peer j, let $q_{t,i,k}^{(j)} \in \{0, 1\}$ denote the k-th judgment. The vote is derived programmatically rather than generated as an additional score:

$$v_{t,i}^{(j)} = 2 \prod_{k=1}^{5} q_{t,i,k}^{(j)} - 1. \tag{2}$$

All five judgments must therefore be true for a positive vote. Malformed reviews are invalid votes, not automatic negative votes. Let $\mathcal{V}_{t,i}$ be set of peers with valid reviews and let $C_{t,i} = 1$ indicate that all six expected reviewer records are present. For a non-abstaining candidate, contribution credit is

$$\kappa_{t,i} = \begin{cases} \min_{j \in \mathcal{V}_{t,i}} v_{t,i}^{(j)}, & C_{t,i} = 1 \text{ and } \mathcal{V}_{t,i} \neq \varnothing, \\ \perp, & \text{otherwise}, \end{cases} \tag{3}$$

where $\perp$ denotes unresolved credit. Positive credit requires unanimity among a nonempty set of valid votes after all six expected reviewer records are present; it does not require six valid approvals. Credit governs contribution admission and visibility during inference; the underlying model weights remain fixed.

For non-anchor slot i, let $\alpha_{t,i} = 1$ exactly when the candidate satisfies the contribution-format and action-reference requirements, receives $\kappa_{t,i} = +1$, and is not an exact duplicate ACTION;

otherwise, $\alpha_{t,i} = 0$. The ledger update is

$$\mathcal{L}_{t,i} = \begin{cases} \mathcal{L}_{t,i-1} \parallel [(c_{t,i}, \kappa_{t,i})], & \alpha_{t,i} = 1, \\ \mathcal{L}_{t,i-1}, & \alpha_{t,i} = 0, \end{cases} \tag{4}$$

where $\parallel$ denotes ordered append. Exact ACTION matching is restricted to the current state's anchor and previously admitted ACTIONs; it does not apply to EVIDENCE or SYNTHESIS. Duplicates are still reviewed, but only admitted non-anchor content and metadata are propagated, without raw review texts. Parsing and serialization details appear in Appendix B.3.

3.4 Independent Execution and State Transition

After all six non-anchor slots, the Final Judge generates a complete next outward action:

$$a_t \sim \pi_\theta^{\mathrm{J}}\big(\cdot \mid S_t, b_t, \mathcal{L}_{t,6}\big). \tag{5}$$

The Judge may retain, revise, or generate an action absent from the ledger; neither credit nor anchor visibility determines execution. Only a_t reaches the environment, whose actual observation o_t updates $H_{t+1} = H_t \| [a_t, o_t]$. New user messages are added when received, and final answers follow the benchmark interface. A fresh ledger is constructed at each state, while committed history carries information forward without resets or trial execution of candidates. A normally completed state uses $50 - 6k_t$ model calls, where k_t is the number of valid contributor abstentions.

4 Experiments

We evaluate Qwen3-4B-Instruct-2507 and Llama-3.1-8B-Instruct, using the same backbone for all generation roles within each setting. Each generation has a total context budget of 131,072 tokens, with up to 126,976 input tokens and 4,096 output tokens. Methods deliberate internally before submitting a selected response to the environment. Executed actions and their observations remain in a single interaction trajectory, without environment resets or trial execution of candidates. We retain method-specific deliberation and sampling settings, so total inference compute varies across methods. Detailed prompts and implementation settings are provided in Appendices A and B.

4.1 Datasets

BFCL v4 multi-turn. The Berkeley Function Calling Leaderboard (BFCL) (Patil et al., 2025), introduced at ICML 2025, evaluates function calling in executable, stateful settings. We use its v4 release and focus on the multi-turn component, covering Base, Missing Function, Missing Parameter, and Long Context. This evaluation set contains 800 multi-turn tasks in total. These tasks require agents to maintain state, select appropriate tools, and use earlier observations when responding to subsequent requests. We retain the official evaluator's task-level correctness checks and allow at most 20 action steps per user turn.

ToolHop. ToolHop (Ye et al., 2025), introduced at ACL 2025, evaluates multi-hop tool use through query-specific, locally executable tools with meaningful dependencies and verifiable answers. We use its *Mandatory* condition, which instructs agents to solve the query using the provided tools. We evaluate on the full ToolHop set of 995 tasks. Each task permits at most nine outward assistant responses, including the final answer; internal collaboration does not consume this interaction budget. We preserve actual tool feedback and use ToolHop's official correctness evaluator, which checks final-answer matches and, where applicable, eligible tool feedback immediately preceding the final answer.

4.2 Baselines

We evaluate homogeneous, single-trajectory implementations of all baselines under shared environment-access constraints; method-specific adaptations are described below and in Appendix B.5.

Vanilla. A single agent generates the next response from the request, tools, and committed history, without additional deliberation.

Table 1: **Overall accuracy and average tool errors per task on BFCL v4 multi-turn and ToolHop.** Results are grouped by backbone, with the same model used across methods within each block. Higher accuracy and lower tool-error values are preferred. Green shading marks the top three distinct ranks within each backbone and metric, with the best values in bold. Accuracy is reported as a fraction, and tool-error statistics are untrimmed arithmetic means over the evaluated tasks for all methods and both backbones.

		BFCL [ICML2025]		ToolHop [ACL2025]	
Category	Method	Overall Acc. ↑	Avg. Tool Errors per Task ↓	Overall Acc. ↑	Avg. Tool Errors per Task ↓
Qwen3-4B-Instruct					
Single-Agent	Vanilla	0.125	2.100	0.290	0.190
	SC [ICLR2023]	0.100	2.188	0.310	0.190
	Reflexion [NeurIPS2023]	0.138	1.963	0.310	**0.130**
Multi-Agent	MAD [EMNLP2024]	0.038	7.488	0.238	0.249
	MacNet [ICLR2025]	0.113	3.063	0.268	0.523
	MoA [ICLR2025]	0.163	2.975	0.328	0.198
	FoA [ICML2025]	0.138	2.538	0.397	0.196
Ours	AIP	**0.175**	**1.750**	**0.401**	0.164
Llama-3.1-8B-Instruct					
Single-Agent	Vanilla	0.063	8.347	0.080	1.333
	SC [ICLR2023]	0.125	**3.389**	0.090	1.196
	Reflexion [NeurIPS2023]	0.100	5.111	0.070	0.967
Multi-Agent	MAD [EMNLP2024]	0.000	15.389	0.070	1.016
	MacNet [ICLR2025]	0.038	12.875	0.010	1.122
	MoA [ICLR2025]	0.000	6.194	0.093	1.275
	FoA [ICML2025]	0.013	5.958	0.089	**0.594**
Ours	AIP	**0.143**	4.321	**0.110**	0.856

Self-Consistency (SC). We adapt Wang et al. (2023) to next-response selection: five independently sampled candidates are grouped by normalized output, and the most frequent response is executed with deterministic tie handling.

Reflexion. We adapt Shinn et al. (2023) to a single ongoing trajectory. After committed actions and observations, the agent updates private reflection memory for later decisions, without environment resets or evaluator feedback.

Multi-Agent Debate (MAD). Following Liang et al. (2024), affirmative and negative agents debate the next response for three rounds. A moderator generates the response submitted to the environment.

MacNet. We instantiate Qian et al. (2025) as a four-actor chain with three critic–actor exchanges per edge and edge-local dialogue. Only the sink actor's final artifact is submitted.

Mixture-of-Agents (MoA). We use six proposers, six intermediate aggregators, and one final aggregator (Wang et al., 2025). Each aggregation layer receives all preceding-layer outputs; only the final response is executed.

Fleet of Agents (FoA). We retain 15 proposal–memory paths with value-guided resampling, $k = 4$, and $\gamma = 0.5$ (Klein et al., 2025). Search is restricted to unexecuted proposals; after the selected response is executed, all paths continue from the committed history.

4.3 METRICS

Overall Accuracy (↑). For method m, let $\mathcal{D}_m$ be its evaluated task set and $\tau_i^{(m)}$ its recorded trajectory on task i. We use the corresponding official benchmark evaluator to obtain a binary task-level outcome $y_i^{(m)}$, and compute

$$\mathrm{Acc}(m) = \frac{1}{|\mathcal{D}_m|} \sum_{i \in \mathcal{D}_m} y_i^{(m)}, \qquad y_i^{(m)} \in \{0, 1\}. \tag{6}$$

This measures completed-task correctness rather than candidate quality or individual-step success. Ordinary method failures remain in evaluated task set and contribute to the accuracy denominator.

Average Tool Errors per Task (↓). Let $E_i^{(m)}$ count explicit errors in the submitted tool interactions of method m on task i: nonempty top-level `error` fields in BFCL tool feedback, or recorded executor errors in ToolHop tool events. Repeated errors count separately; unexecuted candidates and model-service retries do not count. We compute

$$\mathrm{AvgToolErrors}(m) = \frac{1}{|\mathcal{D}_m|} \sum_{i \in \mathcal{D}_m} E_i^{(m)}. \tag{7}$$

All evaluated tasks contribute equally to this arithmetic mean, including tasks with zero tool errors. The same untrimmed aggregation is used for both backbones and all methods. This metric complements task accuracy by measuring the explicit tool-execution error burden of the recorded trajectory.

4.4 RESULTS

Task performance. Table 1 shows that AIP achieves the highest overall accuracy in all four backbone–benchmark settings: 17.5%/40.1% for Qwen and 14.3%/11.0% for Llama on BFCL/ToolHop. AIP also has the lowest tool-error mean on Qwen BFCL (1.750), and fewer tool errors than every multi-agent baseline on Qwen ToolHop and Llama BFCL. On Llama ToolHop, it exceeds MoA's 9.3% accuracy while reducing tool errors from 1.275 to 0.856. Its accuracy advantage thus accompanies a lower execution-error burden in multiple comparisons.

Contribution generation and propagation. Figure 3 examines non-anchor contributions with Qwen on BFCL. SYNTHESIS is the most frequent type, averaging 4.04 structurally valid and 1.79 admitted contributions per state; admitted ACTION and EVIDENCE average 0.49 and 0.26, respectively. Contributions are admitted at every position from B to G, although admission becomes less frequent later in the sequence. The visible discussion therefore includes reasons and conclusions beyond alternative actions, with selective propagation throughout deliberation. These statistics describe admission, not contribution correctness or execution.

5 CASE STUDIES

We trace two selected Qwen trajectories illustrating feedback-based recovery and pre-commit correction. These examples document contribution use rather than establish causal necessity. An additional dependency-revision case appears in Appendix C.

5.1 GROUNDED RECOVERY FROM EXECUTION ERRORS

In BFCL task `multi_turn_miss_func_68`, AIP initially batches a fuel query with a 50-gallon refill, observing 15 gallons and a capacity error. Agent C's admitted SYNTHESIS derives $50 - 15 = 35$ additional gallons, and Agent D proposes the corresponding ACTION. Both receive six positive reviews; the Judge executes the 35-gallon refill, and the environment confirms a full 50-gallon tank. AIP passes the complete task while all seven baselines fail. The inspected Vanilla trajectory stops after the capacity error, leaving the fuel-state mismatch unresolved. This trace illustrates an observation–synthesis–action recovery pathway without discarding the failed interaction.

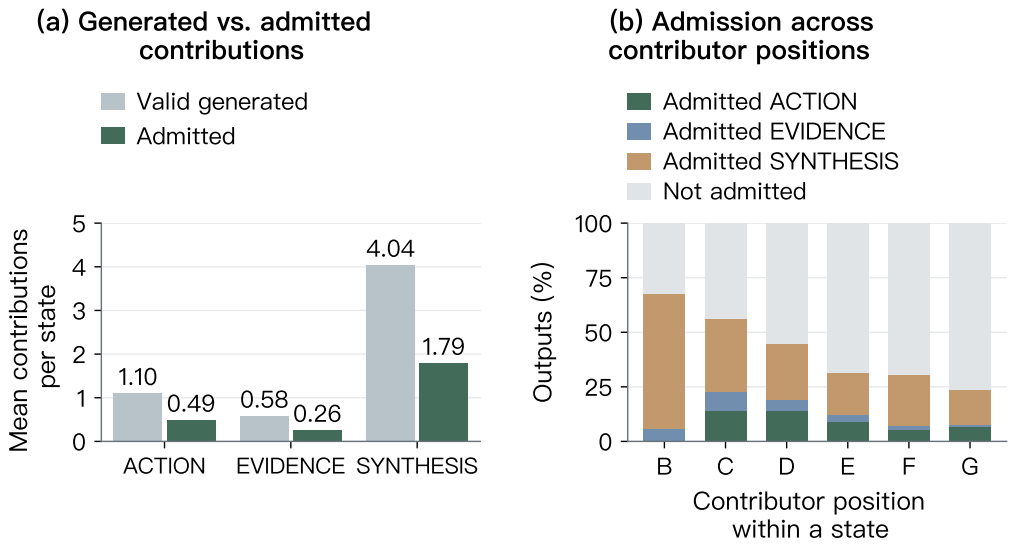

Figure 3: **Contribution generation and propagation with Qwen3-4B-Instruct-2507 on BFCL v4 multi-turn.** (a) Mean structurally valid and admitted contributions per state. (b) Admission outcomes at positions B–G. Both panels exclude the anchor and use states with complete records for all six positions.

5.2 Novel Synthesis for Pre-commit Correction

In ToolHop task 3, tools identify a player's birth year as 1959 and return its unique prime factors $[3, 653]$, yet the anchor answers 3. Agent B's SYNTHESIS derives $3 + 653 = 656$, receives six positive reviews, and enters the ledger. Five subsequent restatements receive negative credit and are not propagated. The Judge answers 656, whereas Vanilla returns 3; only AIP and MacNet are scored correct. This illustrates a retained reasoning increment rather than accumulated agreement. The sum was not returned by a tool, so the case demonstrates answer correction rather than full Mandatory compliance.

6 Ablations

We examine the roles of the contribution space, peer assessment, and deliberation beyond the anchor using Qwen3-4B-Instruct-2507 on BFCL v4 multi-turn. We compare AIP with three ablations under the same task, backbone, context, output, and interaction settings. The AIP results are those reported in Table 1; Figure 4 summarizes the ablation results.

Contributions beyond actions. Action-only restricts the six non-anchor contributors to ACTION or abstention, while retaining peer review, credit-based admission, exact-duplicate rejection, and the Final Judge. Accuracy decreases from 17.5% to 7.5%, the largest drop among the tested variants. Retaining the review and decision structure while removing the EVIDENCE and SYNTHESIS channels therefore does not recover the complete system's observed performance. This comparison supports the joint value of allowing decision-relevant reasons and new conclusions to be contributed explicitly, rather than requiring every non-abstaining contribution to be an executable action.

Peer-guided contribution admission. w/o Peer Review retains all three contribution types but removes peer reviews, review-derived credit, and the positive-credit admission requirement. Structurally eligible contributions remain visible as unreviewed content, subject to the existing reference checks and exact ACTION-duplicate gate. Accuracy falls to 15.0%, while average tool errors increase from 1.750 to 3.088. AIP thus records 43.3% fewer explicit tool errors than this variant. The pattern is consistent with a benefit from peer-guided selective propagation beyond structural eligibility and exact deduplication alone.

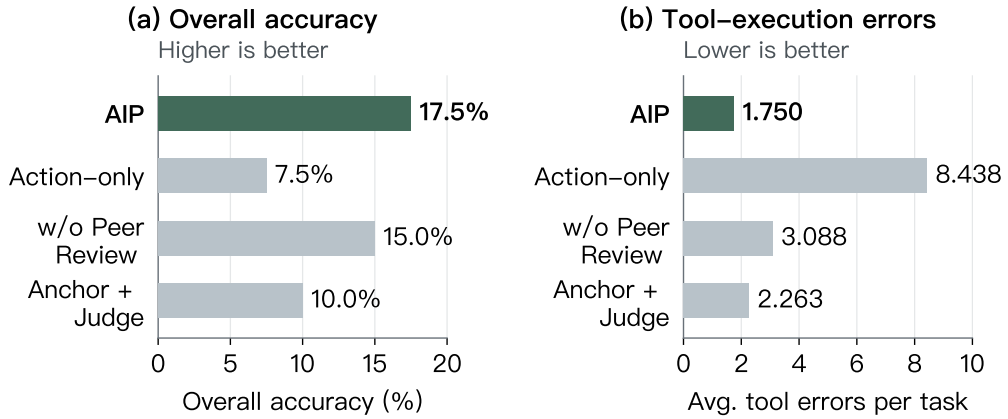

Figure 4: **Ablation results with Qwen3-4B-Instruct-2507 on BFCL v4 multi-turn.** (a) Overall accuracy; higher values are better. (b) Untrimmed average tool errors per task; lower values are better. All conditions retain the Final Judge. The Action-only error mean includes a long-tail trajectory.

Beyond a second decision pass. Anchor + Judge removes all non-anchor contributions and peer reviews, retaining the initial proposal and the same Final Judge decision prompt, with the anchor marked as unreviewed. It reaches 10.0% accuracy and 2.263 tool errors per task, compared with 17.5% and 1.750 for AIP. The complete framework therefore achieves a better observed accuracy–error profile than generating an initial proposal and asking the Judge to reconsider it without additional contributions.

Across these ablations, AIP has the highest observed accuracy and the lowest mean tool-error count, favoring its combination of typed contributions and review-guided admission. This comparison evaluates review-guided admission as a complete module, including its additional inference compute. Additional mechanism ablations and statistical comparisons are provided in Appendix D.

7 Conclusion

We introduced AIP, which organizes collaboration around attributable ACTION, EVIDENCE, and SYNTHESIS contributions through peer review and selective propagation. Across BFCL v4 multi-turn and ToolHop, AIP achieves the highest overall accuracy among the evaluated methods with both backbones, alongside lower tool-error means in multiple comparisons. These results highlight contribution-centered organization as a complementary approach to long-horizon multi-agent collaboration.

Limitations and Future Work

We evaluate AIP with two backbones on two text-based tool-use benchmarks. The comparisons characterize system-level task performance under shared interaction constraints; total inference compute varies by method. Peer-assigned credit guides contribution admission but does not certify correctness. Future work will investigate compute-matched evaluation, broader independent task sets, heterogeneous teams, and adaptive review budgets.

AI Use Statement

LLM-based tools assisted manuscript drafting and revision, and code development. Generative AI also supported figure preparation. The experiments presented in this paper evaluate Qwen3-4B-Instruct and Llama-3.1-8B-Instruct. The authors take responsibility for the final manuscript, methodology, implementation, and reported results.

ETHICS STATEMENT

This study concerns computational evaluation within the benchmark environments in Section 4.1, rather than deployment to live user services. AIP's intellectual property analogy denotes contribution attribution and evaluation, not legal ownership (Section 1). Tool-using agents may produce unsupported actions or expose sensitive information when deployed without appropriate controls. Applications beyond these benchmarks should therefore use restricted tool permissions, data protection, and human authorization for consequential actions. The execution boundary in Section 3.4 complements, but does not replace, such safeguards.

REPRODUCIBILITY STATEMENT

Section 3 specifies the AIP protocol, including contribution types, credit aggregation, ledger updates, and action commitment. Section 4 describes the benchmarks, baseline adaptations, and evaluation metrics, while Section 6 and Appendix D detail the ablation comparisons. Appendix A provides role instructions and output templates. Appendix B records model checkpoints, decoding and seed settings, tool interfaces, contribution validation, context and stopping rules, baseline configurations, and evaluator versions. Together, these descriptions support reimplementation of the methods and reproduction of the evaluation procedure.

REFERENCES

Zhibin Gou, Zhihong Shao, Yeyun Gong, Yelong Shen, Yujiu Yang, Nan Duan, and Weizhu Chen. CRITIC: Large language models can self-correct with tool-interactive critiquing. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=Sx038qxjek.

Sirui Hong, Mingchen Zhuge, Jonathan Chen, Xiawu Zheng, Yuheng Cheng, Jinlin Wang, Ceyao Zhang, Zili Wang, Steven Ka Shing Yau, Zijuan Lin, Liyang Zhou, Chenyu Ran, Lingfeng Xiao, Chenglin Wu, and Jürgen Schmidhuber. MetaGPT: Meta programming for a multi-agent collaborative framework. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=VtmBAGCN7o.

Lars Henning Klein, Nearchos Potamitis, Roland Aydin, Robert West, Caglar Gulcehre, and Akhil Arora. Fleet of agents: Coordinated problem solving with large language models. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 30986–31019. PMLR, 13–19 Jul 2025. URL https://proceedings.mlr.press/v267/klein25a.html.

Tian Liang, Zhiwei He, Wenxiang Jiao, Xing Wang, Yan Wang, Rui Wang, Yujiu Yang, Shuming Shi, and Zhaopeng Tu. Encouraging divergent thinking in large language models through multi-agent debate. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (eds.), *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 17889–17904, Miami, Florida, USA, November 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.emnlp-main.992. URL https://aclanthology.org/2024.emnlp-main.992/.

Shishir G Patil, Huanzhi Mao, Fanjia Yan, Charlie Cheng-Jie Ji, Vishnu Suresh, Ion Stoica, and Joseph E. Gonzalez. The berkeley function calling leaderboard (BFCL): From tool use to agentic evaluation of large language models. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 48371–48392. PMLR, 13–19 Jul 2025. URL https://proceedings.mlr.press/v267/patil25a.html.

Chen Qian, Zihao Xie, YiFei Wang, Wei Liu, Kunlun Zhu, Hanchen Xia, Yufan Dang, Zhuoyun Du, Weize Chen, Cheng Yang, Zhiyuan Liu, and Maosong Sun. Scaling large language model-based multi-agent collaboration. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=K3n5jPkrU6.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: language agents with verbal reinforcement learning. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 8634–8652. Curran Associates, Inc., 2023. doi: 10.52202/075280-0377. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/1b44b878bb782e6954cd888628510e90-Paper-Conference.pdf.

Junlin Wang, Jue Wang, Ben Athiwaratkun, Ce Zhang, and James Zou. Mixture-of-agents enhances large language model capabilities. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=h0ZfDIrj7T.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=1PL1NIMMrw.

World Intellectual Property Organization. *What Is Intellectual Property?* World Intellectual Property Organization, Geneva, 2020. URL https://www.wipo.int/edocs/pubdocs/en/wipo_pub_450_2020.pdf.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R Narasimhan, and Yuan Cao. ReAct: Synergizing reasoning and acting in language models. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=WE_vluYUL-X.

Junjie Ye, Zhengyin Du, Xuesong Yao, Weijian Lin, Yufei Xu, Zehui Chen, Zaiyuan Wang, Sining Zhu, Zhiheng Xi, Siyu Yuan, Tao Gui, Qi Zhang, Xuanjing Huang, and Jiecao Chen. ToolHop: A query-driven benchmark for evaluating large language models in multi-hop tool use. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 2995–3021, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-251-0. doi: 10.18653/v1/2025.acl-long.150. URL https://aclanthology.org/2025.acl-long.150/.

A PROMPTS USED IN AIP

AIP uses the same role instructions and contribution-handling rules with both backbones. Task instructions and action-output formats follow the benchmark, while model-specific chat templates provide message serialization. This appendix presents the task instructions, role-specific prompt passages, and output templates. Braced placeholders denote dynamic inputs such as the task, tool schemas, or visible contributions. Quoted passages retain their instruction wording; chat-template control tokens and transport headers are omitted, and line wrapping is for presentation.

A.1 TASK INSTRUCTIONS AND ANCHOR GENERATION

The anchor is generated from the current task, available tools, and committed interaction history, using the ordinary action-generation interface without contributor instructions. On BFCL, the system message contains the following task instructions and function schemas; the interaction history and current user message follow it.

```
You are an expert in composing functions.You are given a question and a
set of possible functions. Based on the question, you will need to make
one or more function/tool calls to achieve the purpose. If none of the
functions can be used, point it out. If the given question lacks the
parameters required by the function, also point it out.

You should only return the function calls in your response.

If you decide to invoke any of the function(s), you MUST put it in the
format of [func_name1(params_name1=params_value1,
params_name2=params_value2...), func_name2(params)].  You SHOULD NOT
include any other text in the response.

At each turn, you should try your best to complete the tasks requested by
the user within the current turn. Continue to output functions to call
until you have fulfilled the user's request to the best of your ability.
Once you have no more functions to call, the system will consider the
current turn complete and proceed to the next turn or task.

Here is a list of functions in json format that you can invoke.
{TOOL_SCHEMAS}
```

On ToolHop, tools are supplied through the tool-calling interface. The initial user message uses the benchmark's Mandatory instruction:

```
You will be asked a question with some tools, and should provide a short
final answer.
Please note that you must call the tool at every step, you must not use
your own knowledge. Your final answer must also be returned from the
tool.
If the final answer is a date, format is as follows: YYYY-MM-DD (ISO
standard)
If the final nswer is a name, format it as follows: Firstname Lastname
If the final answer contains any number, format it as a number, not a
word, and only output that number. Do not include leading 0s.

Please provide the final answer in the following format: <answer>final
answer here</answer>
Answer as short as possible.
Question: {CURRENT_USER_REQUEST}
```

The same benchmark task instructions are used for all evaluated methods. Subsequent task-facing messages contain the selected assistant responses and actual tool feedback.

A.2 CONTRIBUTION GENERATION

Contributors receive a quoted representation of the current task and interaction history, followed by the role instruction below. The visible ledger contains the anchor and previously admitted contributions. The grounding paragraph beginning "Use the user's request" is shared by the contributor, reviewer, and Final Judge. The BFCL contributor instruction is:

```
You are {CONTRIBUTOR_ID}.

ACTION - Propose a complete, executable next tool-call action grounded in
the current state and available tool schema.
EVIDENCE - Provide a grounded, decision-relevant reason that supports or
opposes a visible ACTION.
SYNTHESIS - Integrate the current state and visible contributions into a
new, decision-relevant conclusion that is not already stated; do not
merely restate or summarize existing content.

Use the user's request, tool schema, and actual execution history to
assess the current state. Contributions are proposals or inferences, not
new observations; acceptance does not establish truth, execution, or
success. Judge novelty relative to the current state and visible
contributions, not to how unusual an operation or its wording is.

Contribute one useful addition to the next decision, using the most
suitable IP type. Add a grounded action, reason, or conclusion rather
than merely repeating visible content. A new grounded reason to retain an
existing ACTION can be useful. Distinguish proposed actions and expected
results from completed actions and observed results. If no suitable
grounded, decision-relevant addition is available, return
{"type":"ABSTAIN"}. Return only one JSON object in one of the formats
below.

ACTION format:
{"type":"ACTION","action":"[func(arg=value)]","relation":"independent",␣
"target_action_id":null}
The action is a complete BFCL next-action batch, including multiple calls
when needed, or [] for a grounded stop. Use actual schema
function/argument names. relation=independent requires
target_action_id=null; relation=corrects requires an earlier visible
ACTION ID and a complete replacement correcting a grounded function or
argument error.

EVIDENCE format:
{"type":"EVIDENCE","target_action_id":"<earlier ACTION
ID>","stance":"support","claim_type":"schema","claim":"<one atomic
claim>","grounding_refs":["<official source reference>"]}
Replace format placeholders with actual content and a currently visible
ACTION ID. stance is support or oppose. claim_type is user_requirement,
schema, parameter_grounding, tool_observation, state_consistency,
execution_progress, or stop_or_scope. grounding_refs is the only optional
field: provide a non-empty list of precise references, or omit it.
EVIDENCE cannot contain or replace a tool call or target EVIDENCE,
SYNTHESIS, or a future contribution.

SYNTHESIS format:
{"type":"SYNTHESIS","synthesis":"<decision-relevant synthesis>"}
SYNTHESIS has no target and is advisory, not an embedded or replacement
tool call. Do not add fields or text outside the JSON object.

Visible ledger:
{VISIBLE_CONTRIBUTIONS}
```

The format examples specify contribution structure; their placeholders are filled using the current state. The literal {"type":"ABSTAIN"} object is a control output, not an additional contribution type.

ToolHop action format. The ToolHop interface permits either a tool-call batch or a final answer. Its ACTION definition is:

```
ACTION - Propose a complete next outward action-an executable tool-call
batch or a final answer-grounded in the current state and available tool
schema.
```

An ACTION retains four fields: `type`, `action`, `relation`, and `target_action_id`. The instruction for the response payload is:

```
The action field is a JSON string encoding exactly role="assistant",
content (text or null), and tool_calls (an ordered array). Each call has
exactly type="function" and function with name and arguments; arguments
is its original JSON-object string. Use actual schema function/argument
names. Do not include transport call IDs. For a final answer, use its
complete text in content and an empty tool_calls array. For a tool-call
batch, preserve all calls, their order and any accompanying content.
relation=independent requires target_action_id=null; relation=corrects
requires an earlier visible ACTION ID and a complete replacement
correcting a grounded response error.
```

The other contribution definitions and the shared instructions for grounding, type selection, and abstention are unchanged. Both backbones use these benchmark-specific response formats.

A.3 Peer Review

Each reviewer receives the same contribution definitions and shared grounding instruction as the contributor, together with the candidate and the ledger visible before that candidate. The reviewer-specific passages below specify the five judgments and required output fields. The candidate is fixed for all six reviews, which do not expose one reviewer's output to another.

```
You are {REVIEWER_ID}. Review only the frozen candidate from
{CANDIDATE_AUTHOR}, ID {CANDIDATE_ID}, independently of other reviewers.

Review this candidate against the current state and the contributions
visible before it. Evaluate the candidate's added content, not whether
you can certify the entire future trajectory. A routine operation may be
novel here; a paraphrase is not. A new grounded supporting reason may be
useful without changing the ACTION. Set each boolean according to its
definition, and use the reason to identify the added value or the
concrete defect. Return only the required JSON object; do not output a
score.

novelty:
Adds a decision-relevant action, reason, or conclusion relative to the
current state and visible contributions, rather than merely repeating
them.
effectiveness:
Can help choose, retain, revise, or appropriately end the next action in
service of the user's request; it need not guarantee success of the whole
task.
grounding_valid:
Is structurally valid and consistent with the tool schema and available
information; it does not present an unobserved result as an established
fact.
relation_valid:
For ACTION, independent has no target; corrects targets an earlier
visible ACTION and provides a full replacement correcting a grounded
function or argument error. For EVIDENCE, support or oppose relates to an
earlier visible ACTION. SYNTHESIS has no target; do not impose
ACTION/EVIDENCE relation fields on it.
non_recursive:
EVIDENCE gives a non-circular reason about a previously visible ACTION,
not about EVIDENCE, SYNTHESIS, or a future item. SYNTHESIS is target-free
advisory content, not an embedded executable action.
```

```
For EVIDENCE, grounding_refs is an optional locator: absence is not
automatic rejection and presence is not proof.

Required fields:
novelty, effectiveness, grounding_valid, relation_valid,
non_recursive: JSON booleans.
reason_code: a non-empty lowercase_snake_case string.
reason: a non-empty string giving a brief grounded reason.

Diagnostic checks (not gold):
{DIAGNOSTIC_CHECKS}

Contributions visible before this candidate:
{VISIBLE_CONTRIBUTIONS}

Frozen candidate raw output:
{CANDIDATE}

Existing structural parser result:
{CANDIDATE_PARSE}
```

Diagnostic checks provide parsing, function-availability, reference, and repetition information derived from the visible state and candidate. They are not benchmark correctness labels. For ToolHop, the ACTION definition uses the response interface in Section A.2, and the `relation_valid` description becomes:

```
For ACTION, independent has no target; corrects targets an earlier
visible ACTION and provides a full replacement correcting a grounded
response error. For EVIDENCE, support or oppose relates to an earlier
visible ACTION. SYNTHESIS has no target; do not impose ACTION/EVIDENCE
relation fields on it.
```

The remaining review instructions and output fields are unchanged. Reviews use JSON-object generation followed by local field and type validation; the vote is computed from the five Boolean judgments as described in Section B.3.

A.4 Final Judge

The Final Judge receives the benchmark's task-facing messages and visible ledger. Its decision instruction follows the shared grounding paragraph from Section A.2 in the system message. The BFCL decision passage is:

```
Choose the next complete executable action batch that best advances the
user's request from the current state. Use contributions for their
grounded, decision-relevant content. Retain justified parts of a visible
ACTION, and revise them when the current state or a grounded contribution
supports the change. The anchor is a candidate, not a mandatory action or
a default execution rule. You may produce an appropriate action that no
contributor proposed. Decide whether to continue or stop from the user's
requirements, available tools, and actual observations. Follow the
existing action-output interface; do not output contribution or review
JSON.

{VISIBLE_CONTRIBUTIONS}
```

For ToolHop, the opening phrase "Choose the next complete executable action batch" is replaced by "Choose the next complete outward response (a tool-call batch or a final answer)." The rest of the decision instruction is unchanged. The Judge returns through the benchmark's ordinary action interface, rather than the typed contribution or review interface.

B Implementation Details

B.1 Models and Decoding

We use Qwen3-4B-Instruct-2507 and Llama-3.1-8B-Instruct. Their pretrained weights remain unchanged. The Qwen checkpoint revision is `cdbee75f17c01a7cc42f958dc650907174af0554`. For Llama, we use the ModelScope distribution at revision `72956b202e77350459703a6bd652025126e6f8ab`. Inference uses vLLM 0.28.0 with BF16 weights and no quantization. Six NVIDIA L20 GPUs host independent single-GPU replicas, with parallelism applied across tasks. Within each experiment, every generation role uses the same backbone. Qwen's thinking mode is disabled; no thinking-specific request field is required for Llama.

Each generation has a total context limit of 131,072 tokens, an input budget of 126,976 tokens, and a maximum output of 4,096 tokens. AIP uses one anchor, six sequential contributors, six peers per reviewed candidate, and one independent Final Judge, with one deliberation round per state. Table 2 summarizes the role-specific decoding settings shared by both backbones.

Table 2: **Role-specific decoding settings for AIP.** Both backbones use the same settings. T denotes temperature, and a dash indicates that the request does not explicitly override the corresponding parameter.

Role	**Requested** T	**Effective** T	**Top-**p	**Seed**
Anchor	0.001	0.01	—	—
Contributor	0.7	0.7	1.0	Derived
Reviewer	0.001	0.01	—	Derived
Final Judge	0.001	0.01	—	—

Here, T denotes temperature and a dash denotes no explicit request override. The serving engine maps a requested temperature of 0.001 to 0.01. Top-k is not explicitly set; unspecified decoding fields use the serving defaults. Contributor and reviewer seeds are derived from base seed 42 and the task, turn, step, round, contributor, role, and reviewer identifiers. The derivation takes the first four bytes of a SHA-256 digest as a big-endian integer and masks it to 31 bits. The anchor and Judge have no explicit request seed. Replica identity and scheduling order do not enter the seed derivation.

B.2 Messages and Tool Interaction

BFCL uses textual Python call lists, with function schemas in the task system message and actual tool feedback incorporated into the interaction history. ToolHop uses assistant text and native tool calls, with one tool response associated with each call identifier. Model-specific chat templates encode this interface. For Llama on ToolHop, a text-preserving output adapter retains non-call content alongside the parsed function calls and preserves multiple-call order; it does not replace function names or arguments.

Contributor and reviewer requests use an active role instruction and a JSON-serialized task reference. Embedded role labels and instructions inside that reference are treated as quoted data rather than additional active chat messages. This separates the typed contribution interface from the benchmark's outward-response format while keeping task facts, tool schemas, and observations available. Internal roles do not receive executable tools. The Judge uses the outward-response interface and is the only AIP role whose response is submitted to the environment.

A submitted batch is fixed before execution, and calls are executed in their generated order. Feedback from one call does not change the already generated arguments of a later call in the same batch. On ToolHop, ordinary execution errors are returned as tool feedback; a ten-second tool timeout ends the batch and task. Each task has a single committed interaction history. Internal candidates are not trial-executed, and committed observations are not discarded through alternative environment branches.

B.3 Contribution Validation and Ledger Construction

The contribution parser checks that an output is a single JSON object with permitted fields, types, enumerations, and references to earlier visible ACTIONs. Duplicate keys and nonstandard constants are rejected. Recognized outer formatting is removed before parsing. BFCL ACTION checks cover call decoding, known function names, required-argument presence, and applicable path-format restrictions. ToolHop ACTION checks cover the assistant-response structure, argument objects, known functions, and required keys. These checks establish structural eligibility; peer review evaluates grounding, novelty, and decision relevance.

The review parser requires five JSON Booleans and the two reason fields shown in Section A.3. All five judgments must be true to produce a positive vote; otherwise, a valid review produces a negative vote. An additional score field, if generated, does not override the vote derived from the judgments. Malformed reviews are invalid votes rather than negative votes, and the explanation is not reparsed to substitute a different vote.

Credit is resolved after records from all six expected peer identities are present. Any valid negative vote gives negative credit; a nonempty set of valid votes that are all positive gives positive credit. An incomplete reviewer set or an empty valid-vote set leaves credit unresolved. A valid ABSTAIN receives zero credit, skips its six reviews, and is not admitted. Other malformed candidates are reviewed but remain structurally ineligible. The anchor is reviewed and remains visible regardless of credit.

Non-anchor contributions enter the ledger only when they satisfy the contribution-format and action-reference requirements, receive positive credit, and are not exact duplicates of visible ACTIONs. On BFCL, this check compares the complete ACTION string with the anchor and previously admitted ACTIONs in the same state, without whitespace removal or argument reordering. On ToolHop, the comparison uses a fixed serialization of the assistant-response payload without transport-only identifiers, retaining the text, original argument strings, and call order and multiplicity. It does not perform semantic matching and does not apply to EVIDENCE or SYNTHESIS. Duplicate candidates still receive reviews; the admission decision does not overwrite their peer-assigned credit.

The ledger retains each admitted contribution's full content, type, identifier, author, relations, order, and credit, along with the anchor. Subsequent contributors and the Judge receive this visible ledger, but not rejected candidates or raw review texts. Each decision state starts a new ledger. Long-horizon information is carried by executed actions and observations in the committed history, rather than by a separate persistent contribution memory.

B.4 Context Management and Stopping Rules

Input accounting uses the corresponding model's tokenizer and chat template and includes task history, tool schemas, generation prefixes, and internal contribution content. The input-budget check includes a safety allowance of $\max(256, \lceil 0.01n \rceil)$ tokens, where n is the rendered input length. The same context-management policy is used across backbones within each benchmark.

On BFCL, an input overflow triggers deterministic history compaction with successive character budgets of 24,000, 12,000, 6,000, and 3,000. The system message and function schemas are retained. The suffix beginning with the most recent user-role message is preserved, and earlier history is collected from the end; a boundary message retains its beginning and ending. BFCL tool feedback is also represented by user-role messages. A system notice identifies the compaction and includes up to 12,000 characters of successful-call history, instructing the agent to use the retained observations and avoid unnecessary repetition. Compaction uses no additional model generation and leaves the output allowance unchanged. A request that still exceeds the budget terminates the task. ToolHop does not use the BFCL character-compaction branches and terminates on an input-budget overflow.

BFCL permits at most 20 action decisions per user turn. An empty decoded call list, including `[]`, ends the current turn. Action-decoding failure also advances to the next turn, while the step or context limit ends the task. ToolHop permits at most nine outward responses, including the final answer. A response without tool calls ends the task; its nonempty text is retained as the final answer. Multiple calls in one submitted batch consume one outward decision, and internal deliberation does not consume these environment-interaction budgets. When generation reaches the output-token

limit, the returned content is processed by the ordinary parser without an additional continuation call.

B.5 Baseline and Ablation Settings

The seven baselines share the benchmark task instructions, outward-response interface, and interaction budgets with AIP. Table 3 summarizes their deliberation structures and request-level sampling parameters. Each method uses the same backbone for all of its generation roles.

Table 3: **Baseline deliberation and sampling settings.** Temperatures T and top-p values are request parameters; a dash indicates no explicit override. The same backbone is used across all generation roles within each method.

Method	**Deliberation structure**	T	**Top-p**
Vanilla	One outward response per state	0.001	—
SC	Five candidates; consistency-based selection	0.7	0.95
Reflexion	Actor and feedback-based memory updates	0.001	—
MAD	Three affirmative–negative rounds and one moderator	0.001	—
MacNet	Four-actor chain; three critic–actor exchanges per edge	0.001	—
MoA	Six proposers, six intermediate aggregators, one final aggregator	0.7	—
FoA	Fifteen proposal–memory paths and value-guided resampling	1.0	1.0

A dash again denotes no explicit override. Temperatures apply to the generation roles listed for each method, including reflection, critique, aggregation, and value estimation where relevant. Low-temperature requests use the effective value in Section B.1.

Reflexion updates its memory from executed actions and observations within the same task, without restarting the environment. Its BFCL memory interface permits up to 32 entries per field, 500 characters per entry, and a 60,000-character context. MacNet uses a 12,000-character overflow context; MoA uses a 12,000-character overflow context and 3,000 characters per reference. FoA uses a 120,000-character BFCL value context. These method-specific character-window branches are not applied to ToolHop.

FoA retains proposal lineage and cached values while restricting search to unexecuted candidates. It resamples 15 parents from eligible archived proposals every $k = 4$ outward decisions. Resampling weights are proportional to $\max(v_i, 0)\gamma^{t-t_i} + 10^{-6}$, where $\gamma = 0.5$, v_i is the proposal value, and $t - t_i$ is its age in outward decisions. Predicted values on a 1–10 scale are divided by ten, and unparseable values are assigned zero. The selected response alone is submitted, and later proposals use the resulting committed history.

Ablations. Each ablation shares the checkpoint, task set, interaction rules, and decoding settings of its AIP reference. Table 4 summarizes the contribution spaces, review settings, and per-state model-call counts. Action-only restricts the six non-anchor generation slots to ACTION or ABSTAIN while retaining peer review and the original Judge. The absence of a suitable new ACTION is sufficient for abstention in this condition. A contribution of a disallowed type is reviewed but not admitted.

The conditions without peer review retain structural and reference validation and the exact duplicate gate for non-anchor ACTION candidates. Visible entries have null credit and are explicitly marked as unreviewed with the instruction "Visible contributions in this condition have not received peer review." The Judge's decision instruction is otherwise unchanged. Anchor + Judge further removes all non-anchor contribution generation; the Judge is still invoked when the anchor is `[]`.

The counts apply to normally completed decision states; k_t is the number of valid contributor abstentions. AIP, Action-only, and w/o Peer Review permit abstention in their non-anchor slots. These counts include all internal generation calls, not only the response submitted to the environment.

Table 4: **AIP and ablation configurations.** The table lists the non-anchor contribution space, use of peer review, and total model calls per normally completed decision state. Here, k_t denotes the number of valid contributor abstentions.

Condition	Non-anchor contribution space	Peer review	Calls/state
AIP	ACTION / EVIDENCE / SYNTHESIS	Yes	$50 - 6k_t$
Action-only	ACTION	Yes	$50 - 6k_t$
w/o Peer Review	ACTION / EVIDENCE / SYNTHESIS	No	8
Anchor + Judge	None	No	2

B.6 Evaluation

All methods within each backbone–benchmark setting are evaluated on the same task set using the corresponding official evaluator. We use version 2026.3.23 of the official BFCL evaluator. ToolHop uses its official correctness implementation at revision `b439d7279af359fda46e8117ae4f0245b75f5c6b`, including final-answer matching and eligible tool feedback immediately preceding the final answer. Evaluation operates on recorded task-facing interactions rather than internal contributions or reviews.

Explicit tool errors are identified from BFCL tool feedback with a nonempty top-level `error` value and from ToolHop executor events with a recorded error. Repeated errors and errors from different calls in one batch are counted separately. Unexecuted candidates, model-service retries, and decoding failures before tool execution are not tool-error events. For both backbones, average tool errors per task are computed as the untrimmed arithmetic mean over all evaluated tasks. The same aggregation rule is used in the main experiments and ablations.

Each task contributes one recorded evaluation trajectory. Candidate ensembles and parallel model replicas are internal computation rather than independent task-level repetitions. Ordinary model and tool failures remain part of the evaluation. Methods share environment-interaction budgets and per-generation context and output limits, while their total inference computation follows their respective deliberation structures.

C Additional Case Study

C.1 Dependency-aware Action Revision

ToolHop task 9 requires identifying a play associated with *Suite from Henry V*, retrieving its author, and calculating the combined length of the author's first and last names. The anchor bundles an adaptation query with an author lookup for the placeholder "the play" and name extraction from an unverified author guess. An admitted SYNTHESIS recommends confirming the source play before relying on the author lookup. Although the anchor also receives positive credit, the Judge submits only the adaptation query. Subsequent actions use the returned title to retrieve William Shakespeare, extract both name components, and obtain their combined length, 18, from a tool. AIP returns 18 and passes the official evaluation.

The inspected Vanilla trajectory performs the unsuccessful placeholder lookup and later answers 16 despite obtaining the correct name components. AIP's revised sequence is consistent with the admitted dependency advice and ends with a tool-supported computation. This illustrates selective revision of the initial batch, rather than treating positive credit as a binding action choice; the record does not isolate the synthesis's causal contribution to the final outcome.

D Additional Ablations

We extend the main ablation study with two targeted tests of novelty-guided admission and contributor-to-contributor sharing. Both use Qwen3-4B-Instruct-2507 on BFCL v4 multi-turn, with the same task set, decoding settings, 131,072-token context budget, 4,096-token output limit, and interaction rules as their AIP reference. We reuse the AIP reference from the main ablation study.

Both variants retain all generation roles and six peer reviews per non-abstaining candidate, without changing the static role prompts. Table 5 summarizes the results.

Table 5: **Additional ablations of AIP with Qwen3-4B-Instruct-2507 on BFCL v4 multi-turn.** Accuracy is expressed as a percentage, and tool-error values are untrimmed task means. Both variants retain the contribution-generation and peer-review stages and the Final Judge. Best metric values are bold.

Condition	Overall Acc. (%) ↑	Avg. Tool Errors per Task ↓
AIP	**17.50**	**1.750**
w/o Novelty Veto	11.25	2.175
w/o Contributor Sharing	12.50	3.000

D.1 Novelty-Guided Admission

w/o Novelty Veto retains all five review judgments and both reason fields but excludes novelty when deriving the admission vote for non-anchor contributions. All fields remain subject to the original parser. With novelty indexed by $k = 1$, a valid review yields

$$\widetilde{v}_{t,i}^{(j)} = 2 \prod_{k=2}^{5} q_{t,i,k}^{(j)} - 1, \qquad i = 1, \ldots, 6.$$

Anchor scoring, six-peer credit aggregation, structural and reference checks, and the exact ACTION-duplicate gate remain unchanged. The ablation therefore tests the novelty veto at admission, not the removal of novelty instructions or exact deduplication.

Accuracy decreases from 17.50% to 11.25%, while average tool errors rise from 1.750 to 2.175. On this variant's recorded candidates and reviews, 317 admitted contributions, including 288 syntheses, would be excluded by the original five-judgment rule. This within-run rule comparison shows that novelty affects admission beyond exact ACTION matching; it is not a replay of AIP's trajectory. The task-level results favor retaining novelty in contribution selection.

D.2 Sharing Contributions During Generation

w/o Contributor Sharing restricts the generation inputs of contributors B–G to the current state and anchor:

$$c_{t,i} \sim \pi_\theta^{\mathrm{C}}\big(\cdot \mid S_t, b_t, \varnothing\big), \qquad i = 1, \ldots, 6.$$

The slots remain sequential. Reviewers see the full admitted prefix, and the Final Judge receives the complete final ledger. Contributors may reference only the visible anchor, while exact-duplicate checks still use the global admitted ACTION set. Thus, peer assessment and final aggregation remain intact.

Accuracy decreases to 12.50%, and average tool errors increase to 3.000. The variant still admits 149 EVIDENCE and 2,276 SYNTHESIS contributions for the Judge, but no non-anchor ACTIONs. The observed accuracy–error pattern is consistent with a benefit from exposing contributors to earlier admitted additions during generation, beyond making those additions available only to reviewers and the Judge.

Interpretation and inference budget. Both additional ablations retain AIP's generation roles, peer-review structure, and normal $50 - 6k_t$ calls per state. Their total input-plus-output token counts are 676.5M and 664.2M, compared with 692.8M for AIP, placing both within 4.2% of the reference token volume. AIP attains higher accuracy and lower tool-error means in both comparisons. These interventions therefore examine admission and information sharing while retaining the deliberation structure, although total inference compute is not exactly matched. Paired two-sided exact McNemar tests yield raw p-values of 0.267 and 0.344; Holm correction across the five exploratory ablation comparisons gives $p = 0.801$ for both. The results provide consistent directional evidence within the evaluated setting, rather than statistically significant accuracy differences.