COMPOSITIONAL RECURSIVE SELF-IMPROVEMENT

Anonymous authors
Paper under double-blind review

ABSTRACT

Recursive self-improvement is becoming an important direction for building agents that can learn from their own experience and improve their future behavior. Yet current RSI systems do not clearly expose which parts of a successful harness should be retained, how compatible discoveries from different executions should be combined, or how the value of one control changes with its collaborators. Whole-harness updates couple these decisions and make reusable improvement units difficult to preserve. We address this gap with CoRSI, a compositional harness-adaptation method for RSI systems. CoRSI retains typed harness modules while evaluating the complete assembled harness: base inheritance preserves unchanged controls, guarded assembly combines compatible discoveries, and cooperation memory informs later task-conditioned composition. Across multiple domains, datasets, and baseline systems, CoRSI improves transfer performance and rubric-scored analysis quality while producing traceable workflow and artifact evidence. The method provides a compositional interface for retaining and reusing control improvements in recursive self-improvement systems.

1 INTRODUCTION

Recursive self-improvement is becoming an important direction for building agents that learn from their own execution. Feedback loops revise responses after critique, experience systems retain skills for later tasks, and agent frameworks expose planning, tools, routing, and coordination as reusable controls (Madaan et al., 2023; Shinn et al., 2023; Wang et al., 2024; Yao et al., 2023; Wu et al., 2023; Hong et al., 2024). Recent systems also search over workflows and agent programs, extending adaptation from answer generation to the programs that organize and evaluate future behavior (Khattab et al., 2024; Zhang et al., 2024; 2025a; Hu et al., 2024; Fernando et al., 2023; Zelikman et al., 2023; Zhang et al., 2025b). However, these advances do not make clear which parts of a successful harness should remain available for later composition. For a deployed agent, this question concerns the harness around the model: context, tools, generation, orchestration, memory, and verification determine what the agent can do, what it can retain, and what it accepts (Yang et al., 2024; Lee et al., 2026; Ren et al., 2026).

Current RSI systems leave three related questions implicit. Which parts of a successful harness should survive after an execution? How can compatible discoveries from separate executions be combined without importing unrelated changes? How does the value of one control depend on the other controls with which it is executed? Whole-harness updates couple these decisions: a complete candidate can express a rich team or workflow, but a local improvement may be discarded when another field regresses, and two compatible candidates compete for one retained successor. Experience retrieval preserves episodes and reflections, but does not identify the smallest reusable harness control (Shinn et al., 2023; Wang et al., 2024). Workflow and program search expose richer execution structures, but generally retain a complete program or configuration rather than a typed control that can be reused independently (Zhang et al., 2024; 2025a; Khattab et al., 2024; Lee et al., 2026). This is also a contextual evaluation problem: cooperative coevolution shows that a component's value is determined by the complete solution formed with its collaborators (Potter & De Jong, 1994).

We address this gap with CoRSI, a compositional harness-adaptation method for RSI systems. CoRSI separates the modification unit from the evaluation unit. The modification unit is a typed harness module that records a local control change; the evaluation unit is the complete assembled harness or workflow that receives the task outcome. Base inheritance preserves every field that is not changed by a selected module. Guarded assembly then combines compatible modules from

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.

different executions, routes their named artifacts through the declared workflow, and evaluates the complete behavior. Cooperation memory records successful module partnerships and makes this history available to later task-conditioned composition. These operations preserve the locality of a discovery while keeping the reward attached to the complete behavior that a user observes. The same interface supports independent participants, sequential artifact handoffs, and task-specific workflow structures and dependency topologies; the benchmark adapter determines how the assembled workflow runs.

We study this interface across mathematics, scientific question answering, cross-model transfer, and biomedical data analysis. The experiments combine frozen-bank transfer, mechanism ablations, round-resolved state records, and execution-level traces. Our contributions are:

- Task-conditioned recall that improves retrieval precision by matching module content and cooperation history to the current task and base harness.
- A bounded recall design that controls how many candidate modules enter a composition, keeping the search space focused while retaining complementary discoveries.
- A compositional recall-and-assembly procedure that combines typed field changes across sources, inherits untouched fields, and materializes selected modules as participants, artifact routes, and task-specific topologies.
- Experiments across mathematics, scientific question answering, cross-model transfer, and biomedical data analysis, including frozen-bank transfer, mechanism ablations, round-resolved records, and execution traces.

Section 3 defines the composition interface; Section 4 evaluates transfer and mechanism effects; and Appendix D.1 with the remaining appendices connects the measurements to executed workflows, retained modules, and cooperation structure.

2 Related Work

The update-unit question connects two lines of work: agent systems specify what can be composed, and RSI systems determine what survives feedback. We position `CoRSI` at their intersection; Appendix E traces the progression from local revisions to editable improvement operators.

Agent Systems. Surveys of autonomous agents distinguish planning, tool use, memory, and multi-agent coordination as coupled system components (Wang et al., 2023). ReAct and Toolformer make reasoning–action interaction and tool selection explicit (Yao et al., 2023; Schick et al., 2023); SWE-agent demonstrates the importance of the agent–computer interface (Yang et al., 2024). AutoGen and MetaGPT coordinate specialized agents through conversations and structured artifacts (Wu et al., 2023; Hong et al., 2024). DSPy optimizes declarative programs, while graph-of-thoughts and graph-based agent design expose intermediate structure, communication, and control flow as optimization variables (Khattab et al., 2024; Besta et al., 2023; Zhuge et al., 2024). Workflow-generation systems search executable programs and structural alternatives (Zhang et al., 2024; 2025a; Wang et al., 2026); recent surveys organize these systems around planning, routing, memory, and graph structure (Feng et al., 2026b; Bei et al., 2025). `CoRSI` makes the configuration inside each participant compositional: the planner jointly selects typed field patches, participant cardinality, and an artifact topology, and evaluates the resulting complete harness.

Recursive Self-Improvement. Self-Refine and Reflexion turn feedback into answer revisions and reusable reflections (Madaan et al., 2023; Shinn et al., 2023). Voyager and ExpeL retain skills and distilled experience across tasks (Wang et al., 2024; Zhao et al., 2023). Promptbreeder and STOP extend adaptation to mutation prompts and the improvement program itself (Fernando et al., 2023; Zelikman et al., 2023); ADAS, the Darwin Godel Machine, and Hyperagents explore progressively richer editable agent and meta-agent programs (Hu et al., 2024; Zhang et al., 2025b; 2026). Recuris studies validation-gated working and experiential memory updates (Yu et al., 2026). MemoHarness decomposes the harness into editable control dimensions and retrieves execution experience for later cases (Huang et al., 2026); Meta-Harness studies end-to-end optimization of the harness itself (Lee et al., 2026). Continual harness learning and skill-use studies further move adaptation into the execution scaffold (Kang et al., 2026; Han et al., 2026). These approaches differ in their update targets and feedback mechanisms, as organized in recent surveys (Ren et al., 2026). `CoRSI`

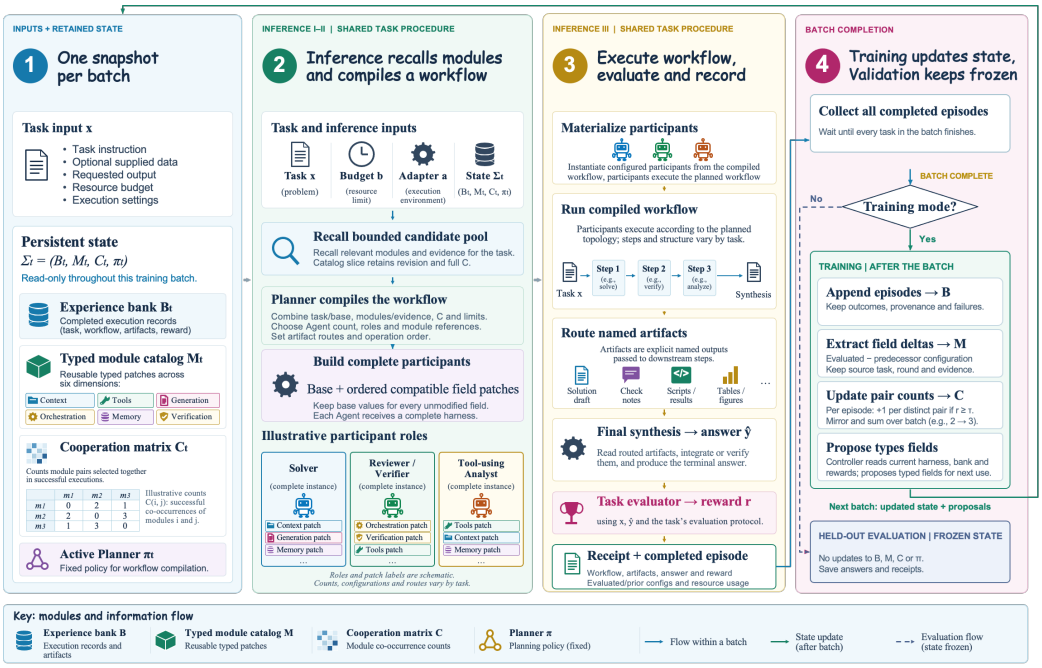


Figure 1: Overview of CoRSI training and inference.

focuses on cross-candidate module recombination and collaborator-conditioned evaluation, drawing on cooperative coevolution (Potter & De Jong, 1994). Appendix E develops this progression from local feedback to recursive improvement operators.

3 Compositional Harness Adaptation

CoRSI separates the unit of change from the unit of evaluation. A complete harness is factorized into typed modules along six dimensions: Context, Tools, Generation, Orchestration, Memory, and Verification. The Planner uses this factorization to choose how many participants to instantiate, which modules to assign to each participant, and how their workflow is connected. Every recalled change is applied to a clone of the base harness; fields without a recalled change inherit the base configuration, and the evaluator scores one complete harness after execution. The detailed data schema, validation rules, and adapter contracts are given in Appendix A.1. Let $\mathcal{D} = \{\text{Context}, \text{Tools}, \text{Generation}, \text{Orchestration}, \text{Memory}, \text{Verification}\}$ denote the typed dimensions, let $H_0 = (h_d)_{d\in\mathcal{D}}$ be the base harness, and let a module be a typed patch $m = (d, p_m)$. A selected module set S produces a complete harness by inheritance and composition:

$$H(S) = H_0 \oplus S, \qquad h_d(S) = \begin{cases} p_m, & (d, p_m) \in S, \\ h_d, & d \notin \mathrm{dom}(S), \end{cases} \tag{1}$$

so the reusable object is a typed patch while the reward-bearing object is the decoded complete harness.

3.1 Training and Inference

Training and inference share the same composition path and differ in what happens after evaluation. In training, each completed episode is retained in an experience bank. Typed field changes become reusable module entries, and successful module co-occurrences update the cooperation memory. The retained state and its round transition are summarized by

$$\Sigma_t = (\mathcal{B}_t, \mathcal{M}_t, C_t), \qquad \Sigma_{t+1} = \mathcal{T}(\Sigma_t, \mathcal{Q}_t), \tag{2}$$

where $\mathcal{Q}_t$ contains the completed episodes of round t. For a successful episode with selected module set S_q, reward r_q, and threshold τ, the cooperation memory records pairwise evidence as

$$C_{t+1}(u, v) = C_t(u, v) + \mathbf{1}[(u, v) \subseteq S_q]\mathbf{1}[r_q \geq \tau]. \tag{3}$$

The next round therefore sees the accumulated bank, module catalog, and cooperation history; successful combinations can be retrieved as new candidates for later Planner calls.

In inference, the retained state is frozen. For a new task, the Planner receives the task summary, base-harness summary, bounded recall pool, cooperation statistics, and workflow budget, and returns a typed workflow program:

$$P_t = \mathrm{Planner}(Z_t(x; \Sigma_t)), \qquad H_t^\star = \mathrm{Decode}_{H_0}(P_t). \tag{4}$$

The program specifies participant roles, module assignments, artifact dependencies, and operation order. Each participant is materialized as $H_{j,t} = H_0 \oplus S_{j,t}$, guarded compatibility checks are applied, and the declared workflow routes named artifacts before the benchmark evaluator returns the final reward:

$$(\widehat{y}_t, r_t) = \mathrm{Eval}_x(\mathrm{Exec}(P_t, H_t^\star)) \,. \tag{5}$$

This keeps the evaluation unit identical in both phases while allowing the training phase to expand the reusable state used by subsequent inference. Conflict rules, artifact semantics, and contextual interaction definitions are collected in Appendix A.1 and Appendix C.

3.2 Compositional Harness Adaptation for RSI

`CoRSI` gives an RSI system a compositional harness interface: the reusable object is a compatible typed module while the evaluated object remains a complete harness. The method records field-level changes with base inheritance, assembles compatible changes from multiple candidates into one executable workflow, and uses cooperation history to condition later Planner compositions.

A complete workflow gives every module its value in context. The same Context, Generation, Orchestration, Memory, Tools, and Verification field can support different outcomes when it is paired with different collaborators or routed through a different artifact graph. `CoRSI` records these pairwise outcomes and feeds them back into module recall, participant assignment, and workflow compilation. Successful compositions therefore become reusable building blocks for subsequent tasks.

Cooperative coevolution supplies the corresponding perspective: components are evaluated through the complete solution formed with their collaborators (Potter & De Jong, 1994). `CoRSI` extends this principle to typed harness modules, task-conditioned participant groups, artifact dependencies, and complete-harness evaluation. The formal construction, contextual marginal value, and bank-mediated state transition appear in Appendix C; the experiments connect them to transfer, case-level artifacts, cooperation matrices, and round-resolved bank growth.

4 Experiments

We examine whether configurations learned on earlier tasks transfer across competition editions, scientific domains, base models, and tool-based analyses. Four benchmark families expose these forms of reuse under frozen banks. The empirical study varies the execution condition while keeping the learned bank and task-level evaluation contract explicit: **RQ1:** How does performance change from in-distribution (IID) tasks to out-of-distribution (OOD) tasks with new scientific content or task structure? **RQ2:** Can the same retained controls be reused when the task category, problem family, or competition edition changes? **RQ3:** Can a bank learned with one model be executed effectively after the base model changes? **RQ4:** Can automated harness adaptation surpass agents designed by human experts with domain knowledge and tools in professional scientific domains? **RQ5:** Which recalled modules and workflow mechanisms contribute to the held-out result when each is removed from the composition path? **RQ6:** Does the training process produce progressively stronger complete workflows across rounds on a demanding case study?

Setup. The main study compares frozen-bank transfer on mathematics, GPQA-Diamond, CMT-Bench, and BiomniBench-DA. We report exact-match accuracy for the first three and rubric scores for biomedical analysis; the detailed task splits, models, evaluators, budgets, and baseline records are

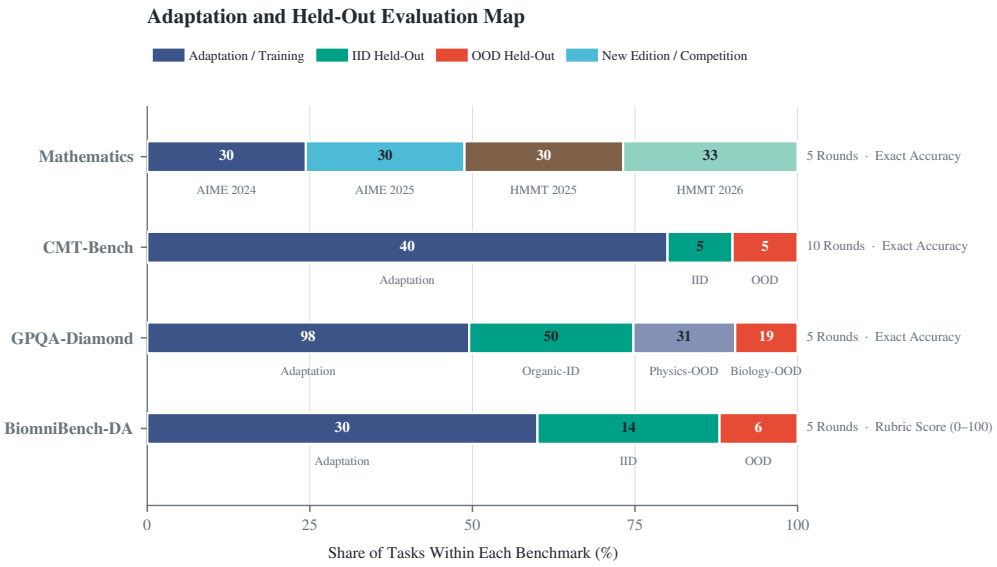


Figure 2: Adaptation and held-out evaluation panels across benchmark families.

collected in Appendix B.1. We call a held-out item Domain IID when it remains within the domain represented by the adaptation panel, and Domain OOD when its scientific or task domain changes. Within a domain, Sub-domain IID preserves the adaptation sub-domain and task structure while using new items; Sub-domain OOD keeps the broad domain but changes the sub-domain or problem family. Mathematics and CMT-Bench use their native competition and category transfer partitions, while GPQA-Diamond and BiomniBench-DA expose the domain and sub-domain distinctions directly. Figure 2 summarizes these adaptation and held-out panels; the complete item-level partition rules are given in Appendix B.1.2.

4.1 RQ1: IID-to-OOD transfer across scientific content

We study RQ1 on GPQA-Diamond, which contains organic chemistry, physics, and biology questions. The held-out partition contains Organic-ID questions from the adaptation-covered subdomain, Physics Subdomain-OOD questions from held-out physics subdomains, and Biology Domain-OOD questions from a new scientific domain. We train the bank for five rounds on the fixed adaptation panel and then evaluate the frozen bank on 100 held-out questions. We compare `CoRSI` with non-training agent-harness records, including Codex and Claude Code, using the same task-level evaluator. Figure 3 reports the six-model comparison on 100 held-out GPQA-Diamond questions (Rein et al., 2023). Organic-ID contains 50 new organic-chemistry questions from an adaptation-covered subdomain; Physics Subdomain-OOD contains 31 questions from held-out physics subdomains; Biology Domain-OOD contains 19 questions from an entirely held-out domain.

Across models, `CoRSI` improves the held-out GPQA endpoint over MemoHarness in every reported setting, with the clearest separation on Organic-ID and Biology Domain-OOD. The pattern is consistent with two forms of reuse: controls transfer within a familiar scientific structure and remain useful when the domain changes. The figure is therefore read as evidence for content transfer, with the model spread showing how strongly each executor can use the retained controls.

The domain split distinguishes reuse from extrapolation: Organic-ID keeps the task structure while Biology Domain-OOD changes the scientific domain. The same bank helps in both settings, supporting the claim that `CoRSI` stores transferable controls rather than answers tied to one question set. The GPQA ablation results in Appendix B.1.7 attribute this behavior to module granularity and workflow context; the complete partition rules, baseline records, and executed ablation details are in Appendix B.1.2–B.1.7.

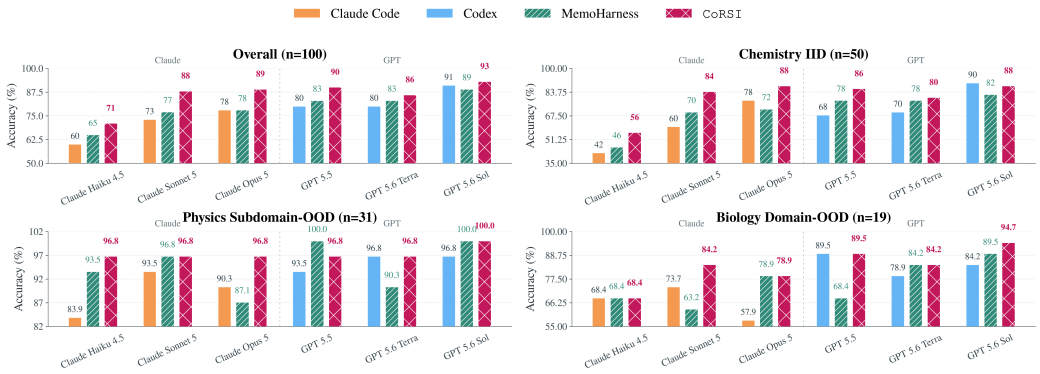


Figure 3: GPQA-Diamond accuracy across held-out transfer settings.

Table 1: Results on held-out mathematics benchmarks.

Methods	AIME 2025				HMMT 2025				HMMT 2026			
	CoRSI	MemoHarness	Codex	Claude Code	CoRSI	MemoHarness	Codex	Claude Code	CoRSI	MemoHarness	Codex	Claude Code
GPT-5.6 Sol	**100.0%**	96.7%	93.3%	-	**100.0%**	86.7%	96.7%	-	**100.0%**	78.8%	93.9%	-
GPT-5.6 Terra	**100.0%**	93.3%	70.0%	-	**100.0%**	90.0%	96.7%	-	**100.0%**	78.8%	84.8%	-
GPT-5.5	**100.0%**	90.0%	83.3%	-	**100.0%**	93.3%	86.7%	-	**100.0%**	84.8%	75.8%	-
Claude Haiku 4.5	**90.0%**	70.0%	-	70.0%	**80.0%**	56.7%	-	53.3%	**69.7%**	48.5%	-	45.5%
Claude Sonnet 5	**83.3%**	63.3%	-	80.0%	**70.0%**	56.7%	-	50.0%	**75.8%**	51.5%	-	60.6%
Claude Opus 5	**86.7%**	76.7%	-	70.0%	**76.7%**	63.3%	-	73.3%	**72.7%**	57.6%	-	63.6%

4.2 RQ2: Reuse across task families and competition editions

We study RQ2 on mathematical competition tasks. The adaptation panel is AIME 2024, while the held-out panels are AIME 2025, HMMT 2025, and the pinned HMMT 2026 snapshot, covering both a new edition and a different competition family. We train the bank for five rounds on AIME 2024 and then freeze it for held-out evaluation. CoRSI is compared with non-training agent harnesses such as Codex and Claude Code under the same exact-match evaluator. Table 1 evaluates banks trained for five rounds on AIME 2024. The test sets are AIME 2025 (30 questions), HMMT 2025 (30), and the pinned HMMT 2026 snapshot (33). Across the three held-out panels, CoRSI has the higher aggregate accuracy and the advantage is repeated across model settings. The important result is that the bank learned on AIME 2024 transfers to both a new edition and a different competition family. The exact-state and verification cases in Appendix D.1 shows how this transfer is implemented: reusable controls preserve a derivation while independent participants check it before finalization. Additional task-level traces and participant walkthroughs are in Appendix D.1 and Appendix D.5.

4.3 RQ3: Reuse after changing the base model

We study RQ3 on CMT-Bench, whose adaptation panel covers some task categories and whose held-out panel contains both new tasks from covered categories and tasks from novel categories. The bank is trained for ten rounds and then executed with a changed base model on five IID and five OOD tasks. We compare CoRSI with corresponding non-training agent-harness records using the same task set and exact-match evaluator. Figure 4 shows performance on five unseen tasks from training-covered categories and five from novel categories. Public CMT-Bench outputs are rescored on this same task set. Each partition therefore reports an exact five-task count. CoRSI reaches the same 40% endpoint on the IID and OOD partitions for both GPT-5.5 and Sonnet 4.6. The result matters because one bank crosses both a category boundary and a model boundary without changing its typed interface. The cross-model case explains the mechanism: transferred controls organize an inventory that a downstream synthesis step can check. The complete task records, transfer protocol, and harness-level evidence are provided in Appendix D.1.4 and Appendix A.2.

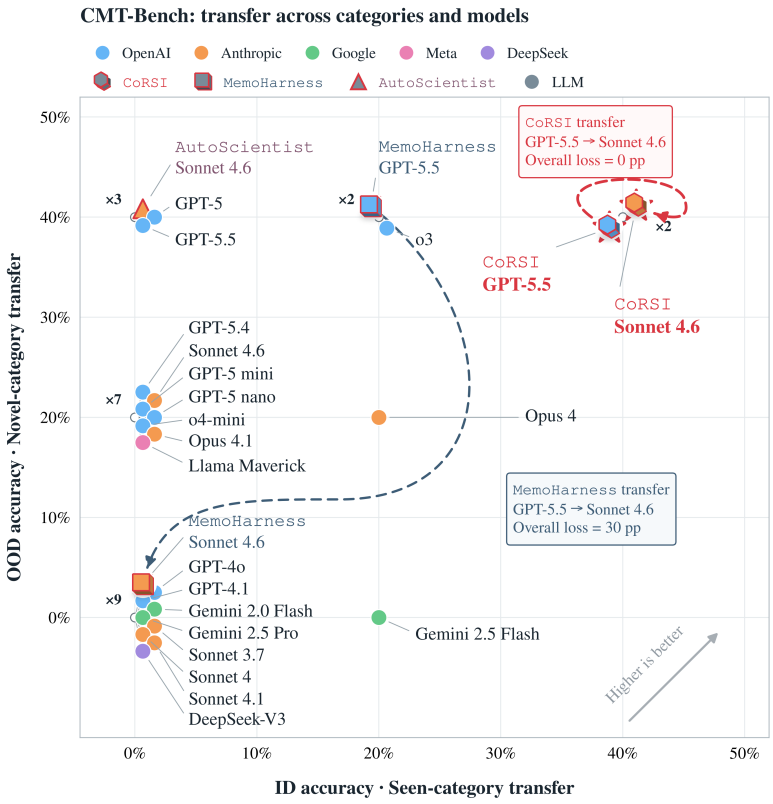


Figure 4: CMT-Bench transfer across categories and models. Shapes denote methods and colors denote vendors.

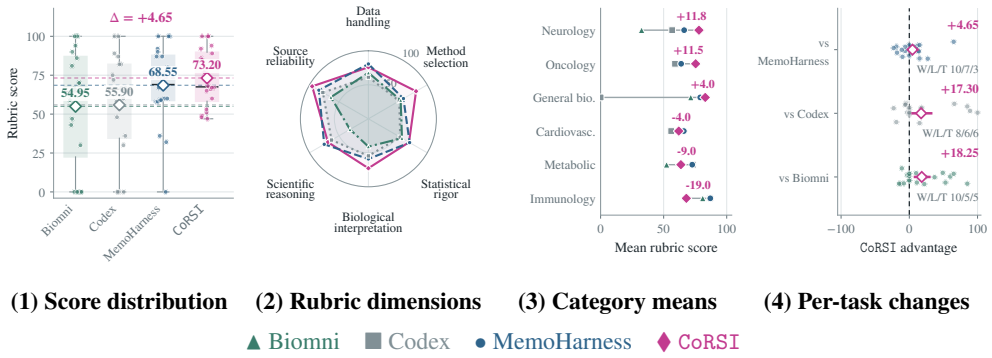


Figure 5: BiomniBench-DA quality and paired comparisons on held-out tasks.

4.4 RQ4: Automated adaptation in professional scientific domains

We study RQ4 on professional scientific data-analysis tasks evaluated with a process-level rubric. The adaptation and held-out panels remain within the biomedical domain while varying analysis categories and task instances. We train the complete workflow across the recorded rounds, freeze the resulting bank, and compare its held-out rubric scores with non-training scientific-agent baselines equipped with domain knowledge and tools. On the frozen panel, CoRSI has the highest mean rubric score among the compared internal conditions, and its OOD gain is larger than its IID gain. The figure is read through the rubric: the endpoint summarizes quality, while the dimensions identify which parts of the evidence chain changed.

The rubric profile concentrates the gain in method selection, biological interpretation, and source reliability. These dimensions correspond directly to the typed Context, Orchestration, and Verification controls, so the score change has a workflow interpretation rather than being only an aggregate improvement. The threshold curves show how the gain is distributed across partially completed analyses, while the external panel places the result in the wider biomedical landscape. Criterion-level cases then trace this coverage to cohort construction, metric computation, and interpretation, showing how routed artifacts survive into the final scientific claim. The rubric definitions, threshold analyses, and criterion-level records are given in Appendix D.3.1–D.3.4. The criterion-level cases trace the score change to cohort construction, metric computation, and interpretation: an intermediate result matters when it survives into the final conclusion. The resource cases show the same principle at execution level: the Planner changes workflow depth with the demand, adding participants when a longer evidence chain is useful and keeping the path short when finalization is sufficient.

4.5 RQ5: Contributions of recalled modules and workflow mechanisms

We design the ablation study to isolate the mechanisms that turn recalled modules into a transferable complete workflow. Each condition disables one operation and is retrained before evaluation. Figure 6 reports the change from the complete condition separately for AIME 2025 (Panel A) and HMMT 2025 (Panel B), allowing the same mechanism to be examined under edition transfer and competition-family transfer.

The cooperation variants remove the accumulated cooperation memory or preserve its counts while shuffling the module-pair assignments. The representation variants restrict assembly to one history source, limit each participant to one recalled module, or replace field-level patches with coarser dimension-level changes. The workflow variants remove module-guided workflow selection, the task-conditioned workflow context, or multiple participants. The execution variants remove intermediate artifact handoff or final synthesis, while the retrieval variant removes experience retrieval but retains the typed catalog. Together, these variants correspond to the cooperation, representation, workflow, execution, and retrieval markers shown in the figure.

Most variants produce a decrease for at least one of the two models, although the decrease is not uniform across transfer settings. In AIME 2025, several variants remain at the complete-condition endpoint for one model, while others already change the score. We therefore carry the apparently neutral cases into the HMMT 2025 analysis. In Panel B, removing cooperation memory lowers the Terra/Haiku endpoints by 16/12 points, removing module-guided workflow lowers them by 16/14 points, and removing the multi-participant workflow lowers them by 21/15 points. These results indicate that the mechanisms can remain interchangeable when the bank transfers within one competition family, but lose that substitutability when the workflow must coordinate a broader competition-family transfer. Cooperation memory can help identify module partnerships that have worked together; module-guided workflow can help turn recalled modules into a task-appropriate organization; and multiple participants can preserve complementary derivations and expose them to later synthesis. The observed HMMT drops are consistent with these roles becoming more valuable when transfer requires broader coordination. The executed configurations, and verification protocol are listed in Appendix B.1.7 and Appendix B.3.

4.6 RQ6: Progressive improvement of complete workflows

We study RQ6 on the biomedical analysis tasks used for the round-wise case study. The training panel is evaluated over ten rounds, and each round produces a complete workflow whose full-rubric score is recorded. We compare the round-wise `CoRSI` curve with the released non-training scientific-agent baselines shown as horizontal reference levels; the round-wise records and cooperation matrices are given in Appendix B.2.

The key pattern is the crossing of the reference levels: `CoRSI` starts below the strongest external agent and reaches 20.0% full-rubric success by Round 5, above the 15% Biomni reference and the other displayed agents. This makes the plot a direct training case study: successive bank updates give the Planner workflows that solve more complete biomedical analyses than the fixed expert agent configuration.

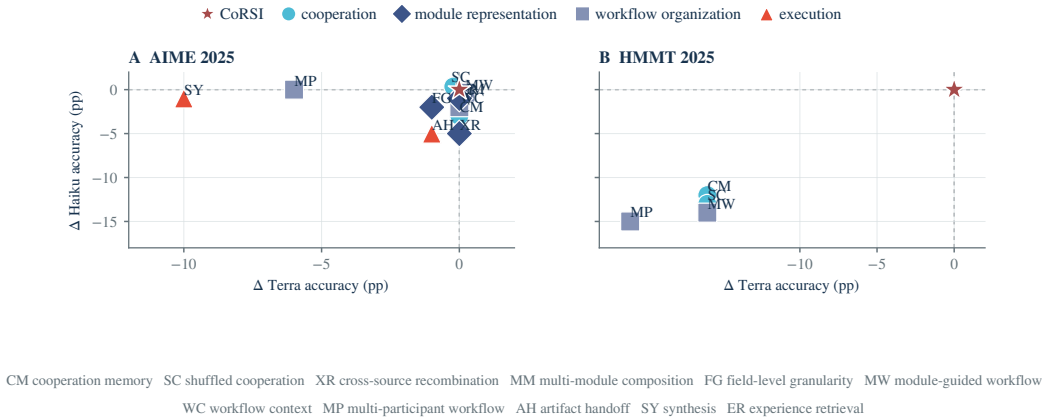


Figure 6: Ablation effects relative to the complete `CoRSI` condition. The star marks the reference; each point is one ablation.

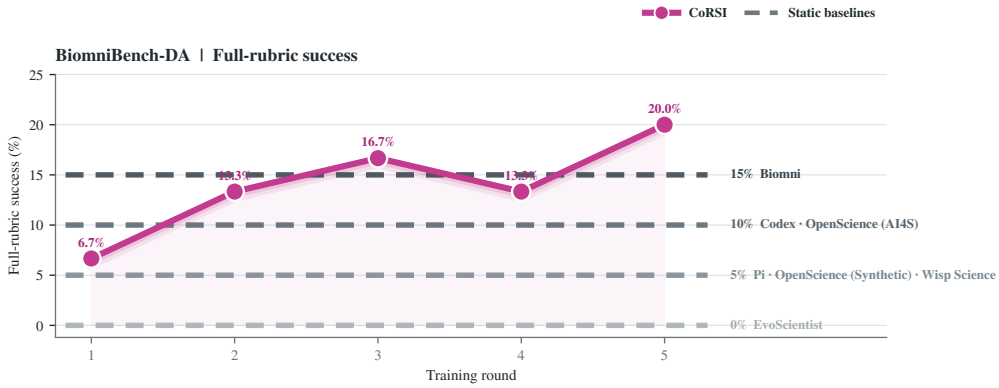


Figure 7: Round-wise full-rubric success on BiomniBench-DA. The magenta curve shows CoRSI and dashed segments show static baselines.

The curve is interpreted together with the appendix state records: bank content and cooperation structure grow as the training process adds new executable compositions. The held-out results then measure the accumulated state on new tasks, while this case shows the improvement emerging during adaptation. The round-resolved state, cooperation matrices, and additional case records are provided in Appendix B.2 and Appendix D.3.5.

5 Conclusion

We presented `CoRSI`, a compositional harness-adaptation method for recursive self-improvement systems that retains typed control modules, recombines compatible improvements, and evaluates the resulting complete workflows. Across mathematical, physical-science, and biomedical tasks, `CoRSI` improves performance over the corresponding non-training agent systems and transfers learned controls across tasks, domains, and model settings. In the biomedical case study, continued training produces workflows that eventually outperform the evaluated expert-designed scientific agent, showing the practical value of compositional harness adaptation for recursive self-improvement.

REFERENCES

Yuanchen Bei, Weizhi Zhang, Siwen Wang, Weizhi Chen, Sheng Zhou, Hao Chen, Yong Li, Jiajun Bu, Shirui Pan, Yizhou Yu, Irwin King, Fakhri Karray, and Philip S. Yu. Graphs Meet AI Agents: Taxonomy, Progress, and Future Opportunities, 2025.

Maciej Besta, Nils Blach, Ales Kubicek, Robert Gerstenberger, Michal Podstawski, Lukas Gianinazzi, Joanna Gajda, Tomasz Lehmann, Hubert Niewiadomski, Piotr Nyczyk, and Torsten Hoefler. Graph of Thoughts: Solving Elaborate Problems with Large Language Models, 2023. URL https://arxiv.org/abs/2308.09687.

Tao Feng, Haozhen Zhang, Zijie Lei, Peixuan Han, and Jiaxuan You. GraphPlanner: Graph memory-augmented agentic routing for multi-agent llms, 2026a.

Yuyuan Feng, Zhishang Xiang, Chaobin Yang, Qichao Ma, Zerui Chen, Yujing Zhang, Ke Huang, Chuanjie Wu, Zhaoxu Liu, Yili Wang, Xin He, Jiapu Wang, Zijin Hong, Hao Chen, Yuanchen Bei, Kun Wang, Shengyuan Chen, Ningyu Zhang, Enyan Dai, Linhao Luo, Qingyi Pan, Qi Wang, Wenqi Fan, Guangjing Wang, Na Zou, Yangqiu Song, Xin Wang, Zechao Li, Xia Hu, Qing Li, Xiao Huang, Zhihong Zhang, Jinsong Su, Qinggang Zhang, and Yi Chang. Graph engineering in the era of llm agents: From individual intelligence to system intelligence, 2026b.

Chrisantha Fernando, Dylan Banarse, Henryk Michalewski, Simon Osindero, and Tim Rocktäschel. Promptbreeder: Self-Referential Self-Improvement Via Prompt Evolution, 2023. URL https://arxiv.org/abs/2309.16797.

Jinyi Han, Yuanjian Xu, Ying Liao, Xinyi Wang, Zishang Jiang, Zixiang Di, Fanyang Lu, Zhichao Hu, and Yanghua Xiao. Skill-use: Can llms actually use skills in agentic harnesses?, 2026.

Sirui Hong, Mingchen Zhuge, Jiaqi Chen, Xiawu Zheng, Yuheng Cheng, Ceyao Zhang, Jinlin Wang, 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 *International Conference on Learning Representations*, 2024. URL https://arxiv.org/abs/2308.00352.

Shengran Hu, Cong Lu, and Jeff Clune. Automated Design of Agentic Systems, 2024. URL https://arxiv.org/abs/2408.08435.

Kexin Huang, Serena Zhang, Hanchen Wang, Yuanhao Qu, Yingzhou Lu, Yusuf Roohani, Ryan Li, Lin Qiu, Gavin Li, Junze Zhang, Di Yin, Shruti Marwaha, Jennefer N. Carter, Xin Zhou, Matthew Wheeler, Jonathan A. Bernstein, Mengdi Wang, Peng He, Jingtian Zhou, Michael Snyder, Le Cong, Aviv Regev, and Jure Leskovec. Biomni: A General-Purpose Biomedical AI Agent. *bioRxiv*, 2025. doi: 10.1101/2025.05.30.656746. URL https://doi.org/10.1101/2025.05.30.656746.

Yue Huang, Wenjie Wang, Han Bao, Yuchen Ma, Xiaonan Luo, Yi Nian, Haomin Zhuang, Zheyuan Liu, Yue Zhao, and Xiangliang Zhang. MemoHarness: Agent Harnesses That Learn from Experience, 2026. URL https://arxiv.org/abs/2607.14159.

Borui Kang, Jinrui Gu, Junhan Lv, Wenbin Li, Lei Wang, and Yang Gao. Harness continual learning: Continual adaptation beyond model parameters, 2026.

Omar Khattab, Arnav Singhvi, Paridhi Maheshwari, Zhiyuan Zhang, Keshav Santhanam, Sri Vardhamanan, Saiful Haq, Ashutosh Sharma, Thomas T. Joshi, Hanna Moazam, Heather Miller, Matei Zaharia, and Christopher Potts. DSPy: Compiling Declarative Language Model Calls into Self-Improving Pipelines. In *International Conference on Learning Representations*, 2024. URL https://arxiv.org/abs/2310.03714.

Yoonho Lee, Roshen Nair, Qizheng Zhang, Kangwook Lee, Omar Khattab, and Chelsea Finn. Meta-Harness: End-to-End Optimization of Model Harnesses, 2026. URL https://arxiv.org/abs/2603.28052.

Zelong Li, Shuyuan Xu, Kai Mei, Wenyue Hua, Balaji Rama, Om Raheja, Hao Wang, He Zhu, and Yongfeng Zhang. AutoFlow: Automated Workflow Generation for Large Language Model Agents, 2024. URL https://arxiv.org/abs/2407.12821.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. Self-Refine: Iterative Refinement with Self-Feedback, 2023. URL https://arxiv.org/abs/2303.17651.

Alexander Novikov, Ngan Vu, Marvin Eisenberger, Emilien Dupont, Po-Sen Huang, Adam Zsolt Wagner, Sergey Shirobokov, Borislav Kozlovskii, Francisco J. R. Ruiz, Abbas Mehrabian, M. Pawan Kumar, Abigail See, Swarat Chaudhuri, George Holland, Alex Davies, Sebastian Nowozin, Pushmeet Kohli, and Matej Balog. AlphaEvolve: A Coding Agent for Scientific and Algorithmic Discovery, 2025. URL https://arxiv.org/abs/2506.13131.

Haining Pan, James V. Roggeveen, Erez Berg, Juan Carrasquilla, Debanjan Chowdhury, Surya Ganguli, Federico Ghimenti, Juraj Hasik, Henry Hunt, Hong-Chen Jiang, Mason Kamb, Ying-Jer Kao, Ehsan Khatami, Michael J. Lawler, Di Luo, Titus Neupert, Xiaoliang Qi, Michael P. Brenner, and Eun-Ah Kim. CMT-Benchmark: A Benchmark for Condensed Matter Theory Built by Expert Researchers, 2025. URL https://arxiv.org/abs/2510.05228.

Mitchell A. Potter and Kenneth A. De Jong. A cooperative coevolutionary approach to function optimization. In *Parallel Problem Solving from Nature III*, pp. 249–257. Springer, 1994.

Yuanhao Qu, Yingzhou Lu, Xinming Tu, Serena Zhang, Tianwei She, Alexander Glenn Shaw, Jou-Ho Shih, Bingqing Zhao, Minjie Shen, Haochen Yang, Jielin Yan, Rongchuan Zhang, Xinze Wu, Tingting Li, Bin Zhou, Ning Wang, Adam Ma, Le Cong, Xiaobo Hu, Yuan Jiang, Jiayun Dong, Tao Peng, Jure Leskovec, and Kexin Huang. BiomniBench: Process-level Evaluation of LLM Agents for Real-world Biomedical Research. *bioRxiv*, 2026. doi: 10.64898/2026.05.12.724604. URL https://doi.org/10.64898/2026.05.12.724604.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R. Bowman. GPQA: A Graduate-Level Google-Proof Q&A Benchmark, 2023. URL https://arxiv.org/abs/2311.12022.

Zhe Ren, Yimeng Chen, Dandan Guo, Guowei Rong, Tonghui Li, R. B. Xiong, Qingfeng Lan, Wenyi Wang, Li Nanbo, Yibo Yang, Mingchen Zhuge, and Jürgen Schmidhuber. Self-Improvements in Modern Agentic Systems: A Survey, 2026. URL https://arxiv.org/abs/2607.13104.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language Models Can Teach Themselves to Use Tools. In *Advances in Neural Information Processing Systems*, 2023. URL https://arxiv.org/abs/2302.04761.

Noah Shinn, Federico Cassano, Edward Berman, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language Agents with Verbal Reinforcement Learning, 2023. URL https://arxiv.org/abs/2303.11366.

The Open Science Desktop Contributors. Open Science Desktop: a local-first, model-agnostic AI research workbench. Zenodo software release, version 0.5.2, 2026. URL https://github.com/ai4s-research/open-science.

Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An Open-Ended Embodied Agent with Large Language Models. *Transactions on Machine Learning Research*, 2024. URL https://arxiv.org/abs/2305.16291.

Lei Wang, Chen Ma, Xueyang Feng, Zeyu Zhang, Hao Yang, Jingsen Zhang, Zhiyuan Chen, Jiakai Tang, Xu Chen, Yankai Lin, Wayne Xin Zhao, Zhewei Wei, and Ji-Rong Wen. A Survey on Large Language Model based Autonomous Agents, 2023. URL https://arxiv.org/abs/2308.11432.

Xiang Wang, Zongtao Yang, Zhuojian Hong, Shuhao Zhang, and Wei Wei. FusionFlow: Enabling deep structural exploration for automated agentic workflow generation. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics*, pp. 27718–27760, 2026. doi: 10.18653/v1/2026.acl-long.1278.

Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Beibin Li, Erkang Zhu, Li Jiang, Xiaoyun Zhang, Shaokun Zhang, Jiale Liu, Ahmed Hassan Awadallah, Ryen W White, Doug Burger, and Chi Wang. AutoGen: Enabling Next-Gen LLM Applications via Multi-Agent Conversation. In *arXiv preprint arXiv:2308.08155*, 2023. URL https://arxiv.org/abs/2308.08155.

Zhougeng Xu, hoptop, Cursor Agent, Emberwhirl, Dinghow, Codex, Yu-Qiao-sjtu, Johnny Chen, jarxunlai, Chris Lou, Peng, wsx, Zac-lzh, and George Pickett. xuzhougeng/wisp-science: v1.5.0: Share. Zenodo software release, version 1.5.0, 2026. URL https://github.com/xuzhougeng/wisp-science.

John Yang, Carlos E. Jimenez, Alexander Wettig, Kilian Lieret, Shunyu Yao, Karthik Narasimhan, and Ofir Press. SWE-agent: Agent-Computer Interfaces Enable Automated Software Engineering, 2024. URL https://arxiv.org/abs/2405.15793.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. ReAct: Synergizing Reasoning and Acting in Language Models. In *International Conference on Learning Representations*, 2023. URL https://arxiv.org/abs/2210.03629.

Junwei Yu, Yepeng Ding, and Hiroyuki Sato. DynTaskMAS: A dynamic task graph-driven framework for asynchronous and parallel llm-based multi-agent systems, 2025. URL https://ojs.aaai.org/index.php/ICAPS/article/view/36130.

Zhaochen Yu, Yingcheng Wu, Zhenfei Yin, Kaiyuan Chen, Zhe Zhao, Mengdi Wang, Shuicheng Yan, and Ling Yang. Recursive experiential-working memory evolution for long-horizon agent harnesses, 2026.

Sukwon Yun, Jie Peng, Pingzhi Li, Wendong Fan, Jie Chen, James Zou, Guohao Li, and Tianlong Chen. Graph-of-Agents: A graph-based framework for multi-agent llm collaboration, 2026.

Eric Zelikman, Eliana Lorch, Lester Mackey, and Adam Tauman Kalai. Self-Taught Optimizer (STOP): Recursively Self-Improving Code Generation, 2023. URL https://arxiv.org/abs/2310.02304.

Guibin Zhang, Kaijie Chen, Guancheng Wan, Heng Chang, Hong Cheng, Kun Wang, Shuyue Hu, and Lei Bai. EvoFlow: Evolving Diverse Agentic Workflows On The Fly, 2025a. URL https://arxiv.org/abs/2502.07373.

Jenny Zhang, Shengran Hu, Cong Lu, Robert Lange, and Jeff Clune. Darwin Godel Machine: Open-Ended Evolution of Self-Improving Agents, 2025b. URL https://arxiv.org/abs/2505.22954.

Jenny Zhang, Bingchen Zhao, Wannan Yang, Jakob Foerster, Jeff Clune, Minqi Jiang, Sam Devlin, and Tatiana Shavrina. Hyperagents, 2026. URL https://arxiv.org/abs/2603.19461.

Jiayi Zhang, Jinyu Xiang, Zhaoyang Yu, Fengwei Teng, Xionghui Chen, Jiaqi Chen, Mingchen Zhuge, Xin Cheng, Sirui Hong, Jinlin Wang, Bingnan Zheng, Bang Liu, Yuyu Luo, and Chenglin Wu. AFlow: Automating Agentic Workflow Generation, 2024. URL https://arxiv.org/abs/2410.10762.

Andrew Zhao, Daniel Huang, Quentin Xu, Matthieu Lin, Yong-Jin Liu, and Gao Huang. ExpeL: LLM Agents Are Experiential Learners, 2023. URL https://arxiv.org/abs/2308.10144.

Mingchen Zhuge, Wenyi Wang, Louis Kirsch, Francesco Faccio, Dmitrii Khizbullin, and Jürgen Schmidhuber. Language Agents as Optimizable Graphs, 2024. URL https://arxiv.org/abs/2402.16823.

Appendix Contents

A Method and Runtime Details **13**
A.1 Notation and Composition Procedure . . . 13
A.2 Saved Harnesses and Concrete Compositions . . . 16
A.3 Planner Inputs, Session Prompts, and Runtime Organization . . . 20

B Experimental Protocol and State Records **21**
B.1 Experimental Setup and Completed Mechanism Ablations . . . 21
B.2 Round-Resolved Cooperation Structure . . . 24
B.3 Archived Case Evidence and Provenance . . . 25

C Theoretical Guarantees **25**
C.1 The Method Objects Used by the Theory . . . 25
C.2 Constructive Reduction to Cooperative Coevolution . . . 25
C.3 Contextual Marginals and Recombination Gains . . . 26
C.4 Bank-mediated State Update . . . 26
C.5 Cooperation Memory and Matched Interaction Measurements . . . 26

D Case Studies and Evidence **26**
D.1 Comparative Case Studies: From Harness to Answer . . . 26
D.2 Experience Bank Case Study . . . 31
D.3 Rubric Profiles and Scientific Execution . . . 31
D.4 Workflow Shapes . . . 38
D.5 Compact Held-Out Agent Walkthroughs . . . 39

E Related Work and Positioning **39**
E.1 A common language for the progression . . . 42
E.2 Feedback improves the current solution . . . 42
E.3 Experience becomes a persistent skill or memory . . . 42
E.4 Programs and workflows become optimization targets . . . 42
E.5 The update operator enters the improvement space . . . 42
E.6 Composition makes component value contextual . . . 42

F Discussion **42**

G AI Assistance **43**

A Method and Runtime Details

A.1 Notation and Composition Procedure

The main method connects typed edits to complete-harness reward and retained state. This appendix makes that connection executable: we align the notation, give the round-level procedure, then specify ordered compatibility and the binding of outcomes to the next state. Table 2 organizes the objects used by Algorithms 1 and 2.
For a task x, let H^{prev} denote the recorded predecessor configuration and H^{eval} the configuration evaluated by the benchmark evaluator. The typed episode delta and retained round state are

$$\Delta = H^{\text{eval}} \ominus H^{\text{prev}}, \tag{6}$$

$$\Sigma_t = (\mathcal{B}_t, \mathcal{M}_t, C_t), \tag{7}$$

where $\mathcal{B}_t$ is the experience bank, $\mathcal{M}_t$ the typed module catalog, and C_t the cooperation memory. For a completed episode q with module set S_q, reward r_q, and success threshold τ, the co-success update is

$$C_{t+1}(u,v) = C_t(u,v) + \mathbf{1}[(u,v) \subseteq S_q]\,\mathbf{1}[r_q \geq \tau]\,, \tag{8}$$

with each unordered pair counted once per successful episode.
For task-conditioned recall, the Planner and decoder are written as

$$P_t = \text{Planner}(Z_t(x;\Sigma_t)), \qquad H_t^\star = \text{Decode}_{H_0}(P_t), \tag{9}$$

and a participant j with recalled module set S_j is materialized by

$$H_j = H_0 \oplus S_j. \tag{10}$$

The declared operation program routes named artifacts through its dependency graph and produces the execution record

$$\mathcal{E}_t(x) = \text{Exec}(P_t, H_t^\star, x; A_s), \tag{11}$$

where A_s is the named artifact store.

Table 2: Notation shared by the method, algorithms, and theoretical reduction.

Symbol	Meaning
x, t, b	Task, training round, execution budget
H_0	Complete base configuration for the episode
m, p_m, d	Module occurrence, its typed field patch, dimension
$\mathcal{B}_t, \mathcal{M}_t, C_t$	Experience bank, module catalog, cooperation memory
Σ_t	The three-part retained state
$S_{j,d}, S_j, \sigma_j$	Dimension-specific selections, full module selection, declared application order
ℓ_j, H_j, k	Participant label, complete participant configuration, participant count
$P_t, G_t = (\mathcal{V}_t, E_t)$	Construction program and its dependency graph
$\rho, I(v), O(v)$	Participant assignment, named input handles, named output handles
a, A_s, V_t	Execution adapter, artifact store, compatibility and operation record
$H^c, H_t^\star, \mathcal{E}_t(x)$	Committed configuration, complete executable harness, execution record
$\widehat{y}, r, R, \tau$	Final answer, reward, task evaluator, success threshold
$\mathcal{Q}_t, L_q$	Completed round records and retained modules in record q
$H_q^{\text{eval}}, H_q^{\text{prev}}$	Evaluated configuration and recorded predecessor for delta extraction
K_t, k_t, J_t, λ_t	Recall-pool size, selected-module count, participant count, recall-strength vector
$\phi_t, \mathcal{R}_t, \text{Prec}_t$	Recall score, task-conditioned module pool, recall precision
$\theta, V_\theta, \delta_i, \kappa_{ij}$	Fixed decoding/evaluation context, expected value, module marginal, pair interaction

A.1.1 Task-conditioned recall: mode, precision, and strength

The recall unit is a typed module occurrence. For a task x, base harness H_0, and retained state Σ_t, the Planner assigns every candidate module m a contextual score

$$\phi_t(m \mid x, H_0) = \alpha s_{\text{task}}(m, x) + \beta s_{\text{base}}(m, H_0) + \gamma s_{\text{coop}}(m; \Sigma_t) + \eta s_{\text{compat}}(m, x, H_0), \tag{12}$$

where the four terms measure task fit, base-harness fit, cooperation evidence, and executable compatibility. The bounded candidate pool is

$$\mathcal{R}_t(x; K_t) = \text{TopK}_{m \in \mathcal{M}_t \cup \mathcal{B}_t} \phi_t(m \mid x, H_0), \tag{13}$$

with ties resolved by the recorded module order. Thus CoRSI recalls typed field patches from the catalog and experience bank; the reusable unit is a module rather than a complete episode.

The recalled pool is compiled into a workflow program

$$P_t = \big(\mathcal{R}_t, \{S_{j,t}\}_{j=1}^{J_t}, \rho_t, G_t, \sigma_t, \omega_t\big), \qquad G_t = (\mathcal{V}_t, E_t), \tag{14}$$

where $S_{j,t}$ is the module set assigned to participant j, ρ_t is the assignment map, G_t is the named-artifact dependency graph, σ_t is the module application order, and ω_t is the operation order. The Planner therefore determines participant count, module assignment, roles, artifact routes, and workflow order in one typed program.

Recall precision has two operational levels. The compatibility precision of a round is the fraction of recalled modules admitted by guarded assembly,

$$\text{Prec}_t^{\text{comp}} = \frac{|S_t^{\text{acc}}|}{|\mathcal{R}_t|}, \tag{15}$$

and useful-recall precision binds admitted modules to a completed executable workflow,

$$\text{Prec}_t^{\text{use}} = \frac{|\{m \in S_t^{\text{acc}} : m \text{ contributes to the completed workflow}\}|}{|S_t^{\text{acc}}|}. \tag{16}$$

The first is determined at assembly; the second is determined from the execution receipt and evaluator outcome.

Recall strength is a vector that controls search and execution scope,

$$\lambda_t = (K_t, k_t, J_t, |E_t|, b_t), \tag{17}$$

where K_t is the candidate-pool size, k_t the number of selected modules, J_t the number of participants, $|E_t|$ the number of artifact dependencies, and b_t the task-level workflow budget. In particular, $k_t \le K_t$ and $\mathcal{R}_t(x; \lambda_t) = \text{TopK}_{K_t} \phi_t(m \mid x, H_0)$. Recall precision determines which candidates survive compatibility and workflow checks; recall strength determines how many candidates and execution resources are exposed to the Planner. These are separate controls in the program and in the receipt.

A.1.2 EXPLICIT RSI STATE RECURSION

Let $\mathcal{Q}_t = \{q_{t,1}, \ldots, q_{t,n_t}\}$ be the completed episode records for round t. For each record, the retained typed change is

$$L_q = H_q^{\text{eval}} \ominus H_q^{\text{prev}}. \tag{18}$$

The round transition is the following explicit recursion:

$$\mathcal{B}_{t+1} = \mathcal{B}_t \cup \{q : q \in \mathcal{Q}_t\}, \tag{19}$$

$$\mathcal{M}_{t+1} = \mathcal{M}_t \cup \text{Extract}\big(\{L_q : q \in \mathcal{Q}_t\}\big), \tag{20}$$

$$C_{t+1} = C_t + \text{CoSuccess}(\mathcal{Q}_t, \tau), \tag{21}$$

$$\Sigma_{t+1} = (\mathcal{B}_{t+1}, \mathcal{M}_{t+1}, C_{t+1}). \tag{22}$$

Here Extract splits each evaluated delta into typed field patches, while CoSuccess forms unordered module pairs within each successful record and increments each pair once. Algorithms 1 and 2 implement these transitions: the complete harness is evaluated first, and only then are the resulting records available to the next round's recall pool.

Table 2 separates three objects that appear together in an execution record. H_j is a participant's full configuration; P_t constructs the workflow; $H_t^\star$ is the resulting complete executable harness. The artifact store A_s records intermediate products, whereas Σ_t retains the bank, catalog, and cooperation history between rounds. This distinction determines what an update changes and which complete solution receives reward.

Algorithm 1 Composition, evaluation, and retained-state transition

Require: State Σ_t, episode bases H_0, tasks, budget b, adapter a, training or evaluation mode
1: Initialize completed records $\mathcal{Q}_t \leftarrow \varnothing$
2: **for** each task x **do**
3: Score typed candidates with Eq. (12) and form $\mathcal{R}_t(x; K_t)$ by Eq. (13)
4: Construct P_t with participant sets, assignments, graph, and operation orders by Eq. (14)
5: Validate references, operation types, dependencies, and budget
6: Preflight participant selections; materialize each $H_j = H_0 \oplus_{\sigma_j} S_j$
7: Decode P_t into $H_t^\star$ using the declared adapter
8: Execute named-artifact routing by Eq. (11); apply Algorithm 2 at configuration merge steps
9: Obtain $\widehat{y}$ and evaluator reward r; record $\mathcal{E}_t(x)$
10: Append the completed episode and its configuration provenance to $\mathcal{Q}_t$
11: **end for**
12: **if** training mode **then**
13: Append experience and extract field deltas by Eq. (6)
14: Update cooperation counts with the adapter's identities and threshold by Eq. (8)
15: Let the Controller propose typed field changes; accept the candidate under the evaluator or canary gate
16: Extract the accepted typed field delta and update the bank and module catalog
17: **else**
18: Keep Σ_t fixed
19: **end if**
20: **return** Completed episode records and the resulting state

The round record separates completed evaluations from proposed changes. The inference transition keeps the training state fixed. The training transition updates the bank, module catalog, and cooperation counts from completed episodes; the next Planner composition reads the resulting state.

A.1.3 ORDERED COMPATIBILITY AND CONFIGURATION TRANSITIONS

A configuration merge receives the ordered module occurrences referenced by the team, flattened first by participant order and then by each participant's module order. For the field-write patches used by the cases, let W_m map fully qualified leaf paths to their proposed values. Start with an empty accepted list S^{acc}, an empty prior-write map, and a clone of H_0. Algorithm 2 gives the deterministic admission rule. Shared equal-valued writes are compatible; conflicting values reject the incoming module as a whole. Prerequisites must already be accepted when the module is visited. The rule is ordered, so the receipt retains the order as well as the accepted identities.

Participant preflight uses this compatibility test and rejects an internally conflicting group before invocation. At the cross-participant boundary, check-compatibility stores accepted module IDs and rejection reasons. Partial merge reads that named artifact, filters the original catalog occurrences by its accepted IDs, preserves their order, and re-applies them to the base. The candidates for this merge are configuration patches from the catalog;

Algorithm 2 Guarded assembly of an ordered field-patch selection

Require: Base H_0, module selection in declared order σ

Initialize candidate configuration $H \leftarrow H_0$, accepted list $S^{\mathrm{acc}} \leftarrow []$, write map $W \leftarrow \varnothing$, conflict log $V \leftarrow []$
for each module m in order σ **do**
  if m or an accepted module declares a mutual conflict **then**
    Record the declared conflict in V; continue
  else if a prerequisite of m is absent from S^{acc} **then**
    Record the missing prerequisite in V; continue
  else if some $f \in \mathrm{dom}(W_m) \cap \mathrm{dom}(W)$ has $W_m[f] \neq W[f]$ **then**
    Record prior writer, incoming module, and field f in V; continue
  else
    Deep-apply p_m to H; append m to S^{acc}; update W with W_m
  end if
end for
return (H, S^{acc}, V)

answer artifacts are inputs to answer synthesis. This explains why a participant's proposed answer and its selected field patch occupy different parts of the receipt.

The remaining configuration operations have explicit return types. For a configuration H and task x, the recorded verifier wrapper implements

$$\mathrm{Verify}_x(H) = \begin{cases} H, & \text{callback accepts, or no callback is installed,} \\ \text{failure}, & \text{callback returns false or raises.} \end{cases}$$

It calls the verifier on a copy and returns the original configuration after acceptance. Stabilize returns the callback's configuration, or an identity copy when no callback is installed; a nonconfiguration result is an error. Commit copies its named configuration input to H^c and records the commit event. A verification predicate therefore applies to the configuration at that verify step. When a stabilizer changes fields, a later verify step can test the changed configuration. The adapter's declared callback bindings and ordered operation record specify this choice. Each operation records failure and follows the program's skip-or-raise instruction. A skip creates no successful output artifact; a later step that requires that artifact encounters the missing-input check.

A.1.4 Binding a Record to the Next State

The experience update retains the actual evaluated configuration and its recorded predecessor. In the audited training loop, the predecessor is the stored configuration from the latest earlier round, with a minimal configuration used for initialization. The saved difference is split into field or dimension patches. Each occurrence carries the source task and round, reward, and success evidence. Occurrence retention is keyed by module ID; content identity supports grouping the same field behavior across sources. This permits the paper's distinct counts of experience records and unique patch contents.

The cooperation update deduplicates the retained modules, forms unordered pairs, and increments each once when reward meets the success threshold. Receipts retain the participant grouping and the module bindings for every pair. The typed-module identity is used by the round-resolved matrices in Appendix B.2; role labels are display annotations.

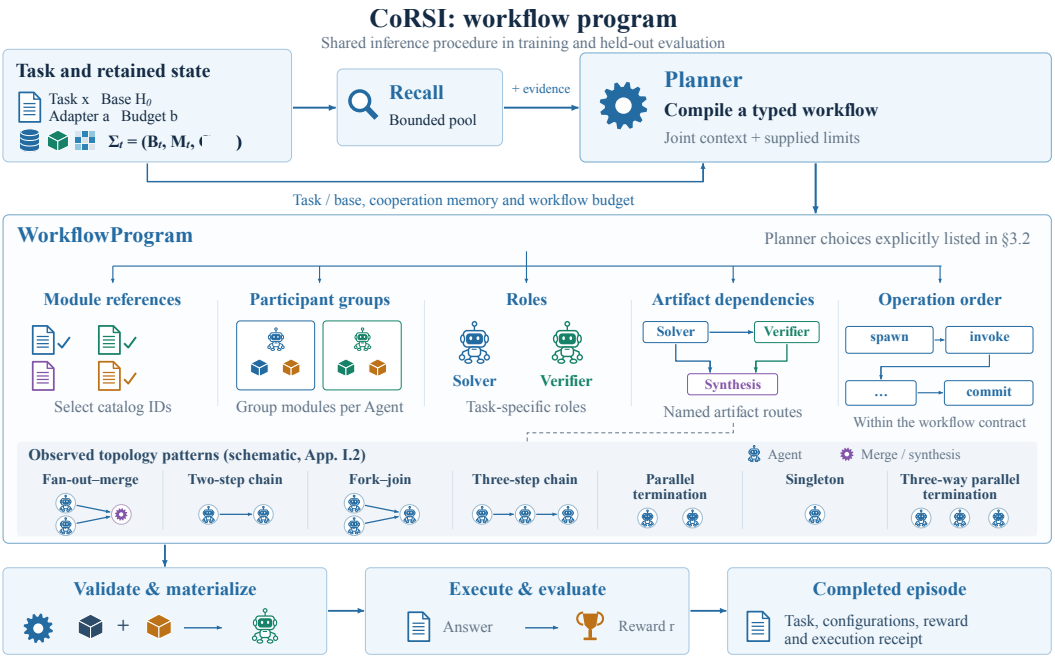
The resulting state records the bank, catalog revision, cooperation counts, retained modules, workflow description and resource accounting used by the next Planner composition.

A.2 Saved Harnesses and Concrete Compositions

The method and theory define what can be recombined; this appendix shows the actual configuration contents. Figure 9 fixes the batch boundary, the training-only state updates, and the frozen held-out branch. The records below then trace instruction changes across training and compare disjoint-field and same-field merges. They supply the concrete configuration evidence behind the workflow program in Figure 8 and the paired case studies in Appendix D.1.

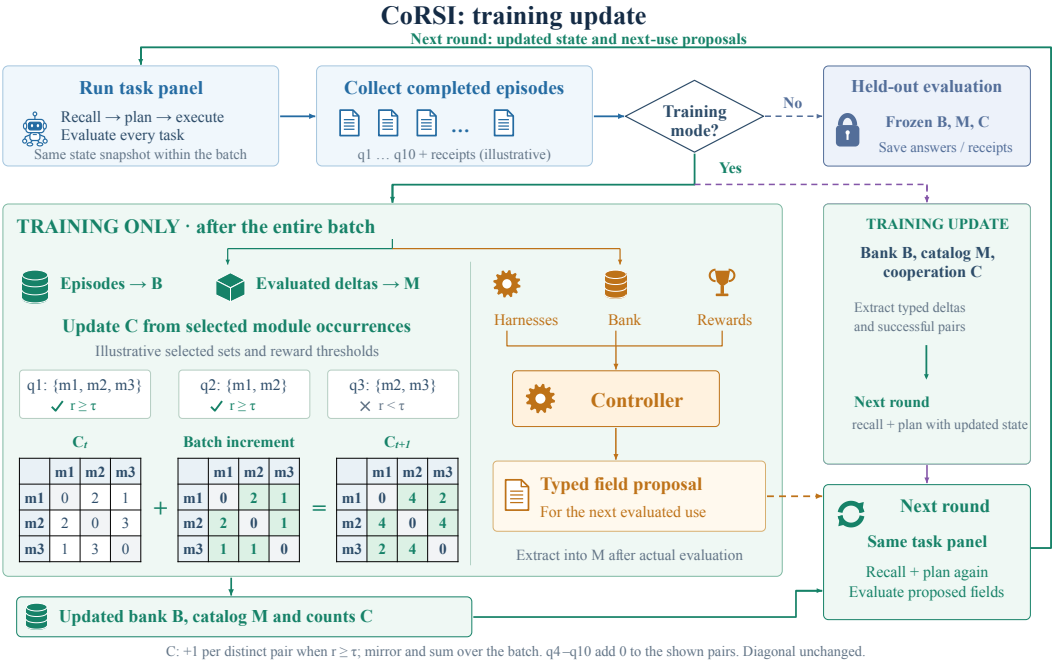
Figure 8 is the appendix-level specification of the per-task workflow. It makes five Planner decisions explicit: which typed modules are recalled, how they are grouped into participants, which roles they serve, which named artifacts they exchange, and in what order operations run. The lower stages apply the compatible patches with base-field inheritance, execute the resulting workflow, and record the answer, reward, and execution receipt. The topology strip summarizes the dependency patterns observed in the experiments; the case studies below instantiate these choices with concrete module contents and artifact traces.

Figure 9 separates the batch-completion boundary from the two runtime modes. Training updates the bank, evaluated module catalog, and cooperation counts only after the task panel is complete, while held-out evaluation keeps the retained state frozen. The next round consumes the updated bank, module catalog, cooperation



§§3.2–3.4; App. I.2 / Fig. 10. Topologies are observed dependency schematics; common finalization stages are omitted.

Figure 8: CoRSI workflow program with typed module recall and artifact routing.



C: +1 per distinct pair when r ≥ τ; mirror and sum over the batch. q4–q10 add 0 to the shown pairs. Diagonal unchanged.
§§3.1, 3.6 / Eq. (7), Alg. 1–2; App. D.1. Controller inputs follow author clarification; proposal -generation rule unspecified.

Figure 9: CoRSI training update and retained state.

state, and task-conditioned proposals. The example cooperation matrix is illustrative arithmetic for the successful pair updates.

A.2.1 THREE STAGES OF THE EVALUATED HARNESS

The records below are the configurations attached to evaluated bank entries at rounds 1, 5, and 10. A controller checkpoint saved after a round can instead describe the next proposed configuration. Each configuration is paired with the round in which it was evaluated; post-round proposals are indexed to their next use.

Tools remains tool-free: no terminal tools, no retrieval, and a zero retrieval count. Memory keeps the complete task in a single-prompt memory with history length one. Generation uses temperature zero, top-p one, and one candidate. Context uses no examples, no compression, and no structured-instruction flag. Orchestration retains direct QA and a formatted-final-answer stop rule. Verification uses raw output, a requested-format validator, and one empty-output retry. The evolving instructions and token limit are:

The evaluated round-1 harness uses a Generation token limit of 4096 and the following instructions.

> **Context:** reason about the scientific question and obey its exact answer format
>
> **Orchestration:** one direct reasoning call with one empty-output retry
>
> **Verification:** return only the answer format requested by the task

The evaluated round-5 harness uses a Generation token limit of 3072 and the following instructions.

> **Context:** solve by decomposing the scientific prompt into givens, target quantity, governing relation, units, and edge conditions; compute explicitly and run a silent dimensional, numerical, and formatting check before answering exactly in the requested format
>
> **Orchestration:** one direct reasoning call that includes an internal derive-then-verify pass before finalization; stop only after the requested answer is complete, numerically consistent, unit-consistent, and format-compliant; retry only if the first output is empty
>
> **Verification:** return only the requested final answer with required labels, ordering, precision, signs, units, significant figures, and delimiters; silently correct arithmetic, unit, or format mismatches before emission; no explanation unless explicitly requested

The evaluated round-10 harness uses a Generation token limit of 4096 and the following instructions.

> **Context:** Classify the task first as calculation, conceptual, extraction, comparison, or format-only. Extract the exact output contract, givens, unknowns, qualifiers, units, signs, and negations. Apply only the relevant domain method from the prompt using the simplest valid derivation, with a brief task-appropriate sanity check. Do not force science-specific checks on general tasks. Preserve the requested format and omit unrequested reasoning.
>
> **Orchestration:** Make exactly one direct QA call. Silently solve the task, check the explicit qualifiers once, and emit the complete requested answer as the response itself. Do not emit drafts, pass labels, analysis, wrappers, or termination markers. Treat any complete nonempty response as final; use the single retry only when the first response is empty, replaying the full task prompt.
>
> **Verification:** Check the complete nonempty response only against the user's explicit output contract. Preserve content, order, units, signs, precision, and requested explanation; minimally correct formatting when needed. Never add labels or wrappers, truncate, or reject a complete answer because it lacks an internal marker. Return only the requested answer.

A.2.2 THE EXECUTED THREE-HARNESS TEAM

The three-participant case materializes three participants from the following selected field records. The records all have source a source experience at an early training round. Every field absent from a participant's retained modules inherits the round-2 base configuration. The inherited Context, Tools, Generation, Memory, and Verification settings match the round-1 values listed above; the round-2 base Orchestration strategy adds an internal scientific and formatting check. The selected records and inherited base fields specify the complete participant harnesses.

Harness A combines Context and Tools through the following field patches.

Context strategy = reason about the scientific question and obey its exact answer format.

Tools tool_access = none.

Tools tool_protocol = none.

Tools strategy = answer from the prompt without terminal tools.

Harness B combines Generation and Orchestration through the following field patches.

Generation max_tokens = 4096.

Orchestration workflow = direct_qa.

Orchestration stop_rule = formatted_final_answer.

Orchestration strategy = one direct reasoning call with one empty-output retry.

Harness C combines Memory and Verification through the following field patches.

Memory memory_policy = single_prompt.

Memory history_keep = 1.

Memory strategy = retain the complete task prompt.

Verification validator = requested_output_format.

Verification fallback = single_empty_output_retry.

Verification strategy = return only the answer format requested by the task.

The declared planner operations are recall, map invocation, compatibility check, partial merge, verify, stabilize, and commit. The saved answer-producing runtime provides three participant artifacts and final synthesis. This execution projection supplies the outputs shown in Figure 8. All three answers choose (c). The five recorded model calls include overhead beyond the three participants.

A.2.3 A COMMITTED CROSS-SOURCE HARNESS

CMT-002 commits both modules below. Their Context and Orchestration strategy fields are disjoint; all other fields inherit the base configuration.

The Context strategy is drawn from a source experience at a later training round:

> strategy = Classify the task first as calculation, conceptual, extraction, comparison, or format-only. Extract the exact output contract, givens, unknowns, qualifiers, units, signs, and negations. Apply only the relevant domain method from the prompt using the simplest valid derivation, with a brief task-appropriate sanity check. Do not force science-specific checks on general tasks. Preserve the requested format and omit unrequested reasoning.

The Orchestration strategy is drawn from CMT-004 at an earlier training round:

> strategy = Use one direct QA call, but require two private passes inside it before emitting. Pass 1 produces a candidate answer from the most direct scientific model. Pass 2 acts as an adversarial verifier: reread the exact question, look for hidden qualifiers, swapped variables, unit conversions, sign conventions, boundary conditions, and tempting but inapplicable formulas; recompute the decisive step with an independent formulation or sanity check. Emit only if both passes agree and the answer satisfies every explicit condition and the requested schema. If they disagree, discard the first candidate and resolve the conflict from prompt givens within the same call. Retry only when the first output is empty.

A.2.4 THREE CANDIDATE HARNESSES AT A COLLISION BOUNDARY

CMT-007 selects the following three pairs in its held-out adaptation receipt. For each candidate, Tools, Generation, Memory, and Verification inherit the same base harness. Harness A's two modules are accepted. Harness B's and Harness C's changes collide with its Context and Orchestration strategy writes, yielding four recorded conflicts. The configuration executor returns one committed harness.

The accepted Harness A contains the following two strategies.

The Context field comes from a source experience, a later training round:

> strategy = Classify the task first as calculation, conceptual, extraction, comparison, or format-only. Extract the exact output contract, givens, unknowns, qualifiers, units, signs, and negations. Apply only the relevant domain method from the prompt using the simplest valid derivation, with a brief task-appropriate sanity check. Do not force science-specific checks on general tasks. Preserve the requested format and omit unrequested reasoning.

The Orchestration field comes from a source experience, a later training round:

> strategy = Make exactly one direct QA call. Silently solve the task, check the explicit qualifiers once, and emit the complete requested answer as the response itself. Do not emit drafts, pass labels, analysis, wrappers, or termination markers. Treat any complete nonempty response as final; use the single retry only when the first response is empty, replaying the full task prompt.

Harness B proposes the following conflicting field writes.

The Context field comes from CMT-004, an earlier training round:

> strategy = Privately parse the prompt as a precision-critical science exam item before solving. Identify the requested answer type, output contract, givens, unknowns, units, signs, reference states, approximations, and any negations or qualifiers. Choose the governing relation only after matching the task type and constraints, then solve symbolically or numerically from prompt data alone. Before finalizing, compare against at least one independent framing: conservation/balance vs formula, limiting case, dimensional check, qualitative direction, or answer-choice elimination as applicable. Preserve the user's requested format exactly and do not expose reasoning unless requested.

The Orchestration field comes from CMT-004, an earlier training round:

> strategy = Use one direct QA call, but require two private passes inside it before emitting. Pass 1 produces a candidate answer from the most direct scientific model. Pass 2 acts as an adversarial verifier: reread the exact question, look for hidden qualifiers, swapped variables, unit conversions, sign conventions, boundary conditions, and tempting but inapplicable formulas; recompute the decisive step with an independent formulation or sanity check. Emit only if both passes agree and the answer satisfies every explicit condition and the requested schema. If they disagree, discard the first candidate and resolve the conflict from prompt givens within the same call. Retry only when the first output is empty.

Harness C proposes the following conflicting field writes.
The Context field comes from a source experience, round 8:

> strategy = First restate privately the exact answer contract and classify the task type: numeric, symbolic, multiple choice, conceptual, comparison, or extraction. Build a minimal variable map from only prompt givens, including units, signs, reference states, constraints, and requested precision. Solve with the governing scientific relation most directly implied by the prompt, then verify by substitution, dimensional analysis, limiting case, and consistency with every stated condition before selecting the final answer. Preserve the user's requested format exactly and do not expose reasoning unless requested.

The Orchestration field comes from a source experience, an early training round:

> workflow = direct_qa; stop_rule = formatted_final_answer; strategy = one direct reasoning call with one empty-output retry

The full recorded operation sequence is recall, spawn, map invocation, review, compatibility check, conflict test, partial merge, verify, stabilize, and commit. The invocation summaries describe local configurations and module references. This receipt documents the configuration merge: accepted fields, rejected collisions, and the final committed harness.

A.3 Planner Inputs, Session Prompts, and Runtime Organization

The previous appendix gives the retained policies; this appendix explains how those policies reach execution. We specify the planner's visible inputs, the native CMT-Bench participant request and synthesis interface, and a sequential tool-enabled adapter. Reading the request format alongside the saved outputs distinguishes declared operations, executed calls, and persistent artifacts.

A.3.1 What the planner sees

The audited planner input contains a bounded task summary and observable features; base-configuration dimension names, field names, and value types; module entries, dimensions, roles, field assignments, source cases and rounds, outcome summaries, requirements and conflicts; a co-success matrix; and the operation schema and budget. The participant receives its fully materialized configuration after the planner summary.
The system instruction defines the planner as a compiler of a bounded, synchronous program, requires the planner to use catalogued module entries, and requires one terminal output. The key operational rules, condensed from the inspected template, are:

1. Group catalog modules into participants; a participant can override multiple harness dimensions and inherits omitted fields.
2. Prefer complementary fields and evidence, but never interpret co-success as causal synergy or invent absent counts.
3. Use named artifacts and explicit dependencies; a planner rationale is explanatory metadata.
4. Copy the supplied budget exactly; use canonical step fields and only allow-listed operations.
5. End a committing program with compatibility checking, partial merge, verification, stabilization, and one final commit.

The CMT-004 receipt specifies ceilings of three participants, three participant invocations, sixteen program steps, one planner call, and 4,096 planner-budget tokens. The planner budget is a schema field; the end-to-end receipt additionally records participant and synthesis usage.

A.3.2 Native participant session

The following is the executable prompt wrapper, with dynamic values represented by angle-bracket fields. The template is paired with the corresponding execution records.

> You are one member of a Planner-selected reasoning team. Solve the task independently, then use any upstream artifacts as evidence. Return a concise candidate answer in the exact format requested by the task.
>
> Agent role: ⟨participant role⟩

Harness dimensions: ⟨module entries by dimension⟩

Composed Harness config: ⟨complete inherited configuration⟩

Upstream artifacts: ⟨bounded artifact mapping⟩

The task itself is the user message. In the native fan-out implementation, participants are launched before their answers are inserted into the shared artifact map. The saved CMT-004 artifacts therefore correspond to independent answers followed by synthesis. The final synthesizer is instructed:

> You are the final synthesizer. Resolve disagreements among the candidate answers and return only the answer format requested by the original task.

The bounded answer-artifact mapping follows this instruction.

A.3.3 Tool-enabled scientific session

The Biomni adapter uses a different, sequential organization. It sends the original task, the role duty, retained modules, the complete six-dimensional configuration, and summaries of earlier participants. The first role is a scoped specialist; the last is the final integrator. The following excerpts preserve the operational wording, with machine locations replaced by descriptive environment names:

> You are one tool-enabled participant in a Planner-selected `CoRSI` workflow.
>
> Work as the scoped specialist. Analyze the task with tools and data, preserve useful artifacts in the task workspace, and end with a concise evidence summary for the next participant.
>
> For the final integrator: inspect prior participant outputs and all files in the task workspace. Complete the requested analysis, verify quantitative results, and create the required analysis trace and final answer.
>
> Selected harness modules: ⟨selected entries⟩
>
> Composed six-dimension Harness config: ⟨participant configuration⟩
>
> Prior participant summaries: ⟨bounded earlier summaries⟩

The inspected wrapper additionally restricts task data and prohibits access to source papers, hidden references, rubrics, evaluator code, plugins, skills, and sub-agents. The wrapper records these access restrictions in each participant prompt. The adapter runs participants in list order, divides remaining wall-clock time among them, and supplies earlier summaries to later calls. It therefore provides an explicit handoff absent from the independent CMT-Bench fan-out.

For DA-13-1, Context asks the context agent to inspect acceptance requirements before completion. The tool agent's Tools strategy asks for workspace inspection and lightweight local probes. Both inherit a 2,048-token generation setting, an agentic loop with a final completion audit, sixteen recent memory exchanges, and workspace output checks. The committed configuration contains both Context and Tools changes. The second saved summary states that the analysis was rerun from the two provided inputs and the tables regenerated consistently. The saved outputs expose artifact production and verification within a composed tool-enabled workflow. In the saved round-2 DA-13-1 trace, the selected Context and Tools modules come from DA-14-1 at round 1. The two participant processes finish in 646.714 and 452.662 seconds; both summaries report 541 significant analytes among 5,414 and nine regimen interactions. The retained analysis files and summaries connect the first participant's work to the integrator's rerun. This adapter example illustrates persistent artifact handoff (Huang et al., 2025). Appendix D.3 extends this training example to the frozen-bank benchmark and criterion-level held-out cases.

B Experimental Protocol and State Records

B.1 Experimental Setup and Completed Mechanism Ablations

The method defines editable fields and a reward-bearing workflow. This appendix records the completed benchmark settings and the executed GPQA-Diamond ablations. Five GPQA-Diamond training rounds and ten CMT-Bench training rounds are sequential bank updates within one training run.

B.1.1 Archived results and protocol

GPQA-Diamond uses 98 adaptation tasks over five rounds and 100 held-out tasks: Organic-ID 50, Physics Subdomain-OOD 31, and Biology Domain-OOD 19. Claude Haiku 4.5, Claude Sonnet 5, and Claude Opus 5 use medium effort and separate banks. `CoRSI` correct counts are 71, 88, 89; MemoHarness counts are 65, 77, 78; Claude Code counts are 60, 73, 78. The task-level scores are retained with their task records, model settings, and evaluator version. CMT-Bench uses ten held-out tasks, five IID and five OOD. The archived CMT-Bench task-level records use the same ten held-out tasks and evaluator setting for every condition. The controlled measurements below focus on CMT-Bench and GPQA-Diamond.

B.1.2 Datasets and benchmark descriptions

The benchmark suite is chosen to separate transfer across problem families, scientific domains, and tool-mediated analysis. Each dataset keeps its native answer format and evaluator; the bank is trained on an adaptation panel and then evaluated on a fixed held-out panel. This organization follows the benchmark-specific task semantics rather than converting all outcomes into a single binary score.

- **GPQA-Diamond.** GPQA-Diamond is a graduate-level multiple-choice science benchmark covering biology, chemistry, and physics (Rein et al., 2023). We use 98 adaptation questions over five rounds and 100 held-out questions, partitioned into 50 Organic-ID, 31 Physics Subdomain-OOD, and 19 Biology Domain-OOD questions. The evaluator uses exact answer matching.
- **AIME 2024, AIME 2025, BeyondAIME, HMMT 2025, and HMMT 2026.** These are competition-mathematics collections with integer-valued answers. AIME 2024 supplies the five-round adaptation panel; AIME 2025, HMMT 2025, and HMMT 2026 provide held-out transfer panels of 30, 30, and 33 tasks, respectively. BeyondAIME is included as an additional exact-integer transfer collection. All mathematics endpoints use exact-match scoring and retain the competition's answer range. The source pages used for the collections are `https://maa.org/student-programs/amc/`, `https://www.hmmt.org/`, and `https://matharena.ai/`.
- **CMT-Bench.** CMT-Bench contains expert-authored condensed-matter theory problems spanning multiple calculation families (Pan et al., 2025). The `CoRSI` campaign runs ten training rounds with 40 tasks per round and reports five IID and five OOD held-out tasks. The evaluator checks the final answer against the benchmark reference and preserves the category labels used for transfer analysis. The archived task collection is available at `https://github.com/JamesRoggeveen/cmt_benchmark_data`.
- **BiomniBench-DA.** BiomniBench-DA is a 50-task biomedical data-analysis subset in which each task supplies data, a research question, and an expert-authored rubric (Qu et al., 2026). Thirty tasks form the five-round adaptation panel and 20 tasks form the frozen held-out panel, including 14 IID and six OOD tasks. Gemini 3.1 Pro assigns criterion-level A/B/C judgments to the recorded scientific execution and aggregates them into a 0–100 task score; the complete rubric construction is given in Appendix D.3.1.

B.1.3 IID AND OOD PARTITION RULES

The partition labels describe what changes between the adaptation panel and the held-out panel. Domain IID keeps the broad scientific or task domain fixed and holds out new items. Domain OOD changes that broad domain. Sub-domain IID keeps both the broad domain and the adaptation sub-domain or problem family fixed, while Sub-domain OOD keeps the broad domain but moves to a held-out sub-domain or family. The labels are applied to the native structure of each benchmark.

- **GPQA-Diamond.** The 98-question adaptation panel supplies the chemistry, physics, and biology experience. Organic-ID is Sub-domain IID: the held-out questions are new organic-chemistry items within the adaptation-covered sub-domain. Physics Subdomain-OOD is Sub-domain OOD: the broad physics domain is retained while the held-out physics sub-domains are not represented in adaptation. Biology Domain-OOD is Domain OOD: biology is held out as a scientific domain from the adaptation panel.
- **Competition mathematics.** AIME 2025 is an edition-transfer panel within the same competition family as the AIME 2024 adaptation panel. HMMT 2025 and HMMT 2026 are family-transfer panels with a different competition structure. We report these as competition-family transfer rather than forcing them into the scientific Domain IID/OOD labels.
- **CMT-Bench.** The five IID tasks preserve categories represented in the training panel while holding out new problems. The five OOD tasks use novel categories. Thus the split is category IID versus category OOD, with the same condensed-matter domain and evaluator.
- **BiomniBench-DA.** The 14 IID tasks preserve the biomedical study categories represented in the 30-task adaptation panel. The six OOD tasks change the study category or analysis setting while retaining the biomedical data-analysis domain. Their rubric scores are aggregated with the same criterion-level evaluator.

B.1.4 BENCHMARK DOMAINS, TASK FAMILIES, AND PANEL CONSTRUCTION

The benchmark suite uses two linked partitions. The adaptation panel supplies the experiences used to construct the bank; the held-out panel supplies new tasks for transfer evaluation. Within the held-out panel, the native domain or task-family labels identify whether a task is an in-domain item, a new sub-domain, or a new domain. The complete task inventory is as follows.

- **GPQA-Diamond.** The benchmark covers three scientific domains: chemistry, physics, and biology. The adaptation panel contains 98 questions used over five rounds and provides experience in chemistry and physics. The held-out panel contains 100 questions: 50 Organic-ID questions from the organic-chemistry sub-domain (Sub-domain IID), 31 physics questions from held-out physics sub-domains (Sub-domain OOD), and 19 biology questions from a held-out scientific domain (Domain OOD). All questions use exact multiple-choice matching.
- **Competition mathematics.** The task families are the AIME family and the HMMT family. AIME 2024 is the adaptation collection used for five rounds. The held-out tasks are AIME 2025 (30 edition-

transfer questions), HMMT 2025 (30 family-transfer questions), and HMMT 2026 (33 family-transfer questions); BeyondAIME is an additional exact-integer transfer collection. The answer for every task is an integer scored by exact matching.

- **CMT-Bench.** The benchmark is organized by condensed-matter problem categories. The adaptation process uses ten rounds of 40 tasks. The held-out set contains five new tasks from categories represented in adaptation (category IID) and five tasks from novel categories (category OOD). The available task records retain the IID/OOD category partition and the common condensed-matter evaluator.
- **BiomniBench-DA.** The biomedical data-analysis tasks are grouped by study domain and analysis setting. The 30-task adaptation panel supplies the five-round bank construction. The 20-task held-out panel contains 14 tasks from study domains represented in adaptation and six tasks from new study domains or analysis settings. Its domain and analysis-setting inventory is: Oncology (8: cell composition, association testing, mutation analysis, clustering, survival analysis, TCR repertoire); General biology (2: predictive modeling); Immunology (1: cell–cell communication); Metabolic (3: association testing, differential expression); Neurology (4, Domain OOD: differential expression, co-expression networks, association testing, multi-omic integration); and Cardiovascular (2, Domain OOD: longitudinal analysis, multi-omic integration). Each task is evaluated with its expert rubric and criterion-level A/B/C scoring.

B.1.5 How adaptation and held-out tasks are selected

The adaptation panels are the benchmark-specific task collections used by the five-round or ten-round bank construction runs. Held-out panels are fixed collections evaluated after bank construction under the benchmark's native evaluator. GPQA uses the organic-chemistry, physics-subdomain, and biology partitions described above; mathematics uses edition and competition-family transfer; CMT-Bench uses seen versus novel categories; and BiomniBench-DA uses seen study domains versus new domains or analysis settings. These labels make the transfer unit explicit while preserving each benchmark's native task format and scoring rule.

B.1.6 Agents, baselines, and comparison conditions

The primary baseline is MemoHarness, a whole-harness adaptation system. It receives the same adaptation tasks and task-level budget as `CoRSI`, and its held-out result is evaluated with the same benchmark evaluator. Codex and Claude Code provide the corresponding native agent-harness records, with the model and reasoning effort fixed within each comparison. The `CoRSI` condition changes the retained object and inference-time composition: typed modules are retrieved from the frozen bank, assembled into a complete harness, and executed under the same task-facing evaluation contract.

The campaign uses GPT-5.5 for the main CMT-Bench and BiomniBench-DA runs, GPT-5.5, GPT-5.6 Terra, and GPT-5.6 Sol for the mathematics and GPQA comparisons, and Claude Haiku 4.5, Claude Sonnet 4.6, Claude Sonnet 5, and Claude Opus 5 for the corresponding Claude conditions. The CMT-Bench cross-model panel transfers a GPT-5.5-trained bank to Claude Sonnet 4.6. The external BiomniBench-DA rows reproduce the released task-level records for Biomni, EvoScientist, OpenScience (Synthetic Sciences), OpenScience (AI4S Research), and Wisp Science; these systems are reported as external reference systems rather than retrained `CoRSI` conditions (Huang et al., 2025; The Open Science Desktop Contributors, 2026; Xu et al., 2026).

For every paired baseline–`CoRSI` result, the comparison keeps the task IDs, model setting, reasoning effort, evaluator, tools, and task-level resource budget aligned. The benchmark-native evaluator remains the scoring authority: exact matching is used for mathematics and GPQA-Diamond, the CMT-Bench reference evaluator is used for CMT-Bench, and the Gemini rubric is used for BiomniBench-DA. The resulting design lets the main tables report endpoint performance while the bank summaries, cooperation matrices, and case studies explain which reusable modules and workflows produced that endpoint.

B.1.7 Executed mechanism ablations

The feedback analysis is instantiated on the same GPQA-Diamond protocol used by our main comparison: a fixed 98-task adaptation panel, five rounds, a fixed 100-task held-out panel, closed-book prompts, the same evaluator, and the same per-task budget. We ran the concrete paths exposed by the current GPQA-Diamond adapter for GPT-5.6 Terra and Claude Sonnet 5. Table 3 compares `CoRSI` with four ablation conditions.

`CoRSI` **w/o Workflow Context** keeps experience and module retrieval with the workflow context channel disabled for the answer call.

`CoRSI` **w/o Multi-module Composition** restricts the candidate pool to one field module and sets the workflow participant and invocation limits to one.

`CoRSI` **w/o Field-level Granularity** groups changed fields into one module per typed dimension.

`CoRSI` **w/o Module-guided Workflow** uses historical experience retrieval without module retrieval or workflow context.

Relative to `CoRSI`, removing workflow context reduces the held-out score by 4.0 points for Terra and 13.0 points for Sonnet 5. Restricting retrieval to a single field module reduces the score by 7.0 and 10.0 points,

Table 3: GPQA-Diamond ablations. Accuracy on the held-out panel.

Base model	CoRSI –	CoRSI w/o Workflow Context	CoRSI w/o Multi-module Composition	CoRSI w/o Field-level Granularity	CoRSI w/o Module-guided Workflow
GPT-5.6 Terra	**86.0%**	82.0%	79.0%	82.0%	83.0%
Claude Sonnet 5	**88.0%**	75.0%	78.0%	78.8%	76.0%

while dimension-level modules reduce it by 4.0 and about 9.2 points. Removing the module catalog reduces the score by 3.0 and 12.0 points. These arms therefore measure the concrete GPQA-Diamond implementation paths executed under the same task panel and evaluator. The GPQA-Diamond adapter uses a deterministic meta-planner to compile task-conditioned workflow context for one solver call.

B.1.8 Formal correspondence of operations and ablations

The controlled conditions are defined by removing one transition from the same state construction used by the complete method. Table 4 states the transition explicitly, so the ablation names refer to executable objects rather than to a list of informal features. The held-out endpoint is then evaluated with the same benchmark contract for every condition.

Table 4: Formal correspondence between CoRSI operations and controlled conditions. Each row names the state transition used by the complete method and the controlled condition that removes or replaces that transition.

Operation	Complete-method transition	Controlled condition
Typed factorization	$H(S) = H_0 \oplus S$ with recalled typed patches applied on their dimensions and base fields inherited elsewhere; evaluation receives the assembled $H(S)$.	w/o Field-level Granularity; w/o Multi-module Composition
Planner compilation	$P_t = \mathrm{Planner}(Z_t(x;\Sigma_t)) \mapsto (J_t, \{S_{j,t}\}, G_t)$, where J_t is the participant count, $S_{j,t}$ are typed module assignments, and G_t is the dependency graph.	w/o Workflow Context; w/o Module-guided Workflow
Guarded assembly	$H_{\mathrm{merge}} = \mathrm{Guard}(H^0, \Delta_1, \ldots, \Delta_n)$, followed by complete-harness evaluation.	w/o Cross-source Recombination; w/o Synthesis
Artifact routing	$a_i = \mathrm{Exec}(v_i, H_i, x, \mathrm{Pred}(v_i))$ and named artifacts are passed along edges of G_t before final synthesis.	w/o Artifact Handoff; w/o Multi-participant Workflow
Cooperation memory	$C_{t+1}(m,m') = C_t(m,m') + \mathbf{1}[(m,m') \subseteq S_q]\mathbf{1}[r_q \geq \tau]$; C_t conditions later module retrieval.	w/o Cooperation Memory; Shuffled Cooperation
Bank evolution	$\mathcal{B}_{t+1} = \mathcal{B}_t \cup \{q : q \in \mathcal{Q}_t\}$ and $\mathcal{M}_{t+1} = \mathcal{M}_t \cup \mathrm{Extract}(\{L_q\})$ for successful reusable modules and workflows.	w/o Experience Retrieval

B.2 Round-Resolved Cooperation Structure

The case studies inspect individual compositions; this appendix follows the state accumulated across training. We reconstruct cooperation from task receipts, inspect typed-module keys and their role-label rendering, and summarize the cooperation structure that emerges from completed workflows. The atlas is therefore a process-level measurement of the contextual value claim: an edge records that two typed roles repeatedly participated in the same successful composition, while its growth records reuse across rounds.

Each of the 400 CMT-Bench bank entries is indexed by training round and task. Matching uses reward, final output, and total tokens to identify each recorded participant assignment. For every successful episode, we add one to each unordered pair of participating modules. The reconstruction agrees with every stored pair

count, and hyphenated and underscored cooperation keys remain distinct so that the measured identities are not collapsed during aggregation.

The cumulative panels show a transition from isolated discoveries to a stable cooperation core. Early rounds contain only a few low-count edges, so the system has not yet established a preferred partner structure. From the middle rounds onward, the verifier–analyst, verifier–solver, and analyst–solver edges become the dominant block and intensify together; by round 10 their counts are 29, 28, and 28 in the role-level view. The repeated growth of these edges is evidence that the bank is retaining compatible controls rather than collecting unrelated patches. The small context–generation edge that persists from the early panels shows that not every useful relation belongs to the large cooperation core.

This pattern supports the compositional mechanism in two ways. The matrix is not a static summary of agent identities: it changes as new successful compositions enter the bank, and the same typed roles acquire different strengths as their collaborators accumulate. At the same time, the dominant edges are not tied to a single task record; they recur across rounds, giving the Planner a retrievable signal for assembling later participants. The atlas therefore connects the local interaction statistic to the global claim that module value is contextual and that successful combinations provide the next round's search signal.

B.3 Archived Case Evidence and Provenance

The preceding cases connect configurations to outputs. This appendix documents how answer receipts and campaign resource records are reconciled with the retained summaries, completing the provenance behind the reported results. The analysis ledger records task records, model, parser version, answers, targets, workflow metadata, and resource usage for the reported campaign records.

The completed GPT medium campaign contributes 2,940 training task-rounds and 600 held-out results across three models and the two harness conditions. The frozen fifth-round evaluation applies the prespecified task-level budget to both workflow conditions, with transport recovery recorded in the same ledger. Held-out correctness is 83/90 for GPT-5.5, 83/86 for GPT-5.6 Terra, and 89/93 for GPT-5.6 Sol (MemoHarness/`CoRSI`). The campaign retains reported tokens, cache fields, retries, and wall time for every condition.

These records provide more than a bookkeeping check. The task-level joins make it possible to compare the two harnesses on identical demands while retaining the execution fields needed to interpret a difference: answer correctness shows the outcome, workflow metadata shows the composed execution, and resource fields show the cost of producing it. Checkpoint identities connect the held-out results to the bank state used for inference, so the reported transfer numbers can be read together with the module and topology evidence elsewhere in this appendix. The provenance ledger is thus the bridge between the aggregate benchmark figures and the concrete state transitions that support the RSI and composition claims.

C Theoretical Guarantees

The main method separates reusable components from complete-solution reward. Here we formalize that separation through cooperative coevolution, contextual module value, and bank-mediated feedback. The propositions connect the implementation's editable objects to the recorded round measurements.

C.1 The Method Objects Used by the Theory

The theory uses the construction in Eqs. (10) and (9). A component is a typed module assigned to a specified participant slot. A selection S identifies those modules, and $\mathrm{Decode}_\theta(S)$ supplies inherited base fields, guarded assembly, participant routing, and the final answer operation. The fixed context θ includes a task distribution, evaluator, model, budget, base, adapter, topology, placement, and application order. Let

$$V_\theta(S) = \mathbb{E}_{x,\xi}[R(x, \mathrm{Run}(\mathrm{Decode}_\theta(S), x; \xi))], \qquad 0 \le R \le 1.$$

Here ξ denotes execution randomness. An unselected dimension inherits its base behavior. All compared selections are feasible under the guard. Changes to topology or default collaborators are represented by a new context θ in the corresponding comparison. The contextual marginal and pair interaction are

$$\delta_i(S) = V_\theta(S \cup \{i\}) - V_\theta(S), \tag{23}$$

$$\kappa_{ij}(S) = V_\theta(S \cup \{i,j\}) - V_\theta(S \cup \{i\}) - V_\theta(S \cup \{j\}) + V_\theta(S). \tag{24}$$

C.2 Constructive Reduction to Cooperative Coevolution

To instantiate classical cooperative coevolution (Potter & De Jong, 1994), restrict the planner to one participant and a fixed execution graph. For each dimension d, form a species $\mathcal{P}_d$ of interchangeable patches or compatible patch bundles. A representative $r_d \in \mathcal{P}_d$ fills that slot. A variation operator changes a member within $\mathcal{P}_d$ while preserving its field type. To evaluate a candidate m from species d, the restricted planner substitutes it for r_d and retains the other representatives. The complete-harness evaluator returns

$$f_d(m; r_{-d}) = V_\theta(\{r_1, \ldots, r_{d-1}, m, r_{d+1}, \ldots, r_6\}).$$

Species therefore correspond to dimension-specific candidate pools; representatives correspond to retained modules; variation generates a new patch; composition supplies collaborators; and selection updates the species and its representative using the complete-harness score.

Proposition 1 (Representative-based coevolution)**.** Under the restriction above, use the species initialization, variation, selection rule, and update schedule of a representative-based cooperative coevolution algorithm. If the decoder is total on the considered tuples, the restricted `CoRSI` transition implements that algorithm.

Proof. Map each species member to its typed patch bundle and map the representative tuple to the planner's single participant. Variation changes exactly the selected species slot. Decoding supplies all other representatives, so both algorithms assign the same contextual fitness f_d. The same species selection therefore yields the same next representative and population. This mapping commutes with each scheduled update; coupling evaluator randomness extends the correspondence to stochastic executions. □

The construction locates classical coevolution inside the compositional interface. Multiple participants, field-level selections, task-conditioned routing, and typed workflow programs enlarge the corresponding choices in the same construction chain.

C.3 Contextual Marginals and Recombination Gains

Equation (24) evaluates four decoded complete harnesses under one context θ. Its identity $\delta_i(S \cup \{j\}) = \delta_i(S) + \kappa_{ij}(S)$ shows exactly how a collaborator changes a module's marginal value. If $\kappa_{ij}(S) > -\delta_i(S)$, a nonpositive marginal becomes positive. This is a discrete improvement direction produced by changing collaborators.

Proposition 2 (Conditional recombination advantage)**.** Let $A = \{i_1, \dots, i_m\}$ be jointly feasible under the fixed decoder, with nonnegative singleton marginals and $\delta_{i_\ell}(\{i_1, \dots, i_{\ell-1}\}) \geq \delta_{i_\ell}(\varnothing)$. Then

$$V_\theta(A) - V_\theta(\varnothing) \geq \sum_{i \in A} \delta_i(\varnothing) \geq \max_{i \in A} \delta_i(\varnothing).$$

Proof. Telescope the union gain in the guard's fixed application order. Each successive contextual marginal dominates its singleton marginal by assumption; nonnegativity then bounds the maximum by their sum. □

The comparator retains one of the same discovered singleton edits. Composition retains their compatible union. The guard establishes feasibility, while complete-harness evaluation determines the utility of the surviving composition.

C.4 Bank-mediated State Update

The executed transition is a bank-mediated update of the retained solver state under the Planner:

$$\Sigma_{t+1} = \mathcal{T}(\Sigma_t, \mathcal{Q}_t), \qquad \Sigma_t = (\mathcal{B}_t, \mathcal{M}_t, C_t), \tag{25}$$

Completed episodes $\mathcal{Q}_t$ are appended to the bank, typed field deltas are retained in the catalog, and successful module co-occurrences update C_t. The Planner reads Σ_t and emits the next participant count, module assignments, roles, artifact dependencies, and operation order. Thus a successful composition changes the next candidate pool and the resulting solver workflow. The recursion updates the retained solver state and supplies candidate programs to subsequent Planner calls.

The state readouts are the held-out score, bank size, module-content frequencies, cooperation counts, and matched contextual interaction values. Together they connect each recorded state transition to the compositions available to the next Planner call.

C.5 Cooperation Memory and Matched Interaction Measurements

The runtime $C_t(u, v)$ in Eq. eqrefeq:co-success-update is indexed by retained modules. CMT-Bench visualizations may render role labels for readability, while each episode receipt retains the module bindings and complete configuration. To measure interaction of specific modules i, j, fix their placement and the remaining context θ, then evaluate $(i, j), (i, \perp), (\perp, j), (\perp, \perp)$, where $\perp$ means base inheritance. Their second difference estimates Eq. (24).

The history selects module partnerships for retrieval, while the matched intervention quantifies their contextual value on complete harnesses.

D Case Studies and Evidence

D.1 Comparative Case Studies: From Harness to Answer

The main results measure answer quality, while the training analysis records what becomes reusable. This appendix connects them through two same-task MemoHarness comparisons and a cross-model transfer case. We organize the evidence by the mechanism it explains: preserving an exact problem state, executing an artifact dependency graph, and transferring typed control fields. The central question is which operation `CoRSI` gets right and how its saved product contributes to the accepted answer. Every case follows one presentation template: task and evaluator, Planner composition, materialized artifacts, decisive verification, and the claim link. Colored headers use the same visual vocabulary across all cases: blue for task and baseline, teal for composed artifacts, generation color for products, verification color for checks, and teal claim panels for the final interpretation. Appendix D.5 supplies the complete participant records behind these analyses.

D.1.1 Case Selection and Reading Order

AIME and HMMT compare the same task, base model, and medium effort under the two methods. We identify each harness by its saved operational fields and request, preserving the original identities in the companion evidence file. CMT-016 addresses a different question: whether GPT-5.5-trained patches can configure a successful Sonnet 4.6 team. CMT-002/007 supply the separate configuration-merge evidence in Appendix A.2.

Table 5: Purpose and evidence of each case. Matched answer comparisons and structural examples have separate roles.

Case / Category	What CoRSI Gets Right	Evidence to Inspect
AIME II-13 / paired	Preserves the exact finite state and validates the correct candidate through disagreement.	Single-solver baseline 3; cross-round solver/verifier artifacts 248 and 48; exact arithmetic merge 248.
HMMT task 3 / paired	Reconstructs the minimizing branch and checks the exponent conversion at final synthesis.	MemoHarness baseline answer; three CoRSI participants and explicit upstream artifacts; $1/1152$ versus $1/576$.
Cross-model transfer	Reuses source-model controls to produce and retain a sector-count inventory in the target model.	Six GPT-5.5-bank patches, three Sonnet 4.6 roles, candidate inventory 9/7/7, evaluator answer 7.
Cross-source merge	Retains compatible cross-source changes and resolves conflicting field writes before execution.	Disjoint Context/Orchestration writes accepted; four colliding writes detected at the configuration boundary.

Table 5 maps each case to one concrete operation in the composition claim: exact-state preservation, artifact-routed verification, typed-field transfer, or guarded compatibility. A participant is always a complete harness $H_j = H_0 \oplus S_j$, where the retained patches S_j modify named fields and every remaining field inherits its base value. The task reward belongs to the final answer of the complete workflow.

D.1.2 Exact-State Preservation and Cross-Round Verification

Claim / Preserve the exact quantity across collaborators

This case explains how field-level reuse and a solver–verifier dependency preserve the finite rational problem until a final arithmetic audit. The comparison uses Claude Haiku 4.5 at medium effort in both methods.

The task defines

$$x_1 = \frac{25}{11}, \qquad x_{k+1} = \frac{1}{3}\left(x_k + \frac{1}{x_k} - 1\right), \qquad x_{2025} = \frac{m}{n},$$

with coprime positive integers m, n, and asks for $(m+n) \bmod 1000$. This target depends on the exact finite-index numerator and denominator.

Baseline / The saved MemoHarness configuration

MemoHarness executes one solver with systematic case-analysis framing. Its system instruction asks for exhaustive configurations, constraint checks, and a final answer after working. Generation specifies one candidate, temperature 0, and an output limit of 16,384; Orchestration is single-solver. Tools and Memory are disabled, and Verification requests a boxed integer. The few-shot strategy setting uses systematic case-analysis framing; the saved request contains the harness and problem.

The baseline initially derives rational updates and computes early terms. It then solves the fixed-point equation $(2L-1)(L+1) = 0$ and treats the large finite index as effectively equal to $L = 1/2$. Its decisive sentence asserts $x_{2025} = 1/2$, giving $m + n = 3$. That step replaces an exact finite-term question with an asymptotic approximation; the finite-index numerator–denominator sum differs from the limiting value. The evaluator extracts 3 and marks it incorrect against 248.

Table 6 distinguishes the local harness from the outer dependency graph. Both CoRSI participants retain a single-solver local policy; the planner connects their separate invocations. Four selected fields come from two training rounds, and all other values are inherited. The planner input includes 94 co-successes for the solver's module pair and 15 for the verifier's pair. These counts record the historical contexts supplied to selection. The baseline also requests verification, so the contrast is the independently materialized harnesses and retained artifacts through which verification is executed.



Figure 10: Role-aggregated cooperation across all ten training rounds. Each panel is one cumulative round; the fixed six-role ordering and common color scale make the emergence and persistence of cooperation edges readable at a glance. The underlying typed-module records are summarized by the role-level view and retained in the training ledger.

Table 6: AIME harnesses aligned by dimension. The MemoHarness column is the recorded whole configuration; the two `CoRSI` columns are materialized participants. R1/R2 denote source training rounds. Selected field patches are distinguished from inherited settings.

Dimension	**MemoHarness Solver**	**`CoRSI` Solver**	**`CoRSI` Verifier**
Context	Systematic case analysis; enumerate configurations and check constraints	R1 system instruction: enumerate branches, verify cases, combine results	R1 framing: enumerate cases and verify; base system instruction inherited
Tools	Disabled; no retrieval	Inherited: disabled	Inherited: disabled
Generation	Limit 16,384; temperature 0; one candidate	R1 limit 12,000; inherited temperature 0 and one candidate	R2 temperature 0.2; inherited limit 8,192 and one candidate
Orchestration	Single solver	Inherited single-solver local policy; upstream input: task	Inherited single-solver local policy; upstream input: solver artifact
Memory	Disabled	Inherited: disabled	Inherited: disabled
Verification	Boxed-integer formatting	Inherited boxed-integer formatting	Inherited boxed-integer formatting
Saved answer	3 (incorrect)	248	48

Artifacts / Two exact-recurrence artifacts

The solver retains the numerator–denominator sum s_k and derives

$$s_1 = 36, \qquad s_{k+1} \equiv 667 s_k^2 \pmod{1000}.$$

It obtains a period of 20 after the initial terms and proposes 248. The verifier receives the solver artifact, recomputes a sequence, and proposes 48 using a different period. Both derivations remain available to the final merge.

Verification / Locate the first divergent residue

The merge identifies the disagreement after $s_{11} = 608$:

$$608^2 \bmod 1000 = 664, \qquad 664 \cdot 667 = 442888 \equiv 888 \pmod{1000}.$$

The verifier's corresponding residue 288 is arithmetically inconsistent. The corrected sequence repeats $s_{23} = s_3 = 208$, giving period 20 and

$$s_{2025} = s_{3+((2025-3) \bmod 20)} = s_5 \equiv 248 \pmod{1000}.$$

The evaluator accepts 248.

Claim link / Preserve the correct derivation through disagreement

The solver's useful product is the exact modular recurrence together with candidate 248. When the verifier proposes 48, the merge retains both derivations and checks the first disputed multiplication. The verified residue 888 preserves the period-20 calculation and the correct answer. Cross-round field composition supplies the participant policies; artifact routing makes their calculations available to this final check.

This connects the same-task improvement, 3 to 248, to an observed operation: maintaining the finite exact state and validating the candidate from it. Figure 15 and the complete responses in Appendix D.5.1 trace that chain from selection to final answer.

D.1.3 From an Audit Instruction to an Executed Graph

Claim / Make the audit an artifact-consuming computation

This case isolates the organization of visible execution: all three `CoRSI` participants share four modules, but the third receives the first two artifacts. Both methods use GPT-5.5 at medium effort on the same task.

The task asks for the smallest xyz over positive real x, y, z satisfying

$$x^{\log_2(yz)} = 2^8 3^4, \qquad y^{\log_2(zx)} = 2^9 3^6, \qquad z^{\log_2(xy)} = 2^5 3^{10}.$$

Baseline / The saved MemoHarness configuration

Context requests classification, derivation, exhaustive branch checks, and independent cross-checking. Generation specifies candidate count 12, temperature 0.2, and an output limit of 8,192. Orchestration requests self-consistency with an exhaustive boundary/period audit and resolution of disputed cases by direct checking. The saved execution record reports one model call and one answer: $1/1152$.

The baseline response contains only the boxed fraction. Its observable discrepancy is a factor of two relative to the evaluator value, so the comparison below locates the exact arithmetic in `CoRSI`'s saved derivations. The candidate-count and audit fields describe the baseline policy supplied to that single invocation; the executed `CoRSI` graph contains separate participant calls and explicit artifact inputs.
Table 7 reports configured fields and recorded calls as separate entries. Each `CoRSI` participant selects the same two Context fields, one Generation field, and one Orchestration field; Tools, Memory, Verification, and unselected fields inherit the base. The initial branches produce independently addressable artifacts, and participant C receives both before synthesis. The receipts establish dependencies independently of wall-clock scheduling.

Table 7: Same-task execution organization on HMMT task 3. Configured candidate count and recorded model calls are separate quantities. CoRSI's five calls comprise planning, three participants, and final merging.

Property	**MemoHarness**	CoRSI
Context policy	Broad exhaustive-audit instruction	Two fields: derive/verify framing and system instruction
Generation policy	12 configured candidates; temperature 0.2	3 configured candidates; temperature 0; same 8,192 limit
Organization	Self-consistency/audit policy in one solver request	Two independent participants; a third depends on both
Recorded calls	One solver	Planner + three participants + merge
Saved artifacts	Final fraction $1/1152$	A: $1/576$; B: $1/288$; downstream C: $1/576$
Evaluator answer	Incorrect: $1/1152$	Correct: $1/576$

Artifacts / An algebraic reduction shared across artifacts

Set $a = \log_2 x$, $b = \log_2 y$, $c = \log_2 z$, and $t = \log_2 3$. Taking logarithms gives

$$a(b+c) = 8 + 4t, \quad b(c+a) = 9 + 6t, \quad c(a+b) = 5 + 10t.$$

The pairwise products are

$$ab = 6, \qquad bc = 3 + 6t, \qquad ca = 2 + 4t, \qquad b^2 = \frac{(ab)(bc)}{ca} = 9.$$

A supplies the derivation and $1/576$; B supplies only $1/288$. The downstream participant returns only $1/576$. The final merge receives all three saved artifacts and explicitly reconstructs the algebra above.

Verification / Reconstruct the minimizing branch

For the minimizing branch, $b = -3$, $a = -2$, and $c = -1 - 2t$. Thus

$$a + b + c = -6 - 2t, \qquad xyz = 2^{-6-2t} = \frac{1}{64 \cdot 9} = \frac{1}{576}.$$

The positive branch gives the reciprocal, so the negative branch is the minimum. This exact exponent conversion validates the final CoRSI answer and distinguishes it from both the baseline's $1/1152$ and artifact B's $1/288$.

Claim link / Validate the minimum from the constraints

The final merge converts candidate disagreement into a concrete check. It derives $b = \pm 3$, selects the branch with smaller $a + b + c$, and evaluates $2^{-6-2\log_2 3} = 1/576$. The graph supplies the candidate artifacts; this recorded computation establishes which value satisfies the minimization problem. The accepted answer is backed by the complete branch calculation.

The same-task improvement is $1/1152$ to $1/576$, with the successful branch selection and exponent conversion visible in the final merge. Figure 16 and Appendix D.5.2 give the corresponding participant configurations and complete outputs.

D.1.4 Transfer Note

The aggregate CMT-Bench result in the main text supplies the cross-model transfer evidence. The typed bank and topology records document the reusable control representation used by that transfer.

D.1.5 From the Cases to the Composition Claim

The two mathematical cases expose complementary composition behaviors. AIME preserves an exact recurrence and checks a disputed residue; HMMT validates the minimizing branch from the original constraints. The aggregate CMT-Bench transfer result in the main text supplies the cross-model evidence, while the retained bank and topology records provide the reusable typed-control view.

The cases identify a useful intermediate product and the operation that carries it into a correct answer. AIME preserves an exact recurrence and checks a disputed residue; HMMT validates the minimizing branch from the original constraints; CMT-Bench retains a sector inventory under transferred control fields. The configuration examples in Appendix A.2 then explain which cross-source changes can coexist: disjoint fields survive together, and same-field collisions are resolved before commit.

These observations align the modification and evaluation units. A module carries a patch and source history; a participant inherits a complete base; the workflow connects participants through artifacts; and the evaluator scores its final answer. Successful training episodes feed the bank and cooperation counts examined in Appendix B.2. This sequence makes discoveries reusable in subsequent collaborator contexts and supplies the concrete state transition behind compositional improvement.

D.2 Experience Bank Case Study

The BiomniBench-DA training campaign provides a direct view of what the two systems retain. Both banks contain the same 150 completed training episodes (30 tasks over five rounds). The distinction is the organization of their contents: `CoRSI` stores typed field patches together with typed-module cooperation, while the MemoHarness bank stores the whole-harness progression as task-specific patch contents. Panel F of Figure 11 reports the round-resolved counts from the serialized bank snapshots.

At round 5, `CoRSI` has 2,220 module occurrences, 33 distinct patch contents, and 36 positive cooperation edges. The corresponding MemoHarness snapshot has 2,297 module records, 200 distinct patch contents, and no cooperation edges. The repeated `CoRSI` contents are the compact typed controls that the Planner can recombine; the round counts show their reuse accumulating as the cooperation matrix grows.

D.2.1 What the bank contains

The retained contents are operational controls that shape task execution. They cover the six harness dimensions: rubric-oriented task framing, workspace and tool discovery, bounded generation, progress-based orchestration, sliding-window memory, and artifact-gated verification. Table 8 shows one representative content family for each dimension.

Table 8: Representative reusable contents in the `CoRSI` bank. These patches recur across the 150 training episodes and enter later Planner recalls as typed candidates.

Dimension	Retained content	Operational role
Context	Inspect verifier requirements before declaring completion.	Align analysis with rubric evidence.
Tools	Inspect the workspace root, probe relevant paths, and use lightweight local probes.	Establish data and tool state.
Generation	Use a bounded completion budget with deterministic single-candidate generation.	Stabilize long analysis turns.
Orchestration	Use progress-based orchestration with an audit pass and a wall-clock safety boundary.	Continue while state changes and close after verification.
Memory	Keep the task anchor and recent tool observations in a sliding window.	Preserve actionable context.
Verification	Treat command errors as observations and require workspace checks before completion.	Gate the final answer on saved artifacts.

These contents form a reusable vocabulary for scientific execution. For example, the orchestration and memory patches jointly turn a long analysis into a sequence of observable progress states, while the verification patch converts the saved files and checks into an acceptance condition. The cooperation matrix records which such fields recur together in successful episodes, so the next Planner composition can retrieve both the content and its collaborator context.

D.2.2 Bank content and task outcomes

The bank view links stored contents to the task-level outcomes in Section 4.4. `CoRSI`'s 33 distinct contents are reused across the 30 training tasks, and the final 20-task held-out panel reaches a mean rubric score of 73.20. The bank therefore supplies the intermediate object between adaptation and evaluation: typed controls are retained once, then assembled into complete scientific workflows for new demands.

D.3 Rubric Profiles and Scientific Execution

The biomedical comparison extends the complete-harness evaluation interface from final-answer correctness to analysis quality. We first specify the common task panel and signed rubric, then expose every threshold and the

individual task scores. Two case studies connect reusable configurations to executed analyses, saved scientific products, and the judge's deductions.

D.3.1 Task Panel, Reward, and Selection Protocol

The 50-task DA collection is the entire task pool. Thirty tasks support adaptation and twenty support frozen-bank evaluation. Both MemoHarness and `CoRSI` use GPT-5.5 at xhigh reasoning effort for one training run of five rounds; each round executes the same 30 adaptation tasks and seals an incremental bank checkpoint. Inference assembles a task-specific harness from the frozen bank, with bank updates defined by the adaptation panel. The solver workflow has a nominal 3,600-second budget, excluding planning. The tool-enabled adapter permits up to five participants and executes their artifact handoffs sequentially. The two developed cases each use three.

The 14-task IID label denotes new studies in adaptation-covered categories: oncology, general biology, immunology, and metabolism. Two tasks use unseen task types. The six OOD tasks cover neurology (four tasks from DA-15) and cardiovascular analysis (two from DA-6), forming two held-out study groups. Figure 11 fixes the exact evaluation IDs.

Gemini 3.1 Pro assigns an A/B/C grade to each task-specific criterion. The evaluator maps each grade to its specified signed weight, including source-reliability penalties of 0, -5, or -10 where prescribed. It retains the raw sum, clips the reported task score s_i to $[0, 100]$, and uses $r_i = s_i/100$ as the continuous training reward. A full-success flag requires $s_i = 100$. In this adapter, the co-success matrix records each selected module pair once for every full-credit episode, while continuous task rewards update the bank. CMT-Bench and Biomni use the same typed-module cooperation key; CMT-Bench uses $\tau = 0.5$, while Biomni uses full credit with $\tau = 1$. Terminal incomplete-task timeouts receive zero under the recorded policy, with their logs preserved.

What the rubric measures. The rubric is a task-specific, evidence-based scoring contract rather than a single generic quality label. Each task contributes criteria that correspond to the scientific operations required by its prompt: data preparation, a valid analysis, interpretation of the resulting quantities, and traceable sources. The evaluator sees the task question, the submitted answer and artifacts, and the task's criterion descriptions; it returns one of the prescribed evidence levels for every criterion. The criterion weights and level descriptions are part of the BiomniBench-DA task specification (Huang et al., 2025); the criterion-level records in this appendix are the evidence used to reconstruct the reported scores. Table 9 makes the operational meaning explicit for the two detailed cases: an A grade requires the listed observable evidence, B records a partial execution, and C records the absence or invalidity of that evidence. The source-reliability criterion is signed: it leaves a fully traceable answer unchanged and subtracts points when values, methods, identifiers, or scientific claims cannot be traced.

For a task with criteria $j = 1, \ldots, m$, the evaluator score is computed as

$$s_i = \mathrm{clip}_{[0,100]}\left(\sum_{j=1}^{m} w_{ij}(g_{ij})\right), \qquad g_{ij} \in \{A, B, C\}, \tag{26}$$

where w_{ij} is the task-provided weight for grade g_{ij}. For example, DA-18-1 has seven criteria with full-credit weights $15 + 10 + 15 + 20 + 20 + 20 + 0 = 100$; a B on TMB changes its contribution from 15 to 7, while an untraceable source criterion contributes -10. DA-1-3 has nine criteria, including baseline filtering, tissue-normalized enrichment, tumor-specificity logic, biological annotation, and translational interpretation. Thus the nine columns S1–S9 in the task ledger are score slots for the task criterion sequence, while the named case tables expose the underlying scientific meanings and grade weights.

The final `CoRSI` DA-10-3 execution scores 100; the other nineteen held-out task scores are unchanged. The twenty-task aggregate is 73.20, with IID 73.43 and OOD 72.67. Its workflow completes in 1,864.877 seconds, and the execution record records the participant artifacts and rubric evaluations.

Seven historical baselines are projected onto these same twenty task IDs: Codex, Biomni, and Pi with GPT-5.5; EvoScientist, OpenScience (Synthetic Sciences), OpenScience (AI4S Research), and Wisp Science with DeepSeek V4 Pro.[1] All statistics are recomputed from continuous task scores, so Full always agrees with pass@1.0. The aligned panel uses the same task identities and rubric family.

D.3.2 Complete Quality-Threshold Profiles

For a group of n tasks, we report

$$\mathrm{pass@}t = \frac{1}{n}\sum_{i=1}^{n} \mathbf{1}[s_i \geq 100t], \qquad \mathrm{AUC}_{10} = \frac{1}{10}\sum_{j=1}^{10} \mathrm{pass@}(j/10). \tag{27}$$

The dashboard summarizes all eleven cutoffs $t = 0.0, 0.1, \ldots, 1.0$ using integer score comparisons. At $t = 0$, even a zero-score task passes, so all systems have 100%; NZ separately counts $s_i > 0$. Full counts $s_i = 100$.

[1] Released project names: `https://github.com/EvoScientist/EvoScientist`, `https://github.com/synthetic-sciences/openscience`, `https://github.com/ai4s-research/open-science`, and `https://github.com/xuzhougeng/wisp-science`.

Table 9: Operational BiomniBench-DA rubric used by the Gemini 3.1 Pro evaluator. Each task supplies its own criterion set; the two panels below are the complete criterion maps for the detailed cases reported in this appendix. A, B, and C are the evaluator's evidence levels, with the signed points shown in parentheses.

Task / criterion	**Evidence required for A**	**Partial or absent evidence**
DA-18-1: genomic landscape of breast-cancer receptor subtypes		
Data loading (15)	Link clinical, mutation, and copy-number tables and retain sample-level identifiers.	B (7): partial links; C (0): missing or incorrect data.
Subtype classification (10)	Define HR/HER2 status and assign all four receptor subtypes.	B (5): partial definition; C (0): no valid classification.
TMB analysis (15)	Report subtype-wise medians and interquartile ranges with quantitative comparisons.	B (7): incomplete or partly incorrect statistics; C (0): absent analysis.
Driver genes (20)	Count recurrent genes by subtype and report subtype-specific frequencies.	B (10): incomplete or unstratified drivers; C (0): absent/irrelevant analysis.
ERBB2 alterations (20)	Separate point mutations from amplifications, quantify both, and interpret subtype enrichment.	B (10): counts without the required separation; C (0): absent analysis.
Biological synthesis (20)	Connect subtype drivers, TMB, ERBB2 events, and clinical meaning in a coherent interpretation.	B (10): incomplete synthesis; C (0): incorrect or absent interpretation.
Source reliability (0, -5, -10)	Trace values and identifiers to data or named sources and cite biological claims.	B (-5): some unsupported claims; C (-10): untraceable or fabricated evidence.
DA-1-3: tumor-specific cell-subpopulation enrichment		
Baseline/tissue filtering (13)	Restrict to baseline and Tumor, Normal, and Blood compartments.	B (8): missing one restriction; C (0): incorrect filtering.
Subtype distribution table (13)	Construct a consistently ordered SubCellType-by-tissue count or proportion table.	B (8): misaligned table; C (0): wrong grouping or no table.
Enrichment metric (16)	Normalize for unequal tissue sizes using Ro/e or within-tissue fold change.	B (10): raw-count or MajorCellType error; C (0): no valid metric.
Metric interpretation (14)	Interpret values above/below neutrality as tissue enrichment/depletion and use them correctly.	B (9): ranking without biological meaning; C (0): incorrect interpretation.
Tumor ranking (13)	Rank all subtypes and identify the top tumor-enriched populations with values.	B (8): incomplete ranking; C (0): absent or incorrect ranking.
Specificity filter (16)	Require enrichment in tumor and lower representation in both normal tissue and blood.	B (10): one-control or unjustified threshold; C (0): no valid filter.
Biological annotation (9)	Explain myeloid, stromal, endothelial, and exhausted-T-cell identities using the observed labels.	B (5): shallow annotation; C (0): absent or incorrect annotation.
Translational interpretation (6)	Link mechanisms and therapeutic relevance, while stating a data-grounded limitation.	B (3): generic implications; C (0): unsupported or absent implications.
Source reliability (0, -5, -10)	Trace labels, values, metrics, and biological claims to data, methods, or literature.	B (-5): partial tracing; C (-10): untraceable or fabricated evidence.

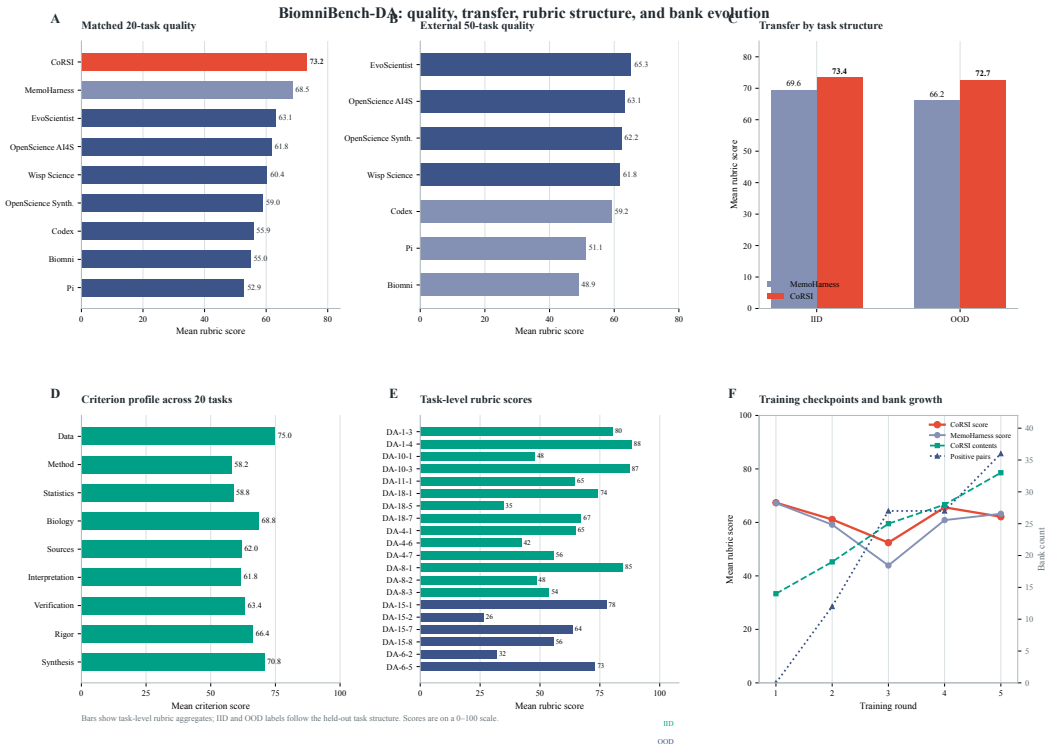


Figure 11: BiomniBench-DA quality, transfer, rubric structure, and bank evolution. Panel A reports the matched 20-task mean rubric score; panel B reports the external 50-task mean. Panel C compares IID and OOD held-out means. Panel D summarizes the nine rubric criterion slots, panel E shows the task-level ledger, and panel F traces training checkpoints together with the growth of the retained bank.

The reported AUC averages the ten positive thresholds, preserving the historical definition; it is the discrete average over the positive thresholds.

Figure 11 gives the overall distribution. `CoRSI` leads on mean score and AUC, with twenty nonzero tasks and no score below 40. Its median of 67.50 is below MemoHarness's 69.00, and at threshold 0.7 its count is 9 rather than 10. Both reach full credit on three tasks.

Panel C of Figure 11 resolves seen-category and unseen-category transfer. Compared with MemoHarness, `CoRSI` gains 3.86 points on IID and 6.50 on OOD. On the six OOD tasks, `CoRSI` has five scores at least 60, versus four for MemoHarness, but the same two scores at least 80. Figure 11 reports every task-level score; for example, DA-6-5 falls from MemoHarness's 100 to 77. The grouped panel reports the corresponding domain-level means.

D.3.3 Complete External Baseline Landscape

The released external comparison contains all 50 BiomniBench-DA tasks. We report it separately from the 20-task matched panel because `CoRSI` and MemoHarness used the remaining 30 task IDs for adaptation. The external rows use the same Gemini 3.1 Pro rubric and retain each system's published model label: GPT-5.5 for Codex, Biomni, and Pi, and DeepSeek V4 Pro for EvoScientist, OpenScience (Synthetic Sciences), OpenScience (AI4S Research), and Wisp Science.

Figure 11 reports the complete external task-keyed distribution. The complete task-keyed source supplies the external comparison, and every reported score is recomputed from task-level values.

D.3.4 Selected Cases: From the Question to Scored Scientific Products

The case-study section uses the common template: task and evaluator, Planner composition, materialized artifacts, decisive verification, and claim link. DA-18-1 follows a complete scientific product; DA-1-3 follows a reusable enrichment SOP and artifact handoff. Figure 12 aligns the paired outcomes.

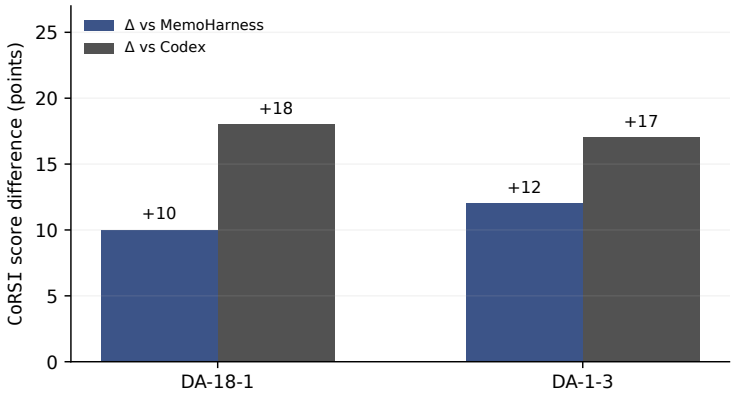


Figure 12: Positive paired contrasts used for the mechanism cases. Each bar is the CoRSI Gemini-rubric score minus the score of the named comparator on the same task. The paired records align each CoRSI execution with its comparator on the same task, making the plot an execution-level mechanism contrast.

Table 10: DA-18-1's selected controls and provenance. All source occurrences are from round 5. The table summarizes the retained fields and their source tasks.

Source task	Field	Retained content
DA-3-5	Context strategy	Inspect acceptance requirements before completion.
DA-3-5	Tools access	Local command-line tools.
DA-3-5	Context refinement	Remove examples, structured instruction, compression.
DA-3-5	Tools strategy	Inspect workspace first; use lightweight local probes.
DA-17-5	Tools strategy	Native command with fallback and bounded bootstrap.

Task / DA-18-1: receptor subtype and genomic alterations

The question is how breast-cancer genomic alterations differ among HR+HER2−, HR+HER2+, HR−HER2+, and triple-negative breast cancer (TNBC). The saved script links clinical, mutation, and copy-number files by sample ID, using receptor labels of the sequenced sample. It excludes 104 ambiguous samples and retains 1,814: 1,362, 132, 70, and 250 in the four groups. Codex and MemoHarness recover the same group counts. The comparison therefore turns on what the analyses preserve and explain after the initial data preparation.

Planner / Typed patch assembly

The Planner selects five patches from two successful round-5 source tasks. Table 10 exposes their exact fields and provenance. The selected Context and Tools fields form the task-specific participant configuration; unselected dimensions inherit the base harness.

Artifacts / Three complete participants

1. Context/retrieval (713.559 s). Context strategy and Tools access from DA-3-5 configure the first participant. It writes a saved analysis script, a context summary, an analysis trace, and CSV/JSON result tables. Its scoped handoff deliberately leaves final-answer creation to subsequent participants.

2. Strategy/review (486.843 s). The Context deletion and workspace inspection strategy configure a new participant in the same workspace. It reruns the analysis, checks subtype counts and statistics against the tables, and writes the final synthesis.

3. Independent tool check (179.070 s). The DA-17-5 Tools strategy configures the final participant. It reruns the saved Python script, checks syntax and required trace headings, and confirms the four subtype counts. All three processes return successfully; total workflow time is 1,379.645 s.

The products are checkable intermediate scientific results. The saved trace retains the tumor mutational burden comparison and the ERBB2 mutation-versus-amplification distinction used by the final answer.
The left panel of Figure 13 locates the score difference. Codex receives full credit for data linkage, subtyping, driver frequencies, ERBB2 separation, and source reliability. Its judge records incomplete TMB summaries and omission of the TMB comparison from the final answer (7/15), plus missing clinical interpretation of ERBB2 mutations in HER2-negative disease (10/20). These deductions exactly explain $100 - 82 = 18$. MemoHarness receives full TMB credit but the same ten-point synthesis deduction, scoring 90. Its final answer supports that distinction: it discusses TMB but omits the non-amplified ERBB2-mutant subgroup. The four external systems score 80, 72, 82, and 0 in table order. Their saved answers reveal a more specific contrast: all three nonempty answers discuss ERBB2 amplification, while the final product records the 13 non-amplified ERBB2-mutant TNBC samples identified by `CoRSI`.

Verification / Rubric criterion path

The rubric assigns `CoRSI` 100, MemoHarness 90, and Codex 82. The decisive criteria are the TMB comparison and the interpretation of non-amplified ERBB2 mutations in TNBC; the saved tables and final answer carry both findings.

Claim link / Composed scientific execution

Cross-source Context/Tools selection creates complete participants with inherited generation, memory, and verification fields. The shared script and result tables make prior work available for subsequent execution and checking, and the final answer preserves findings that the comparators omit. This is a concrete path from retained controls to a higher-scoring scientific product, with the accepted patches, artifacts, and rubric checks recorded in the case analysis.

Task / DA-1-3: tumor-specific cell-population enrichment

The question asks which annotated cell subtypes are specifically enriched in colorectal-cancer tumor tissue relative to adjacent normal tissue and blood. The saved trace validates barcode order and sample joins, restricts the primary tissue panel to Tumor/Normal/Blood, and uses the curated subtype labels. It analyzes 279,886 tumor, 260,294 normal, and 417,162 blood cells from 22 patients. `CoRSI` scores 92, compared with 80 for MemoHarness and 75 for Codex; the external scores remain visible in Figure 12.

Planner / Patient-level enrichment program

The Planner composes a reusable scientific SOP: subtype-by-tissue tables, patient-level proportions, paired Tumor–Normal and Tumor–Blood comparisons, Benjamini–Hochberg adjustment for 91 tests, and a three-fold enrichment rule with low control abundance and FDR below 0.05.

Artifacts / Reusable scientific products

The handoff preserves the script, intermediate tables, and trace for subsequent participants to rerun and verify. The analysis identifies tumor-selective exhausted T-cell, stromal, and macrophage populations, including c23_CD8_Tex_LAYN, c79_Fibro_FAP, c70_Endo_COL4A1, and c64_Mph_SPP1.

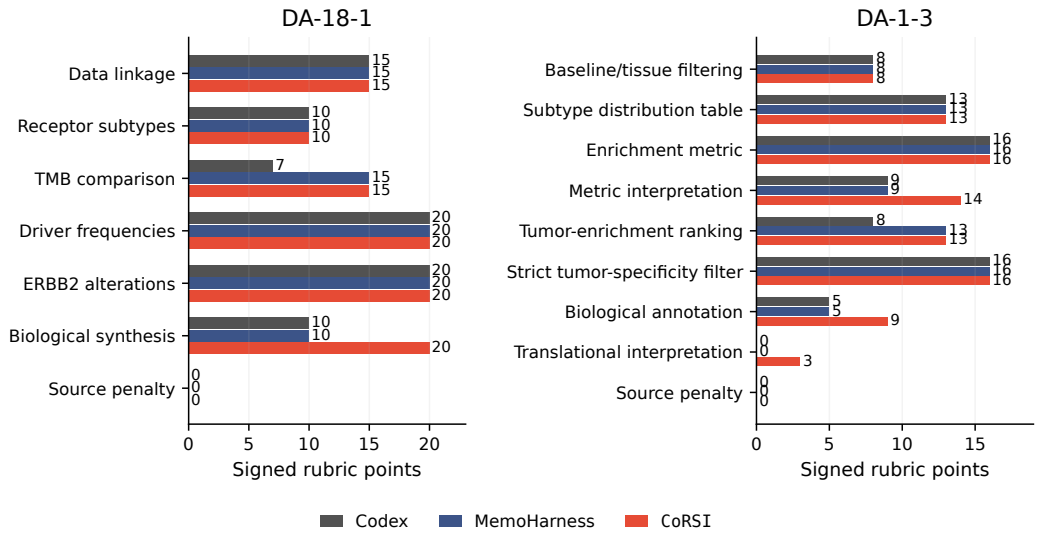


Figure 13: The rubric localizes criterion-level gains. Left: DA-18-1 gains in TMB comparison and biological synthesis. Right: DA-1-3 gains in metric interpretation, ranking, and biological annotation. Reliability criteria have a maximum of zero and allow negative penalties; zero there is full credit.

The right panel of Figure 13 shows where the advantage arises: the composition trace receives full credit for the enrichment metric, strict specificity filter, ranking, and biological annotation.

Verification / Criterion-level score path

The rubric assigns `CoRSI` 92, MemoHarness 80, and Codex 75. The decisive criteria are patient-level enrichment, the specificity filter, ranking, and biological annotation; the saved tables expose each decision rule.

Table 11 compares the same question's explicit analysis choices. The three Fisher-test traces operate on pooled cell counts; `CoRSI` instead declares the patient the biological replicate and uses paired patient-level fractions. OpenScience (Synthetic Sciences) explicitly rejects patient-level testing, whereas `CoRSI` records complete Tumor/Normal/Blood coverage for its 22-patient panel. The table reports the unit of inference and the tumor-specificity rule alongside the score.

Table 11: DA-1-3: scientific SOP choices in saved traces. Scores come from the aligned Gemini rubric; the choices come from the released traces.

System	Inferential unit / test	Tumor-specificity rule	Score
EvoScientist	Pooled cells; Fisher tests	Both controls; FDR; four-fold enrichment	84
OpenScience (Synthetic Sciences)	Pooled cells; Fisher tests	Both controls; FDR; two-fold enrichment	77
OpenScience (AI4S Research)	Pooled cells; Fisher test	Two-fold vs both; FDR; $>$ 50% tumor localization	89
Wisp Science	Cell-count observed/expected ratio	$>$ 70% tumor; low controls; ratio $>$ 1.5	87
`CoRSI`	22 paired patients; Wilcoxon tests	Both controls; FDR; three-fold enrichment; low control abundance	92

The trace separates enrichment from exclusivity: `CoRSI` keeps the epithelial tumor population as tumor-enriched and records its 1.80% presence in adjacent-normal cells. EvoScientist and OpenScience (AI4S Research) include this population under their stated enrichment rules. The `CoRSI` handoff retains both the decision rule and the intermediate tables for subsequent participants to rerun. The evidence supports reusable scientific execution under composed controls, with the decision rule and intermediate tables retained for subsequent analysis.

The execution records retain planning, execution, judging, and artifact fields for both cases; the case analysis focuses on the scientific products and rubric paths.

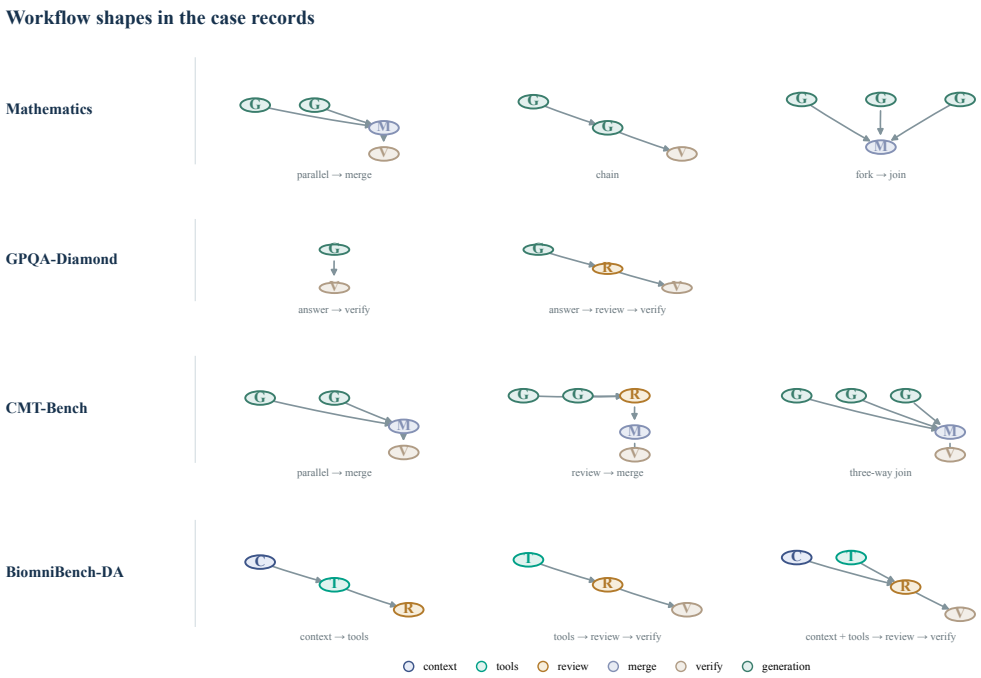


Figure 14: Workflow shapes in the case records. Rows group the observed workflow shapes by benchmark. Letters identify typed stages and arrows show artifact flow.

D.3.5 Checkpoint Evidence

Panel F of Figure 11 follows all five saved adaptation checkpoints. `CoRSI` accumulates 420 to 2,220 module occurrences and 14 to 33 distinct dimension–patch contents. The panel separates module occurrences from distinct dimension–patch contents. Positive typed-module cooperation pairs grow from zero to 36; their accumulated unordered counts are 0, 21, 48, 69, and 97. The resulting cooperation history is available to later Planner compositions.

Training mean score is nonmonotonic for both methods. `CoRSI` moves through 67.37, 61.10, 52.43, 65.63, and 62.13; the corresponding MemoHarness means are 67.23, 59.17, 43.90, 60.87, and 63.23. These are online adaptation outcomes under successive configurations, including recorded incomplete-task zeros.

The traces retain local typed edits, cross-source module composition, and downstream inspection of earlier scientific products.

D.3.6 Reproducibility and Data Release

The evidence package contains the complete system–task scores, split labels, retained module patches, participant configurations, scientific summaries, criterion judgments, and checkpoint statistics. The matched-panel tables and figures are regenerated directly from these records. The task-level tables and aggregate results agree on task coverage, threshold profiles, and signed criterion sums.

D.4 Workflow Shapes

The Planner instantiates a typed workflow for each demand. A topology records participant roles and artifact dependencies, so it captures how a harness is executed rather than only which modules it contains. Figure 14 collects the observed shapes by benchmark and makes the shared execution vocabulary visible.

The atlas gives a concrete explanation for why the same module bank can serve multiple demands. Mathematics uses parallel–merge, sequential chain, and fork–join forms: independent derivations can be combined when their evidence is complementary, while a chain carries a verified intermediate result to the next stage. GPQA-Diamond uses a direct answer–verify path for compact questions and inserts a review stage when an intermediate answer needs an explicit check. CMT-Bench expands the same grammar into parallel–merge, review–merge, and three-way joins, matching its need to compare several candidate harnesses before committing a transfer. BiomniBench-DA uses context–tools, tools–review–verify, and context-plus-tools–review–verify paths, which reflects the fact that scientific analysis must first make data and tooling available before a result can be checked.

Two structural patterns support the composition claim. First, the stage types are reused across rows, while the edges change with the demand; this is module reuse together with topology adaptation, rather than a separate workflow language for each benchmark. Second, a single benchmark contains more than one shape. The

Planner therefore selects a workflow conditional on the demand while retrieving from the same typed vocabulary. The resulting variation is the execution-level counterpart of the module-level recombination measured by the cooperation matrix: modules can recur across tasks, yet their useful ordering and artifact routes remain task-specific.

D.5 Compact Held-Out Agent Walkthroughs

Appendix D.1 maps each case to a composition-claim operation. This appendix keeps the corresponding workflows in a compact visual form: task, Planner composition, participant artifacts, verification, and the accepted answer. The figures expose the execution graph together with the saved products and verification steps.

Context **Tools** **Generation** **Orchestration** **Memory** **Verification** Context sets instructions and problem framing; Tools sets tool access and external retrieval; Generation controls candidate count, length, and sampling; Orchestration defines workflow and stopping rules; Memory retains task history; Verification sets format, validation, and fallback. Every participant is a complete six-dimensional harness, with unselected fields inherited.

Writing $H_a = \mathrm{GuardMerge}(H_0, S_a)$ fixes the unit distinction: S_a contains field patches, and the evaluated object is a complete workflow with its final answer. The three cases below show exact-state preservation, artifact-routed verification, and typed control transfer.

D.5.1 Exact-State Preservation

Planner / two-role exact-state workflow

Claude Haiku 4.5 at medium effort. The Planner assigns a solver and a verifier. The solver combines a round-1 Context patch with a round-1 Generation patch; the verifier combines a round-1 Context patch with a round-2 Generation patch and consumes the solver artifact.

Artifacts / finite recurrence and independent residue

The solver retains the numerator–denominator recurrence and proposes 248. The verifier recomputes the sequence and proposes 48. Both products enter the merge, which checks the first disputed residue and retains the period-20 calculation.

Verification / accepted answer

The merge verifies $608^2 \bmod 1000 = 664$ and $664 \cdot 667 \equiv 888 \pmod{1000}$, then evaluates $s_{2025} = 248$. The complete workflow is accepted against the evaluator answer 248.

D.5.2 Artifact-Routed Verification

Planner / three-participant dependency graph

GPT-5.5 at medium effort. The Planner instantiates three participants with a shared typed module set: two independent algebraic solvers and a synthesizer that receives both upstream artifacts before the final merge.

Artifacts / shared algebraic reduction

The independent branches produce $1/576$ and $1/288$. The synthesizer receives both derivations and reconstructs the pairwise-product identity, preserving the candidate branch information for verification.

Verification / minimizing branch

The final check derives $b = \pm 3$, selects the minimizing branch, and evaluates $2^{-6-2\log_2 3} = 1/576$. The artifact graph therefore carries the exact branch calculation into the accepted answer.

E Related Work and Positioning

The main related work locates CoRSI between agent composition and RSI. This appendix explains the development behind that position: feedback first revises an answer, then becomes persistent experience, then changes programs and improvement operators. Each stage identifies the retained object and the feedback selecting it, connecting prior work to our modification-unit question.

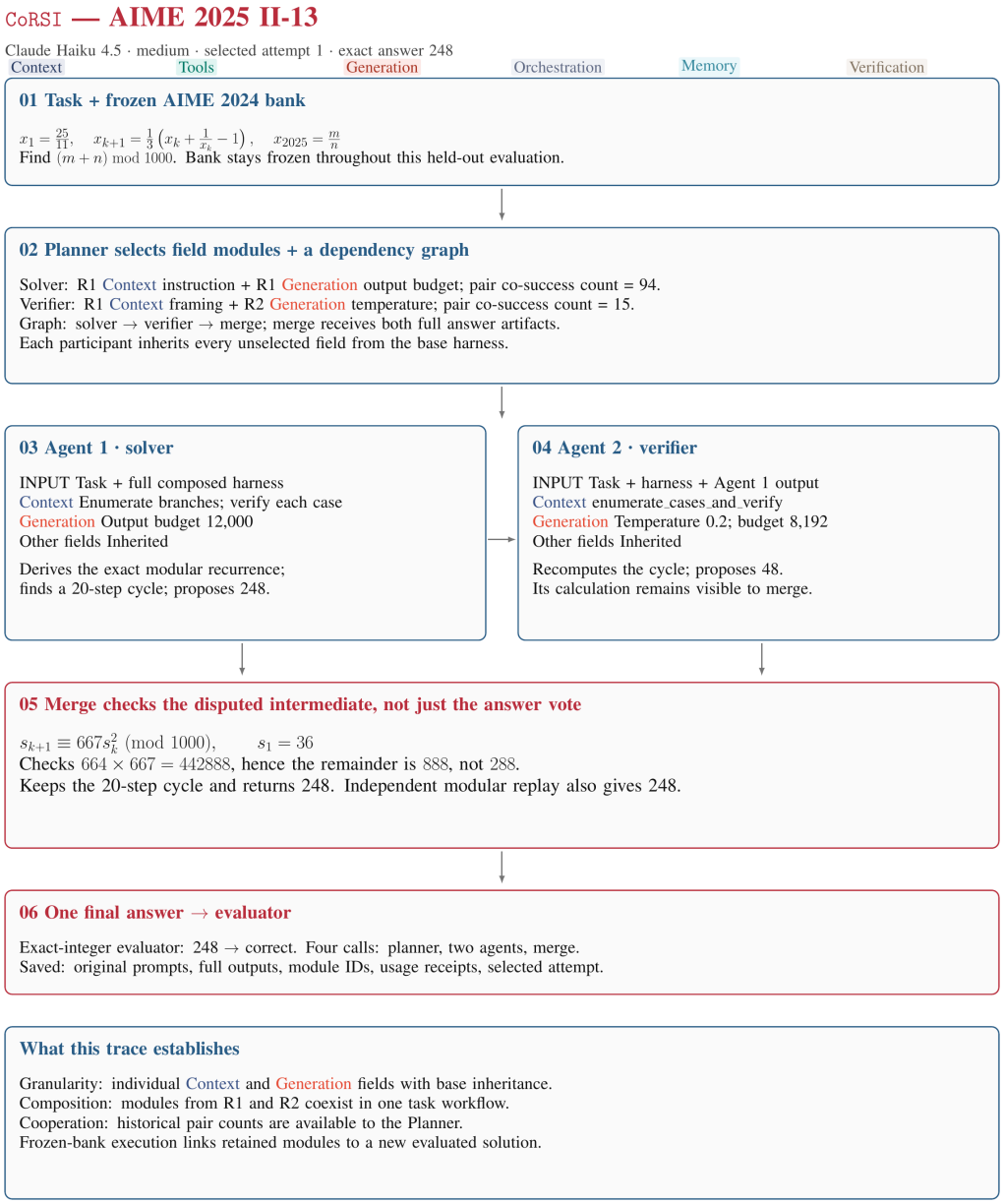


Figure 15: Case workflow / AIME II-13. Task, Planner composition, participant artifacts, and final arithmetic verification are read from top to bottom; arrows identify artifact dependencies.

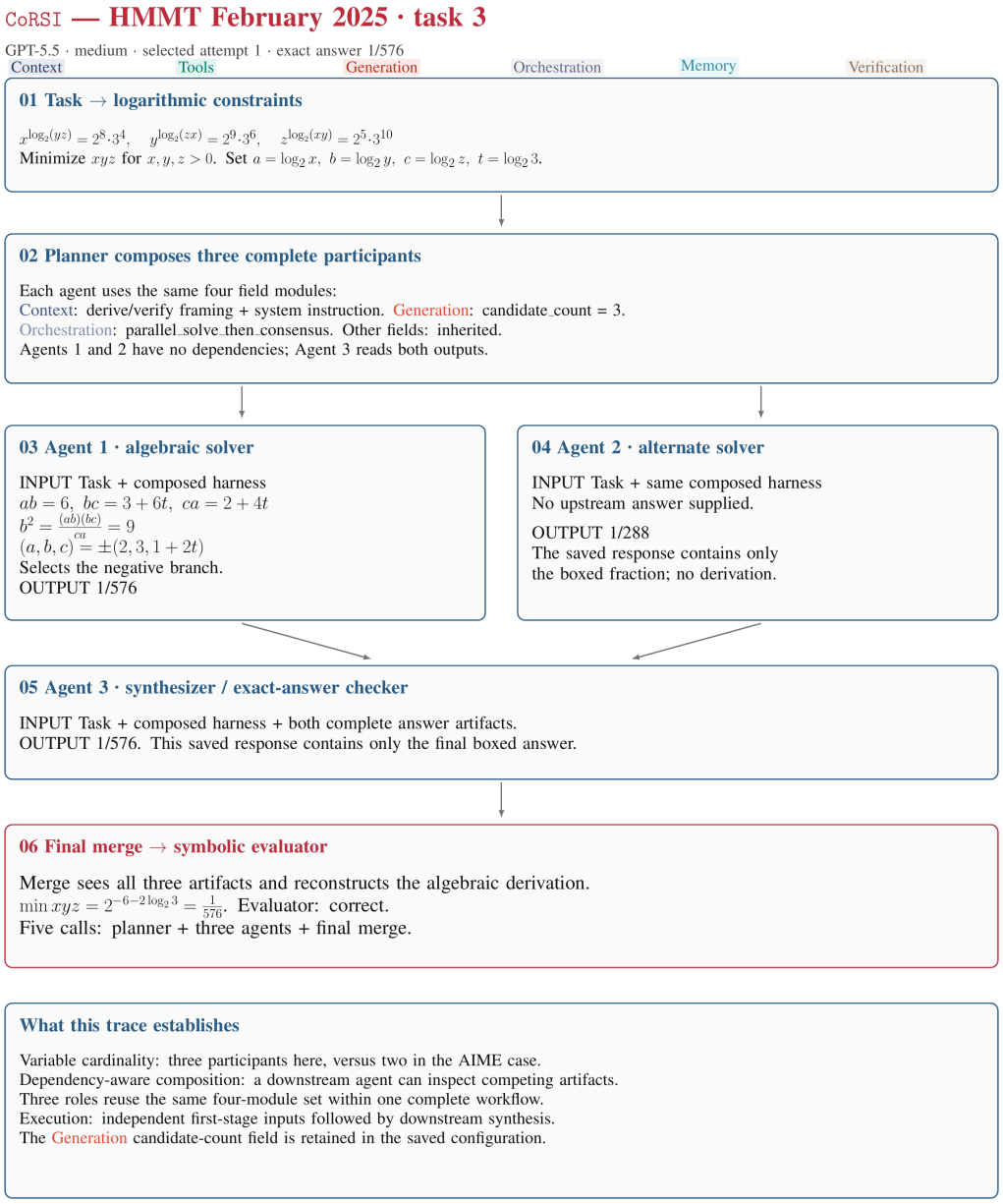


Figure 16: Case workflow / HMMT task 3. Task, Planner composition, independent participant artifacts, downstream synthesis, and exact-answer verification share one visual sequence.

E.1 A COMMON LANGUAGE FOR THE PROGRESSION

The development of self-improving agents can be organized by three questions: what changes, which evidence selects the change, and whether the update operator is itself mutable. A recent survey formalizes agents as coupled model-and-scaffold configurations and organize self-improvement by update targets and driving signals (Ren et al., 2026). This perspective connects answer refinement, experience retention, workflow search, and meta-level adaptation. The stages below are overlapping research directions: a contemporary system can combine several of them within one agent.

E.2 FEEDBACK IMPROVES THE CURRENT SOLUTION

Self-Refine uses a generator, feedback, and refinement loop to revise an initial response (Madaan et al., 2023). Reflexion converts task feedback into verbal reflections that condition subsequent attempts (Shinn et al., 2023). ReAct provides the reasoning–action interface through which agents observe environment feedback (Yao et al., 2023). These systems make feedback actionable at the answer or trajectory level. Their reusable objects include critiques and reflections; the next step changes the solution by conditioning generation on that evidence. For `CoRSI`, feedback additionally indexes the configuration that produced the solution, making successful control decisions retrievable.

E.3 EXPERIENCE BECOMES A PERSISTENT SKILL OR MEMORY

Voyager retains executable skills in a library and retrieves them for new tasks (Wang et al., 2024). ExpeL extracts transferable insights from experience and uses them to guide subsequent reasoning (Zhao et al., 2023). MemoHarness makes this experience-to-harness pathway explicit by storing diagnoses and reusable control patterns across executions (Huang et al., 2026); Meta-Harness optimizes the complete harness as an end-to-end object (Lee et al., 2026). Recuris develops a coupled working-memory and experiential-memory loop with structured execution evidence and validation-gated updates (Yu et al., 2026). This line moves the reusable unit beyond a single answer: experience persists across task boundaries and influences future behavior. `CoRSI` stores typed configuration patches with source evidence, allowing a later participant to combine compatible pieces from different histories while inheriting its other fields from the base harness.

E.4 PROGRAMS AND WORKFLOWS BECOME OPTIMIZATION TARGETS

DSPy compiles declarative language-model programs using modular signatures and optimization (Khattab et al., 2024). GPTSwarm represents agents as optimizable graphs, exposing both nodes and connections (Zhuge et al., 2024). AFlow searches executable agentic workflows, while EvoFlow evolves diverse workflows and adapts their use to incoming tasks (Zhang et al., 2024; 2025a). AutoFlow and ADAS use automated generators to search agent programs and workflows (Li et al., 2024; Hu et al., 2024). ADAS uses a meta-agent to search agent programs and retain discovered designs (Hu et al., 2024). Together these works expand the search surface from prompts to execution structure. Dynamic task-graph systems additionally study asynchronous routing and parallel multi-agent execution (Yu et al., 2025; Yun et al., 2026; Feng et al., 2026a). `CoRSI` couples structure to participant-local configuration: the same typed bank can instantiate different module subsets, team sizes, and artifact dependencies for different tasks.

E.5 THE UPDATE OPERATOR ENTERS THE IMPROVEMENT SPACE

Promptbreeder evolves task prompts and mutation prompts, making the mechanism that proposes changes part of the search (Fernando et al., 2023). STOP recursively applies a code-improvement program to improve that program itself (Zelikman et al., 2023). The Darwin Godel Machine explores and evaluates self-modified coding agents in an archive of agent variants (Zhang et al., 2025b). AlphaEvolve applies evolutionary search to executable code proposals under a task-specific evaluator (Novikov et al., 2025). Hyperagents integrate task and meta agents in an editable program (Zhang et al., 2026). `CoRSI`'s reported implementation uses an experience bank, typed module catalog, and cooperation history to update the solver state while keeping the Planner contract fixed: successful compositions become reusable evidence and modules for subsequent workflow proposals.

E.6 COMPOSITION MAKES COMPONENT VALUE CONTEXTUAL

Cooperative coevolution evaluates a component through the full solution formed with representatives of other components (Potter & De Jong, 1994). The relevant value is therefore conditional on collaborators. Applied to agent harnesses, a Verification patch can increase the utility of a Generation patch, and a Tools policy can change which Context instructions are useful. `CoRSI` uses typed assembly to preserve compatible discoveries and cooperation history to expose successful partnerships. The second difference in Eq. (24) formalizes interaction, while the recorded co-success matrix provides the observed collaboration history. This connects the reusable unit, the complete evaluation object, and the evidence available to the next search step.

F DISCUSSION

This paper develops compositional adaptation at the harness level. Future work could place this mechanism inside an open-ended learning ecosystem in which model parameters, harness programs, task distributions, and

evaluators co-evolve, creating a joint theory of self-improvement across adaptation layers. Realizing that vision would require new benchmarks and experimental infrastructure.

G AI ASSISTANCE

In this work, generative AI tools were used to aid and polish the writing, including language editing, organization, and consistency checks. They were not used to generate data, formulate scientific claims, design experiments, run evaluations, or interpret the scientific results. The authors reviewed and finalized all AI-assisted text and remain responsible for the complete manuscript, including its claims, analyses, and artifacts.