RECURSIVE ENVIRONMENT SCALING FOR GENERAL LONG-HORIZON AGENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

We study environment scaling as a way to increase agent-benchmark difficulty while holding core task objectives fixed. Our framework, Recursive Environment Scaling (RES), transforms existing benchmark environments while preserving their objectives and completion requirements. A proposer changes information access, resource relations, and execution dependencies; execution-grounded checks validate the resulting candidate, and accepted changes are inherited across rounds. This produces related environments for paired evaluation and training-trajectory collection. Experiments on 294 initial tasks from three benchmarks show lower task success for all six evaluation models, with decreases of 57.14–77.14 percentage points on Claw-Eval-Live. With equal trajectory counts and identical optimization settings, fine-tuning on RES trajectories outperforms original-trajectory SFT in all nine model–benchmark comparisons by 7.00–14.61 percentage points. Ablations identify recursive inheritance and dependency composition as important contributors to increased difficulty. Together, these results support environment scaling for both agent evaluation and training.

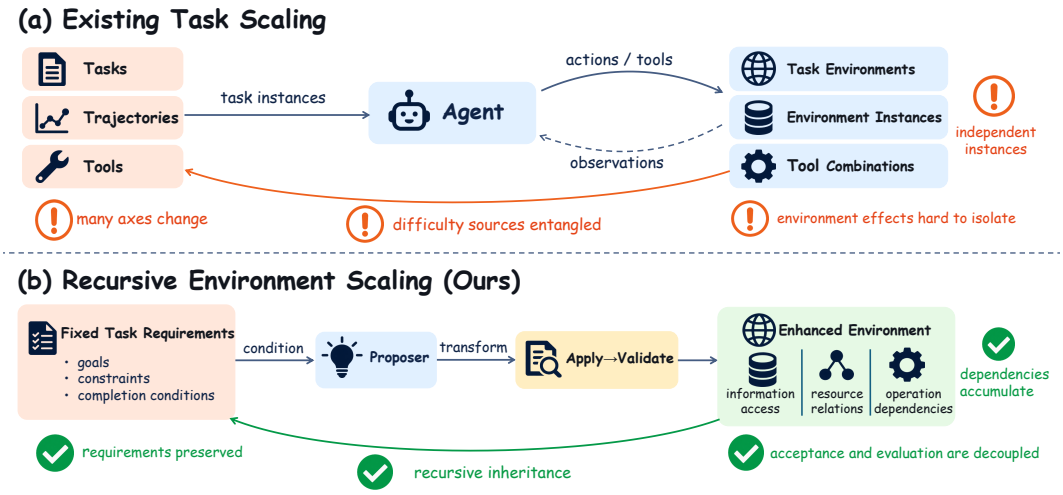

Figure 1: **Task scaling and recursive environment scaling.** Task scaling expands tasks and trajectories. RES preserves core objectives while recursively transforming information access, resource relations, and execution dependencies. Accepted changes accumulate across rounds.

1 INTRODUCTION

Language model agents increasingly perform tasks through extended interaction with digital environments. AgentBench (Liu et al., 2024) examines decisions across diverse environments, OS-World (Xie et al., 2024) tests real computer use, and TheAgentCompany (Xu et al., 2025) evaluates workplace workflows. Across these settings, difficulty depends on how information is organized,

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

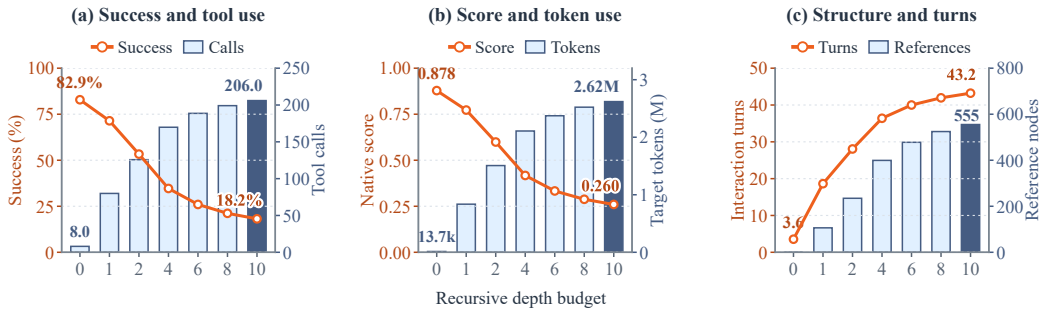

Figure 2: **Environment scaling increases execution demand.** Historical means for 105 Claw-Eval-Live tasks using DeepSeek-V4.1-Flash (one attempt, 50 turns, success threshold 0.80). Budget d averages retained states $0, \ldots, d$, carrying forward early-stopped finals (Appendix C). Tokens include target input and output; reference nodes count added structure, not shortest solution paths.

resources are connected, and earlier actions constrain later ones. This makes the environment itself a useful object of study when scaling agent evaluation.

Recent work expands agent experience through task synthesis and environment construction. TaskCraft (Shi et al., 2025) scales the depth and breadth of tool-use tasks; LOGIGEN (Zeng et al., 2026) grounds task construction in executable policies and state verification. RST (Li et al., 2026b) recursively extends reference solutions and realigns instructions and verifiers with the resulting workflows. At the environment level, EnvScaler (Song et al., 2026) synthesizes tool-interaction sandboxes, and ScaleEnv (Tu et al., 2026) constructs interactive environments and verifiable tasks from scratch. RES transforms existing benchmark environments under fixed task objectives and completion requirements, enabling paired evaluation across descendants.

WorkArena++ (Boisvert et al., 2024) composes multi-step knowledge-work tasks, while τ^2-Bench (Barres et al., 2025) couples agent and user actions through shared state. These settings highlight how dependencies shape task execution. Changing access paths and prerequisites can lengthen a task's workflow while preserving the information and operations needed for completion. This motivates our research question: How can we recursively scale executable environments while preserving core task objectives?

We introduce **Recursive Environment Scaling (RES)** (Figure 1). A proposer inspects a validated parent environment and composes programmatic transformations of its resources and execution dependencies. Execution-grounded checks determine whether the candidate realizes the proposed changes and preserves task validity. Accepted children are inherited for subsequent construction, allowing later transformations to build on earlier structures. Target-agent evaluation measures difficulty separately from environment acceptance. The lineages support paired evaluation and training-trajectory collection. Figure 2 summarizes performance, interaction demand, and environment structure on Claw-Eval-Live.

Across six models and three benchmarks, RES reduces task success in all 18 comparisons. On Claw-Eval-Live, DeepSeek-V4.1-Flash falls from 82.86% to 5.71%. With equal-sized training sets, RES trajectories also improve all three Qwen3.5 Base models over original-trajectory SFT on every benchmark. Our contributions are threefold:

1. **Environment Scaling.** We study existing benchmark environments as a scaling dimension, preserving core task objectives while transforming information access and execution dependencies.
2. **RES Framework.** We combine proposer-guided transformations, execution-grounded validation, and recursive inheritance to accumulate accepted changes along each task lineage.
3. **Empirical Evidence.** RES reduces task success across six models and improves SFT across three backbones; ablations assess inheritance and dependencies.

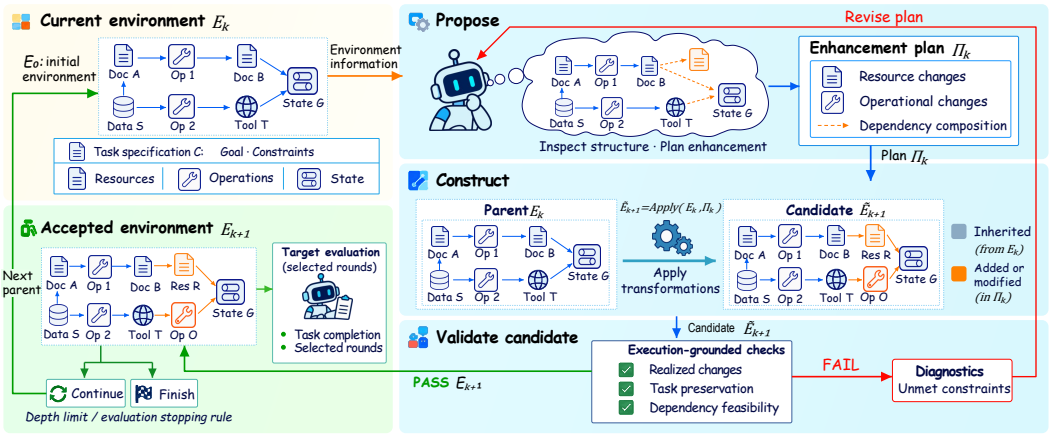

Figure 3: **Overview of RES.** The proposer transforms accepted parent E_k. Execution-grounded checks validate the changes, task validity, and dependency feasibility, guiding revisions before accepting a child. Accepted children become parents; target evaluation separately informs stopping.

2 Related Work

Evaluating agents in executable environments. SWE-bench (Jimenez et al., 2024) and Terminal-Bench (Merrill et al., 2026) cover repository and command-line tasks. WebArena (Zhou et al., 2024) and WebVoyager (He et al., 2024) evaluate website interaction, with BrowserGym (de Chezelles et al., 2025) providing shared interfaces. WorkArena (Drouin et al., 2024) studies enterprise workflows; τ-Bench (Yao et al., 2024) couples tool use with domain policies and final-state checks. AppWorld (Trivedi et al., 2024) spans applications, while Claw-Eval-Live (Li et al., 2026a) and Workspace-Bench (Tang et al., 2026) emphasize execution evidence and file dependencies.

Task and trajectory construction. Mind2Web (Deng et al., 2023) collects human action sequences across websites. TaskCraft (Shi et al., 2025) expands task depth and breadth, while LOGIGEN (Zeng et al., 2026) grounds construction in executable policies and state verification. DIVE (Chen et al., 2026) uses executed tool traces, and CLI-Universe (Hua et al., 2026) grounds terminal tasks in technical material. RST (Li et al., 2026b) extends reference solutions recursively and realigns instructions and verifiers. RES keeps core task requirements fixed while transforming the environment.

Environment construction and adaptation. SWE-Gym (Pan et al., 2025) packages tasks with runtimes and tests, and AgentGym (Xi et al., 2025) supports exploration and training across environments. EnvScaler (Song et al., 2026) synthesizes tool-interaction sandboxes; ScaleEnv (Tu et al., 2026) builds tools, dependency graphs, and verifiable tasks from scratch. Agent-World (Dong et al., 2026) combines environment discovery with a training arena, and Gym-Anything (Aggarwal et al., 2026) configures real software for computer use. SPICE (Liu et al., 2025) grounds self-play in a document corpus, while SPADE (Liu et al., 2026) trains an environment designer jointly with a learner. RES recursively inherits accepted changes to existing benchmarks, supporting paired difficulty evaluation and training-trajectory collection.

3 Method

3.1 Problem Formulation and Overview

We represent a task at recursion step k by (C_k, E_k), where C_k specifies its objective, constraints, and completion conditions, and E_k contains the executable resources, states, operations, and interaction dependencies. RES fixes the core objective, underlying facts, and completion requirements; specification updates describe the access instructions needed after environmental changes. A validated

seed starts a recursive environment lineage:

$$(C_0, E_0) \to (C_1, E_1) \to \cdots \to (C_K, E_K). \tag{1}$$

The proposer composes a transformation plan Π_k for the accepted parent. Executing this plan produces a candidate,

$$\widetilde{E}_{k+1} = \mathrm{Apply}(E_k, \Pi_k), \tag{2}$$

which becomes the next parent only after validation (Section 3.3). Figure 3 illustrates the procedure. Public instructions describe revised access paths and prerequisites, allowing the agent to recover the information needed for the original required outputs.

3.2 Environment Transformation

Planning and execution. The proposer conditions its plan on the task specification and current environment, selecting programmatic primitives, resource bindings, parameters, and their composition. Resource transformations change information access; relational transformations alter relationships among sources; operational transformations modify when actions become available. The plan binds transformations to concrete resources and orders their execution. Applying it produces the interfaces and state transitions that the target encounters (Equation 2). Benchmark-specific resource bindings and the transformation library are described in Appendices A and D.1.

Dependency composition. RES connects transformations so that one operation can consume information or state produced by another. Dependencies may combine several prerequisites before enabling an action or propagate state changes to later operations. Subsequent generations can extend these structures, adding interaction requirements while retaining a valid solution path. For example, an evidence view may require a link query and export preparation. A later transformation can add index recovery before that view. The inherited procedure becomes part of a longer execution path around the same objective, as the reconciliation case in Section 5.4 illustrates.

3.3 Validated Environment Construction

Task preservation and validation. Acceptance checks task preservation, construction consistency, and public executability. The verifier checks the parent lineage and protected task requirements, then independently replays the declared transformations from the fixed parent. The rebuilt executable definition must match the submitted candidate. We write the validation decision as

$$v_k = V(C_{k+1}, E_k, \Pi_k, \widetilde{E}_{k+1}; \xi_k), \tag{3}$$

where ξ_k denotes replay and execution evidence. In a fresh session, the verifier follows public interfaces to resolve references, execute preparation steps, and reconstruct transformed resources. Recovered content is compared with source facts; missing records, invalid references, and unsatisfied prerequisites reject the candidate. These checks cover the transformed access paths while preserving the original completion requirements (Appendix A).

Repair. Failed checks return diagnostics about unsatisfied requirements, construction mismatches, or invalid dependencies. The proposer revises the affected resource bindings, parameters, or operation sequence and resubmits the candidate. Repair changes the proposed child while preserving the accepted parent. Candidates still invalid at the repair limit are discarded, leaving that parent available for subsequent construction.

3.4 Recursive Environment Scaling

The validated child becomes the next parent, preserving its resources and dependencies. Later plans reuse existing access paths and attach prerequisites to inherited operations, so accepted changes accumulate along the lineage.

We evaluate the original and last accepted environments, plus intermediate generations requested by the proposer, under the protocol in Appendix A. At measured nodes, three assessed failures with no measurement errors stop further descendants. Transformation planning and acceptance use the environment and validation evidence. Recursion ends at a budget or stopping condition, with the last accepted snapshot retained as the scaled environment.

Table 1: **Environment scaling lowers task completion across six models and three benchmarks.** Original denotes the released benchmark environment; +RES denotes the last accepted environment after scaling. Each model makes one 50-turn attempt per task in both conditions. All metrics use a 0–100 scale. Claw Pass@1 requires native score ≥ 0.80, and Score is the mean native score. Workspace TCR@80 requires at least 80% of a task's rubric items to pass; RPR is the percentage of passed rubric items pooled across tasks. Terminal Pass@1 uses binary verifier outcomes.

		Claw-Eval-Live $N = 105$		**Workspace-Bench-Lite** $N = 100$		**Terminal-Bench 2.1** $N = 89$
Model	Environment	Pass@1	Score	TCR@80	RPR	Pass@1
DeepSeek-V4.1-Flash	Original	82.86	87.71	63.00	71.19	91.01
	+RES	5.71	14.20	46.00	60.38	43.82
GPT-5.6 Sol	Original	85.71	91.24	61.00	70.60	88.76
	+RES	26.67	35.44	49.00	65.08	44.94
Claude Opus 5	Original	86.67	92.13	70.00	74.98	89.89
	+RES	28.57	37.28	51.00	65.57	47.19
GLM-5.2	Original	80.95	88.64	59.00	69.30	80.90
	+RES	23.81	39.80	45.00	61.19	39.33
Qwen3.8-Max	Original	82.86	90.82	67.00	72.81	86.52
	+RES	22.86	33.23	49.00	63.08	41.57
Kimi K3	Original	84.76	90.16	65.00	72.38	88.76
	+RES	13.33	23.85	48.00	62.70	42.70

4 Experiments

We test whether RES makes existing benchmarks more demanding across six evaluation models and improves training across three Qwen3.5 Base backbones.

4.1 Experimental Setup

Benchmarks and models. We use Claw-Eval-Live (Li et al., 2026a), Workspace-Bench-Lite (Tang et al., 2026), and Terminal-Bench 2.1 (Merrill et al., 2026), with 105, 100, and 89 tasks, respectively. Each task defines a paired comparison between its original environment and the last environment accepted by RES. We evaluate DeepSeek-V4.1-Flash, GPT-5.6 Sol, Claude Opus 5, GLM-5.2, Qwen3.8-Max, and Kimi K3 on both versions. Task success measures whether an agent satisfies the benchmark's completion criterion; native scores and rubric pass rates additionally measure partial completion. The table captions define each metric.

Scaling and evaluation protocol. GPT-5.6 Sol proposes environment transformations, and DeepSeek-V4.1-Flash serves as the scaling-stage target; both use high reasoning effort. Recursion proceeds to at most ten accepted transformations. Original and last accepted environments are always evaluated; the proposer selects any intermediate evaluations. Three 60-turn target attempts inform stopping during scaling. We then freeze the environments and evaluate each model with one 50-turn attempt per task and high reasoning effort, holding tool permissions and scoring fixed within each pair. Appendix A gives additional execution settings and validity checks.

Training comparison. We fine-tune Qwen3.5-4B-Base, Qwen3.5-9B-Base, and Qwen3.5-35B-A3B-Base under two conditions. *+SFT w/o RES* uses original ClawGym trajectories; *+SFT w/ RES* uses only trajectories collected by GPT-5.6 Sol in RES-scaled environments. Both start from the same Base checkpoint and use equal numbers of trajectories with identical optimization settings. All checkpoints are evaluated on the original tasks of the three benchmarks with the same single-attempt protocol. Data selection and hyperparameters appear in Appendix B.

Table 2: **SFT with RES trajectories improves performance on the original benchmarks.** For each Qwen3.5 Base model, +SFT w/o RES uses original-environment trajectories; +SFT w/ RES uses only trajectories collected in RES-scaled environments. Both use the same initialization, sample count, and optimization. Each checkpoint makes one 50-turn attempt per task. Metrics use a 0–100 scale: Claw Pass@1 thresholds native scores at 0.80, Score averages native scores, Workspace TCR@80 thresholds task rubric completion at 80%, RPR is the pooled percentage of passed rubric items, and Terminal Pass@1 uses its binary verifier. Bold marks the best result per model.

	Qwen3.5-4B			**Qwen3.5-9B**			**Qwen3.5-35B-A3B**		
Metric	Base	+SFT w/o RES	+SFT w/ RES	Base	+SFT w/o RES	+SFT w/ RES	Base	+SFT w/o RES	+SFT w/ RES
Claw-Eval-Live ($N = 105$)									
Pass@1	49.52	52.38	**61.90**	60.00	62.86	**71.43**	61.90	64.76	**77.14**
Score	71.51	74.12	**79.15**	77.75	80.32	**81.04**	78.96	80.36	**82.83**
Workspace-Bench-Lite ($N = 100$)									
TCR@80	1.00	4.00	**11.00**	9.00	12.00	**21.00**	11.00	14.00	**28.00**
RPR	7.46	26.59	**33.68**	30.97	35.89	**42.22**	34.70	37.24	**45.62**
Terminal-Bench 2.1 ($N = 89$)									
Pass@1	2.25	5.62	**13.48**	10.11	14.61	**24.72**	13.48	23.60	**38.20**

4.2 Scaling Benchmark Difficulty

Environment scaling reduces task success across models. Table 1 shows lower success in all 18 model–benchmark comparisons. On Claw-Eval-Live, DeepSeek-V4.1-Flash drops from 82.86% to 5.71%, while Claude Opus 5 drops from 86.67% to 28.57%. The reduction spans 12.00–19.00 percentage points on Workspace-Bench-Lite and 41.57–47.19 on Terminal-Bench 2.1. Under the same single-attempt evaluation budget, the scaled environments are consistently harder across the model suite.

Requirement completion also declines. Every model obtains a lower Claw native score and Workspace RPR after scaling. For GPT-5.6 Sol, Claw native score falls from 91.24 to 35.44; Workspace RPR reductions range from 5.52 to 10.81 points across the model suite. Agents therefore satisfy fewer underlying requirements as well as complete fewer tasks. Section 5.2 examines how this decline develops with recursive depth and increased interaction demand.

4.3 Training on Scaled Environments

RES trajectories improve SFT across all three backbones. Training with RES trajectories outperforms original-trajectory SFT in all nine model–benchmark comparisons (Table 2). Averaged over the three backbones, the gains are 10.16, 10.00, and 10.86 percentage points on Claw-Eval-Live, Workspace-Bench-Lite, and Terminal-Bench 2.1. The largest gains occur for 35B-A3B: 12.38, 14.00, and 14.61 points, respectively.

Training gains extend to partial completion. Claw native score and Workspace RPR improve for every backbone. For 4B, Workspace RPR rises from 26.59 to 33.68 while TCR@80 rises from 4.00 to 11.00. The gains therefore span both rubric-level completion and the number of tasks meeting the success threshold.

5 Ablations and Analysis

5.1 Ablation Studies

We examine dependency composition, recursive inheritance, depth, and library breadth on Claw-Eval-Live (Table 3). DeepSeek-V4.1-Flash makes one attempt per environment with a 50-turn limit and success threshold 0.80. Component comparisons share 15 tasks; depth analysis uses 105 frozen lineages. Protocols and per-task results appear in Appendices D and D.2.

Table 3: **Effects of RES components and recursion depth on Claw-Eval-Live.** DeepSeek-V4.1-Flash makes one attempt per environment with a 50-turn limit. Pass@1 uses native score ≥ 0.80; Score is mean native score on a 0–100 scale. Lower values indicate harder environments. A removes composed dependencies and their prerequisite work from saved environments. B uses the same 15 tasks with matched planned depths and proposer budgets: w/o inheritance evaluates the last independent enhancement from the original environment, and Basic retains six operators. Checkmarks mark enabled components; blue shading identifies full RES controls and the default depth cap of ten. C reevaluates 105 separate lineages, retaining the last accepted environment after early stopping.

Variant	Dependency composition	Inheritance	Library	Pass@1 (%) ↓	Score ↓
A. Remove dependencies from saved environments (15 tasks)					
Saved RES	✓	✓	Full	**0.00**	**7.73**
w/o dependencies	×	✓	Full	100.00	92.93
B. Rebuild environments under matched budgets (15 tasks)					
Rebuilt RES	✓	✓	Full	**6.67**	**14.40**
w/o inheritance	✓	×	Full	93.33	88.73
w/ basic library	✓	✓	Basic	80.00	85.73
C. Increase the maximum recursion depth (105 tasks)					
Original benchmark (depth 0)				82.86	87.84
Maximum depth 1				60.00	66.69
Maximum depth 2				17.14	25.17
Maximum depth 3				7.62	15.30
Maximum depth 5				3.81	11.84
Maximum depth 10				4.76	13.37

Dependency composition increases difficulty. Removing cross-operator references and preparation prerequisites from saved RES environments, while retaining task facts and atomic transformation histories, raises pass@1 from 0.0% to 100.0% and mean native score from 7.73 to 92.93. Mean turns fall from 50.0 to 19.1, linking the removed prerequisites to both completion and interaction cost.

Recursive inheritance accumulates enhancement. For planned depth d_i, we compare recursive construction with d_i independent enhancements of the original environment, evaluating the predeclared last construction. Both use the same nominal proposer-call limit per enhancement. The recursive control yields 6.7% pass@1 versus 93.3% independently, with scores of 14.40 versus 88.73 and approximately 2.5 times the mean target-token expenditure. Inheriting earlier changes therefore produces harder environments under matched construction opportunities.

Additional depth strengthens enhancement. Applying depth caps to the same 105 lineages, with early-stopped finals retained, reduces pass@1 from 82.9% at E_0 to 60.0% after one round, 17.1% at depth two, and 4.8% at depth ten. Mean native score falls from 87.84 to 13.37. These retained-environment measurements show gains beyond one round; Appendix C.3 reports all caps and distinguishes them from historical averages.

A richer library enables stronger enhancement. Restricting transformations to a Basic subset of six mechanisms, with recursion and planned depths unchanged, raises pass@1 from 6.7% to 80.0% and native score from 14.40 to 85.73 relative to the shared regenerated control. The full library gives the proposer more effective ways to compose environmental requirements.

5.2 Cumulative Effects of Recursive Environment Scaling

We fix DeepSeek-V4.1-Flash and a 50-turn budget per attempt. Figure 4 reports historical means from E_0 through each depth budget over the same 105 tasks, retaining final environments for early-stopped chains (Appendix C).

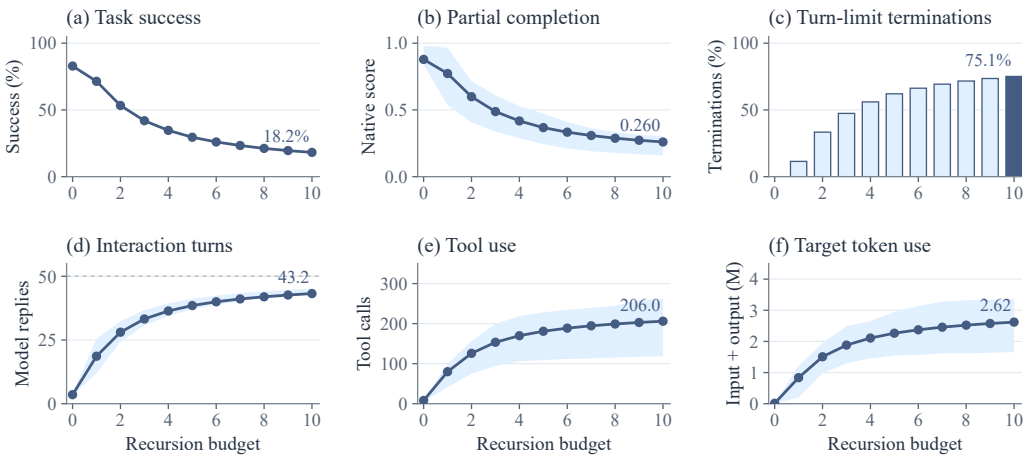

Figure 4: **Historical performance and interaction demand on Claw-Eval-Live.** Lines and bars show 105-seed historical means (Equation 5); shading shows task interquartile ranges. The dashed line marks the 50-turn limit. Tokens include target input and output. Termination rates use logged outcomes; actual depth counts appear in Appendix C.2.

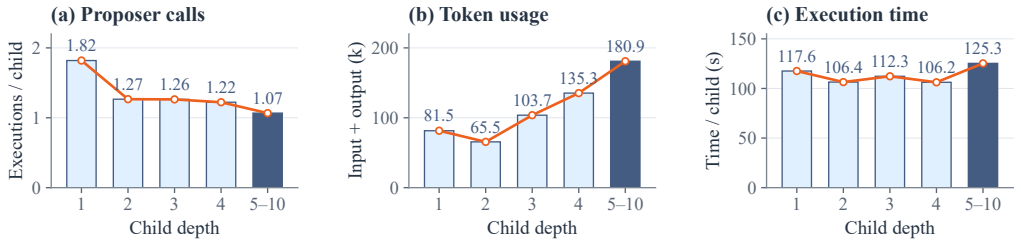

Figure 5: **Proposer cost across recursive stages.** Bars and lines show the same per-child means on Claw-Eval-Live. Depths 1–4 contain 105, 94, 57, and 18 children; dark bars pool 15 children from five seeds at depths 5–10. Time sums proposer execution durations.

Success and partial completion decline together. From budget zero to ten, historical mean success falls from 82.86% to 18.18%, and mean native score from 0.878 to 0.260. The two measures show a decline in both task success and partial completion under this benchmark-level aggregation.

Interaction demand increases. Under the same aggregation, mean turns rise from 3.55 to 43.22 and tool calls from 8.05 to 206.02; the turn-limit rate reaches 75.06%. These measurements include successful and unsuccessful trajectories. Depth and endpoint evaluations use separate batches; Appendix C reports the retained-state results and depth counts.

5.3 Environment Scaling Efficiency and Cost

On Claw-Eval-Live, RES produces 289 accepted children from 105 seeds. Of these, 92.0% require at most two outer proposer executions; mean costs are 1.45 executions, 89.2k tokens, and 112.6 seconds per child. Figure 5 shows fewer outer executions but greater token use at later stages, so execution counts alone do not capture scaling cost. Adaptive stopping also changes which tasks reach later stages.

Costs cover executions associated with accepted children, excluding unsuccessful branches, target solving, and grading. An outer execution may contain multiple model requests and tool calls; time sums execution durations. Full definitions and distributions appear in Appendix A.

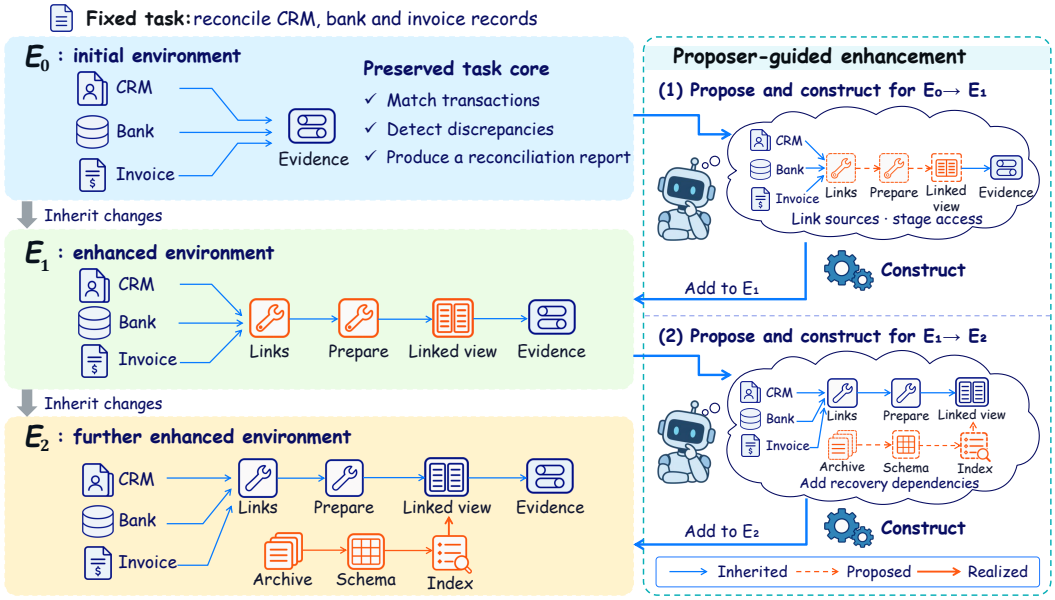

Figure 6: **Recursive scaling of a financial reconciliation task.** Access and recovery dependencies accumulate while the reconciliation objective is preserved. The diagram shows E_0–E_2; endpoint evaluation uses final E_3.

5.4 Case Study: Three-Source Financial Reconciliation

Task A01 in Claw-Eval-Live requires matching CRM, bank, and invoice records, identifying discrepancies, and producing a reconciliation report. In E_0, the agent can read three CSV files directly. E_1 introduces link queries and export preparation before a linked view can be accessed. E_2 inherits these steps and adds archive, schema, and index-recovery dependencies (Figure 6); final E_3 adds encoding constraints. Evidence-access requirements accumulate around the preserved core reconciliation objective.

In the independent paired evaluation, DeepSeek-V4.1-Flash makes 171 tool calls in the scaled environment, compared with three in the original environment, while its native score falls from 0.98 to zero. The trajectory includes reference queries, export preparation, and index rebuilding, with one batched query covering 37 references. The agent reaches the 50-turn limit while still acquiring evidence and does not complete the reconciliation report. The added work is concentrated before reconciliation: the original three reads expose the needed records directly, whereas the scaled environment requires the agent to resolve inherited access and recovery dependencies. This case links the observed increase in interaction demand to the operations introduced along the lineage.

Validation and target evaluation provide complementary evidence in this example. Restoration checks verify that the required records remain recoverable through public operations. The rollout then measures whether the agent can complete the original report within its budget. Its failure during evidence acquisition connects the lower score to the added access procedure. The objective and required deliverable stay fixed throughout this comparison.

6 Conclusion

RES makes existing executable environments a controllable dimension of agent evaluation and training. Under preserved core task objectives, recursively inherited transformations reduce task completion across three benchmarks and six models, while the resulting training trajectories improve all three Qwen3.5 Base models over original-trajectory SFT. Component ablations and trajectory analysis connect these observations to inherited dependencies and increased interaction demand. These results support environment scaling as a practical way to extend existing benchmarks and collect training trajectories that improve agent performance.

AI USE STATEMENT

Generative AI tools were used to assist with migrating the manuscript's LaTeX source to the ICLR 2027 template, improving language and presentation, checking internal consistency, and verifying bibliographic metadata and the resulting build. The authors reviewed all suggested changes, made the scientific and editorial decisions, and take responsibility for the final content.

REFERENCES

Pranjal Aggarwal, Graham Neubig, and Sean Welleck. Gym-Anything: Turn any Software into an Agent Environment. *arXiv: 2604.06126*, 2026.

Victor Barres, Honghua Dong, Soham Ray, Xujie Si, and Karthik Narasimhan. τ^2-Bench: Evaluating Conversational Agents in a Dual-Control Environment. *arXiv: 2506.07982*, 2025.

Léo Boisvert, Megh Thakkar, Maxime Gasse, Massimo Caccia, Thibault Le Sellier De Chezelles, Quentin Cappart, Nicolas Chapados, Alexandre Lacoste, and Alexandre Drouin. WorkArena++: Towards Compositional Planning and Reasoning-based Common Knowledge Work Tasks. In *Advances in Neural Information Processing Systems*, 2024.

Aili Chen, Chi Zhang, Junteng Liu, Jiangjie Chen, Chengyu Du, Yunji Li, Ming Zhong, Qin Wang, Zhengmao Zhu, Jiayuan Song, et al. DIVE: Scaling Diversity in Agentic Task Synthesis for Generalizable Tool Use. *arXiv: 2603.11076*, 2026.

Thibault Le Sellier de Chezelles, Maxime Gasse, Alexandre Lacoste, Massimo Caccia, Alexandre Drouin, Léo Boisvert, Megh Thakkar, Tom Marty, Rim Assouel, Sahar Omidi Shayegan, Lawrence Keunho Jang, Xing Han Lù, Ori Yoran, Dehan Kong, Frank F. Xu, Siva Reddy, Graham Neubig, Quentin Cappart, Russ Salakhutdinov, and Nicolas Chapados. The BrowserGym Ecosystem for Web Agent Research. *Transactions on Machine Learning Research*, 2025.

Xiang Deng, Yu Gu, Boyuan Zheng, Shijie Chen, Sam Stevens, Boshi Wang, Huan Sun, and Yu Su. Mind2Web: Towards a Generalist Agent for the Web. In *Advances in Neural Information Processing Systems*, 2023.

Guanting Dong, Junting Lu, Junjie Huang, Wanjun Zhong, Longxiang Liu, Shijue Huang, Zhenyu Li, Yang Zhao, Xiaoshuai Song, Xiaoxi Li, et al. Agent-World: Scaling Real-World Environment Synthesis for Evolving General Agent Intelligence. *arXiv: 2604.18292*, 2026.

Alexandre Drouin, Maxime Gasse, Massimo Caccia, Issam H. Laradji, Manuel Del Verme, Tom Marty, David Vazquez, Nicolas Chapados, and Alexandre Lacoste. WorkArena: How Capable are Web Agents at Solving Common Knowledge Work Tasks? In *Proceedings of the 41st International Conference on Machine Learning*, pp. 11642–11662, 2024.

Hongliang He, Wenlin Yao, Kaixin Ma, Wenhao Yu, Yong Dai, Hongming Zhang, Zhenzhong Lan, and Dong Yu. WebVoyager: Building an End-to-End Web Agent with Large Multimodal Models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2024.

Zhanbo Hua, Yifan Yao, Weihao Xie, Yongchi Zhao, Minghao Liu, Ruizhi Qiu, Zhewei Huang, Zun Wang, Yiyan Ji, Yunhai Ye, et al. CLI-Universe: Towards Verifiable Task Synthesis Engine for Terminal Agents. *arXiv: 2606.22883*, 2026.

Carlos E Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik Narasimhan. SWE-bench: Can Language Models Resolve Real-world Github Issues? In *International Conference on Learning Representations*, 2024.

Chenxin Li, Zhengyang Tang, Mingxin Huang, Yunlong Lin, Shijue Huang, Shengyuan Liu, Bowen Ye, Rang Li, Lei Li, Benyou Wang, and Yixuan Yuan. Claw-Eval-Live: A Live Agent Benchmark for Evolving Real-World Workflows. *arXiv: 2604.28139*, 2026a.

Zhongzhi Li, Yucheng Shi, Zongxia Li, Ruhan Wang, Anhao Li, Zixun Huang, Junyao Yang, Lei Ke, Ninghao Liu, Haitao Mi, and Leowei Liang. Recursive Synthesis for Long-Horizon Terminal Tasks. *arXiv: 2608.05466*, 2026b.

Bo Liu, Chuanyang Jin, Seungone Kim, Weizhe Yuan, Wenting Zhao, Ilia Kulikov, Xian Li, Sainbayar Sukhbaatar, Jack Lanchantin, and Jason Weston. SPICE: Self-Play In Corpus Environments Improves Reasoning. *arXiv: 2510.24684*, 2025.

Bo Liu, Simon Yu, Yiding Jiang, Ao Qu, Andrew Zhao, Zichen Liu, Junsu Kim, Zijian Zhou, Seungone Kim, Tongzheng Ren, et al. SPADE: Self-Play in Adaptive Synthetic Executable Environments. *arXiv: 2608.19197*, 2026.

Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hangliang Ding, Kaiwen Men, Kejuan Yang, Shudan Zhang, Xiang Deng, Aohan Zeng, Zhengxiao Du, Chenhui Zhang, Sheng Shen, Tianjun Zhang, Yu Su, Huan Sun, Minlie Huang, Yuxiao Dong, and Jie Tang. AgentBench: Evaluating LLMs as Agents. In *International Conference on Learning Representations*, 2024.

Mike A. Merrill, Alexander G. Shaw, Nicholas Carlini, Boxuan Li, Harsh Raj, Ivan Bercovich, Lin Shi, Jeong Yeon Shin, Thomas Walshe, E. Kelly Buchanan, et al. Terminal-Bench: Benchmarking Agents on Hard, Realistic Tasks in Command Line Interfaces. *arXiv: 2601.11868*, 2026.

Jiayi Pan, Xingyao Wang, Graham Neubig, Navdeep Jaitly, Heng Ji, Alane Suhr, and Yizhe Zhang. Training Software Engineering Agents and Verifiers with SWE-Gym. In *Proceedings of the 42nd International Conference on Machine Learning*, pp. 47717–47737, 2025.

Dingfeng Shi, Jingyi Cao, Qianben Chen, Weichen Sun, Weizhen Li, Hongxuan Lu, Fangchen Dong, Tianrui Qin, King Zhu, Minghao Liu, et al. TaskCraft: Automated Generation of Agentic Tasks. *arXiv: 2506.10055*, 2025.

Xiaoshuai Song, Haofei Chang, Guanting Dong, Yutao Zhu, Ji-Rong Wen, and Zhicheng Dou. EnvScaler: Scaling Tool-Interactive Environments for LLM Agent via Programmatic Synthesis. *arXiv: 2601.05808*, 2026.

Zirui Tang, Xuanhe Zhou, Yumou Liu, Linchun Li, Yukai Wu, Weizheng Wang, Hongzhang Huang, Wei Zhou, Jun Zhou, Jiachen Song, et al. Workspace-Bench 1.0: Benchmarking AI Agents on Workspace Tasks with Large-Scale File Dependencies. *arXiv: 2605.03596*, 2026.

Harsh Trivedi, Tushar Khot, Mareike Hartmann, Ruskin Manku, Vinty Dong, Edward Li, Shashank Gupta, Ashish Sabharwal, and Niranjan Balasubramanian. AppWorld: A Controllable World of Apps and People for Benchmarking Interactive Coding Agents. In *ACL*, 2024.

Dunwei Tu, Hongyan Hao, Hansi Yang, Yihao Chen, Yi-Kai Zhang, Zhikang Xia, Yu Yang, Yueqing Sun, Xingchen Liu, Furao Shen, et al. ScaleEnv: Scaling Environment Synthesis from Scratch for Generalist Interactive Tool-Use Agent Training. *arXiv: 2602.06820*, 2026.

Zhiheng Xi, Yiwen Ding, Wenxiang Chen, Boyang Hong, Honglin Guo, Junzhe Wang, Xin Guo, Dingwen Yang, Chenyang Liao, Wei He, Songyang Gao, Lu Chen, Rui Zheng, Yicheng Zou, Tao Gui, Qi Zhang, Xipeng Qiu, Xuanjing Huang, Zuxuan Wu, and Yu-Gang Jiang. AgentGym: Evaluating and Training Large Language Model-based Agents across Diverse Environments. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2025.

Tianbao Xie, Danyang Zhang, Jixuan Chen, Xiaochuan Li, Siheng Zhao, Ruisheng Cao, Toh Jing Hua, Zhoujun Cheng, Dongchan Shin, Fangyu Lei, Yitao Liu, Yiheng Xu, Shuyan Zhou, Silvio Savarese, Caiming Xiong, Victor Zhong, and Tao Yu. OSWorld: Benchmarking Multimodal Agents for Open-Ended Tasks in Real Computer Environments. In *Advances in Neural Information Processing Systems*, 2024.

Frank (Fangzheng) Xu, Yufan Song, Boxuan Li, Yuxuan Tang, Kritanjali Jain, Mengxue Bao, Zora Wang, Xuhui Zhou, Zhitong Guo, Murong Cao, Mingyang Yang, Hao Yang Lu, Amaad Martin, Zhe Su, Leander Maben, Raj Mehta, Wayne Chi, Lawrence Jang, Yiqing Xie, Shuyan Zhou, and Graham Neubig. TheAgentCompany: Benchmarking LLM Agents on Consequential Real World Tasks. In *Advances in Neural Information Processing Systems*, 2025.

Shunyu Yao, Noah Shinn, Pedram Razavi, and Karthik Narasimhan. τ-bench: A Benchmark for Tool-Agent-User Interaction in Real-World Domains. *arXiv: 2406.12045*, 2024.

Yucheng Zeng, Weipeng Lu, Linyun Liu, Shupeng Li, Zitian Qu, Chenghao Zhu, Shaofei Li, Zhengdong Tan, Mengyue Liu, Haotian Zhao, Zhe Zhou, and Jianmin Wu. LOGIGEN: Logic-Driven Generation of Verifiable Agentic Tasks. *arXiv: 2603.00540*, 2026.

Shuyan Zhou, Frank F Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, Uri Alon, and Graham Neubig. WebArena: A Realistic Web Environment for Building Autonomous Agents. In *International Conference on Learning Representations*, 2024.

Table A1: **Protocols for environment scaling and independent evaluation on Claw-Eval-Live.** Scaling-stage attempts determine when to stop recursion; depth reevaluation measures changes over saved lineages; the paired model comparison evaluates original and RES-scaled environments. A model turn is one target-model response, potentially containing multiple tool calls. Stopping uses complete fulfilment; independent success uses a native-score threshold of 0.80.

	Environment scaling	Depth reevaluation	Model comparison
Attempts per environment	3 at evaluated nodes	1	1
Model-turn limit	60	50	50
Success criterion	Complete task fulfilment	Native score ≥ 0.80	Native score ≥ 0.80
Target	DeepSeek-V4.1-Flash	DeepSeek-V4.1-Flash	Six evaluation models
Task cohort	105 seeds	105 seeds / 394 environments	105 matched seeds per model
Primary use	Stopping decisions	Recursive trends	Original vs. RES

A Evaluation and Environment Scaling

Evaluation protocol. We separate the target evaluations used to stop environment scaling from the independent evaluations used to report benchmark performance. During scaling, three attempts test complete task fulfilment. After scaling, we freeze the environments and run one attempt per task. On Claw-Eval-Live, independent evaluation counts native scores of at least 0.80 as successful. Table A1 summarizes the scaling, depth-reevaluation, and model-comparison settings.

Scaling configuration. GPT-5.6 Sol with high reasoning effort proposes transformations. Each lineage has a maximum depth of 10 and a budget of eight outer proposer executions per parent, including repairs and retries. We process 32 seeds concurrently; the three target attempts within each seed run sequentially. Every lineage includes evaluation of its original and last accepted environments, with intermediate evaluations requested by the proposer. Target outcomes determine stopping; the validation procedure in Section 3.3 determines whether a child is accepted.

Target configuration. Across all three benchmarks, target attempts have a 60-turn limit during scaling and a 50-turn limit during independent evaluation, with at most 16,384 output tokens per response. Timeouts follow the task-specific Claw configuration: 180, 240, 300, 420, or 600 seconds, held fixed within each original/scaled pair. All six evaluation models use high reasoning effort. DeepSeek-V4.1-Flash uses thinking with high effort throughout scaling, depth reevaluation, and the frozen endpoint comparison.

Workspace and terminal evaluation. Workspace-Bench-Lite contains 100 tasks and Terminal-Bench 2.1 contains 89. The latter uses the official 2.1 revision, which fixes 28 tasks from the 2.0 release.[1] Both follow the scaling and target configurations above. Paired environments use the same model settings, tool permissions, and scoring protocol. Workspace TCR@80 counts tasks that pass at least 80% of their rubric items; RPR pools passed items across tasks. Terminal-Bench uses its executable verifier's binary result. Success rates weight tasks equally.

Validation and stopping. Claw validation checks lineage and protected-contract hashes, replays the declared operators from the frozen parent, and compares the rebuilt files with the candidate. The original goal, facts, and grader remain fixed. Public traversal reconstructs files and service records, rejecting content loss or duplication, invalid references, nonterminating preparation, and original-file bypasses. The record stores replay hashes and execution traces. Stopping requires three assessed failures with no measurement errors.

Workspace and terminal adapters. Workspace transformations act on files, structured records, and workflow dependencies. The adapter binds derived views to source hashes and checks restored

[1] Terminal-Bench 2.1 release notes.

Table A2: **Proposer construction cost per accepted Claw-Eval-Live child.** Mean, median, and 90th-percentile (P90) costs cover all 289 children constructed by GPT-5.6 Sol with high reasoning effort. Each child's costs aggregate its outer executions and retries. Tokens include input and output; time sums execution durations. Target solving and grading are excluded.

Quantity	Mean	Median	P90
Outer proposer executions	1.45	1.00	2.00
Input + output tokens (k)	89.19	65.27	179.82
Summed execution time (s)	112.61	93.56	184.00

files or projected records against frozen inputs; Office and PDF resources are inspected with native readers. Reported TCR@80 and RPR use the frozen benchmark rubric. Terminal transformations act on container files, build configurations, database states, and service inputs. Frozen parent state and declared resource bindings identify the inputs to each child. Materialized artifacts undergo native readback and execution checks, and task scoring uses the executable verifier. Private tests and reference solutions remain on the evaluator side.

Environment validity and manual review. Three researchers reviewed 20 matched original/scaled pairs from each benchmark: 60 task lineages and 120 endpoint environments in total. They checked core objectives, required information, scoring conditions, answer leakage, and scoring bypasses, alongside programmatic checks of executability and task–evaluation consistency. None of these issues was found in the reviewed samples.

Construction cost. The 289 accepted children account for 420 outer executions, 25.78M input and output tokens, and 9.04 hours of summed proposer execution time. We first add the executions associated with each child, including retries, then compute the statistics in Table A2. Of these children, 193 require one execution and 73 require two, accounting for 92.0% together. Input totals include cached tokens; reasoning tokens are included once in the reported totals. Time measures summed execution durations under concurrent processing. These costs cover accepted-child construction and exclude branches producing no child, target solving, grading, and orchestration overhead.

B Training Data and Optimization

Training conditions. At each model scale, SFT w/o RES and SFT w/ RES start independently from the same Base checkpoint and use 8,428 trajectories with identical optimization settings (Table A3). The original trajectories are drawn from ClawGym and generated by DeepSeek-V4.1-Flash. For SFT w/ RES, RES is applied to this training task pool and GPT-5.6 Sol collects new trajectories in the scaled environments, using the same selection and supervision rules. Evaluation uses the original environments of the 294 benchmark tasks in Table 2.

Original-trajectory data. We select 8,428 distinct tasks from the released ClawGym task and trajectory collections, retaining one structurally complete rollout per task.[2] We deduplicate records across release formats and require a traceable task association, complete messages, matched tool calls and returns, and a released native score of at least 0.8. Filtering uses the released scores without replaying the trajectories. The selected data contain 3,381 full-score and 5,047 partial-score trajectories, each no longer than 49,152 tokens. This original-trajectory set contains 190.52M input tokens and 39.74M supervised target tokens; its median sequence length is 20,195 tokens.

Supervision. Training preserves the available reasoning, tool calls, tool outputs, and final responses. Loss applies to assistant reasoning, responses, and unmasked tool calls. System and user messages and tool outputs provide context. Calls with missing parameters or unverified semantics remain in context but are excluded from the loss; the original-trajectory set contains 78 such calls from 54 tasks. A custom ms-swift template preserves these token-level supervision labels.

[2]Frozen ClawGym releases: tasks (`bd9043cff88b`) and trajectories (`31824a4516a3`).

Table A3: **Optimization settings for the Qwen3.5 Base checkpoints in the training comparison.** +SFT w/o RES and +SFT w/ RES start independently from the same checkpoint and use these identical settings at each model size. Each condition trains on 8,428 trajectories for one epoch; sequence lengths and effective batch size are measured in tokens and examples, respectively. The checkpoint after the single training epoch is used for evaluation.

Setting	4B-Base	9B-Base	35B-A3B-Base
Peak learning rate	2×10^{-5}	10^{-5}	5×10^{-6}
Epoch / updates	1 / 527		
Maximum sequence length	49,152 tokens		
Effective batch size	16 (12 in the last update)		
Learning-rate schedule	Cosine, 3.00% warmup (16 steps)		
Optimizer	DeepSpeedCPUAdam (AdamW)		
Adam $\beta_1, \beta_2, \epsilon$	$0.90,\ 0.95,\ 10^{-8}$		
Weight decay / gradient clip	0.01 / 1.00		
Precision / hardware	BF16 / 4× H800		
Memory and attention	ZeRO-3, CPU optimizer offload, FlashAttention 2		
Gradient checkpointing	Enabled (non-reentrant)		
Packing / truncation	Disabled / overlength records excluded		
Training / data seed	42 / 42		

Optimization. We run one epoch of full-parameter SFT in BF16 on four H800 GPUs. A per-device batch of one with four accumulation steps gives a global batch of 16, with 12 examples in the last of 527 updates. We evaluate the final checkpoint; training uses no held-out validation split. Table A3 lists the hyperparameters. The software stack consists of ms-swift 4.6.0.dev0, PyTorch 2.11.0, Transformers 5.16.1, and DeepSpeed 0.19.6.

Evaluation-overlap checks. We screened the training tasks against all three evaluation benchmarks and found no matches under the checks performed: 105 Claw-Eval-Live, 100 English Workspace-Bench-Lite, and 89 Terminal-Bench 2.1 tasks. The Claw and Workspace records compare task identifiers, task text, input-content hashes, and lexical similarity. These checks address the recorded task and input representations.

C Recursive Evaluation and Aggregation

C.1 Current and Historical Statistics

We reevaluate 394 distinct environments from 105 Claw-Eval-Live seeds using DeepSeek-V4.1-Flash, with one 50-turn attempt per environment. For each seed i, let $m_i(r)$ be its measurement at depth r and D_i its final depth. At budget d, a stopped lineage retains its last accepted environment. The current-state mean is

$$M_m(d) = \frac{1}{N} \sum_{i=1}^{N} m_i(\min(d, D_i)), \qquad N = 105. \tag{4}$$

Figure 4 uses the historical mean, which averages each task's retained measurements over budgets zero through d and then averages equally across tasks:

$$H_m(d) = \frac{1}{N} \sum_{i=1}^{N} \frac{1}{d+1} \sum_{r=0}^{d} m_i(\min(r, D_i)). \tag{5}$$

At budget two, each task averages its measurements at budgets zero, one, and two. All six panels follow this rule, using binary values for success and turn-limit termination. Stopped lineages reuse their final measurement. Token and interaction statistics are retained-state averages; bands show the interquartile range across task-level historical means.

Both statistics keep all 105 seeds: M_m measures the retained environment and H_m includes earlier states. Figures A1 and A2 compare these views. At budget ten, this depth batch has 4.76% current-state success; the separate endpoint batch in Table 1 gives 5.71%.

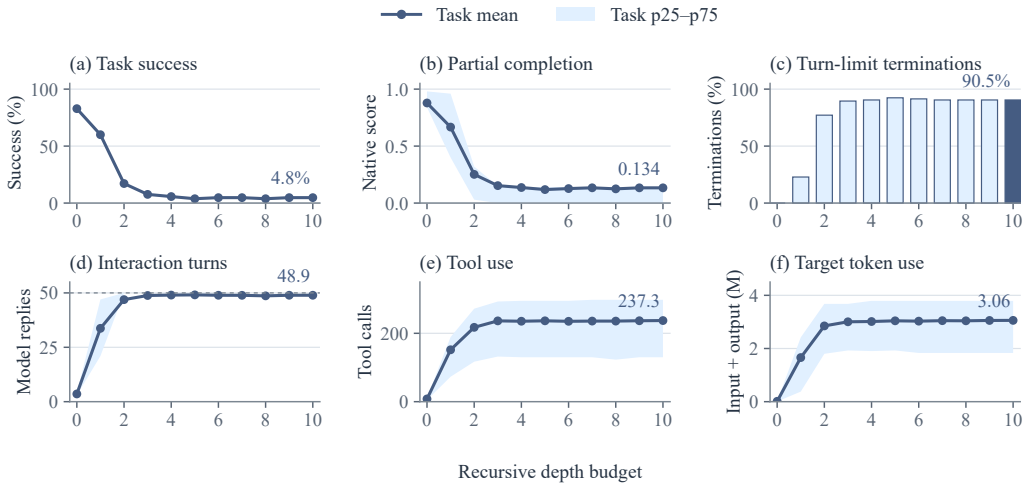

Figure A1: **Current retained-state performance and interaction demand.** Means and interquartile ranges cover all 105 seeds, retaining early-stopped finals without averaging earlier budgets.

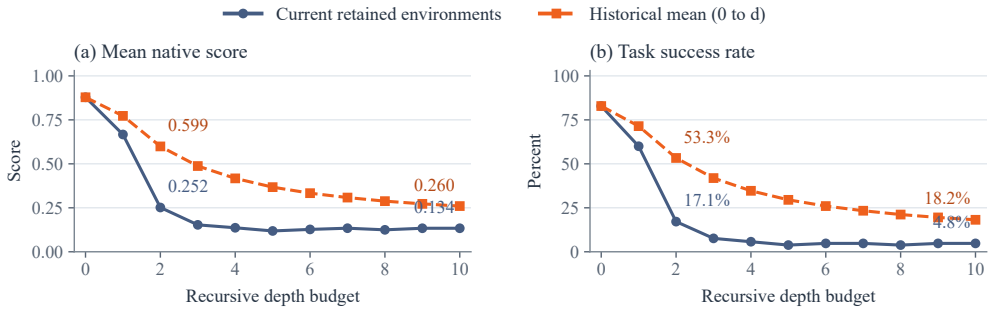

Figure A2: **Current versus historical benchmark means.** Historical means average the retained measurements from budgets zero through the current budget.

C.2 Interaction Demand at Selected Budgets

Figure A3 compares turns, tool calls, and tokens at budgets 0, 2, 4, 6, 8, and 10 using the historical aggregation in Equation 5.

Among the 105 lineages, 11 stop at depth one, 37 at two, 39 at three, 13 at four, one at five, two at six, one at eight, and one at ten. The numbers reaching depths zero through ten are $105, 105, 94, 57, 18, 5, 4, 2, 2, 1, 1$. Higher-budget summaries therefore combine the final environments of many stopped lineages with the few that continue. Keeping all seeds gives a benchmark-level view as the allowed recursion budget increases.

C.3 Performance Across Depth Budgets

We apply the current-state statistic in Equation 4 to the same 105 saved lineages. For pass@1, $m_i(r) = \mathbb{1}[s_i(r) \geq 0.80]$; for mean native score, $m_i(r) = s_i(r)$. Both use the final depth D_i and budget d defined above, retaining the last accepted environment after stopping. Table A4 reports all budgets, and Figure A4 plots the resulting performance.

Evaluation batches. At cap ten, this batch gives 5/105 passes and mean score 0.134, versus 6/105 and 0.142 in the separate endpoint-comparison batch (Table 1). Each batch independently evaluates the original environments using the single-attempt, 50-turn protocol in Appendix A.

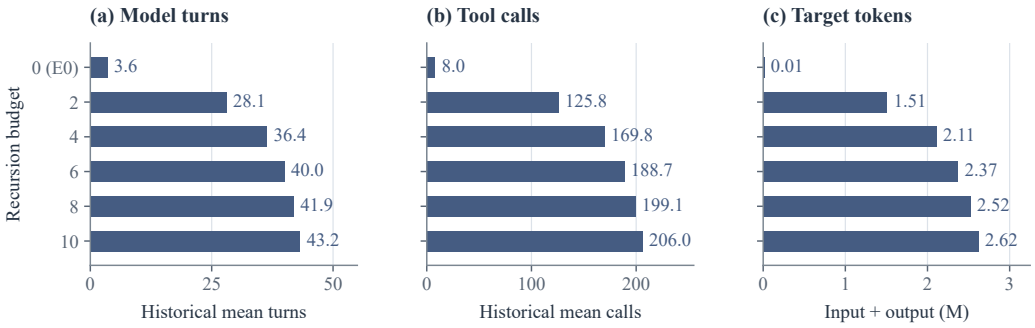

Figure A3: **Historical interaction demand at selected budgets.** Per-task means over retained states $0, \ldots, d$ are averaged equally across 105 seeds, including failed and turn-limited trajectories. Tokens include repeated input context.

Table A4: **Performance and interaction cost across recursion-depth budgets on Claw-Eval-Live.** DeepSeek-V4.1-Flash receives one 50-turn attempt per environment. Each budget d covers 105 tasks, retaining stopped environments. Reach d counts chains attaining depth d; Continue counts chains exceeding it. Pass@1 counts scores ≥ 0.80; Score is the native mean on a 0–100 scale. Turns, Tools, and Tokens are task means; tokens sum target input and output in millions.

d	Reach d	Continue	Pass@1	Score	Turns	Tools	Tokens (M)
0	105	105	87/105	87.84	3.55	8.05	0.01
1	105	94	63/105	66.69	33.71	151.79	1.66
2	94	57	18/105	25.17	46.91	217.47	2.85
3	57	18	8/105	15.30	48.81	236.37	3.00
4	18	5	6/105	13.65	49.00	235.34	3.02
5	5	4	4/105	11.84	49.10	236.54	3.04
6	4	2	5/105	12.71	48.91	235.10	3.03
7	2	2	5/105	13.40	48.90	235.77	3.05
8	2	1	4/105	12.48	48.65	235.73	3.04
9	1	1	5/105	13.37	48.92	236.75	3.05
10	1	0	5/105	13.37	48.92	237.30	3.06

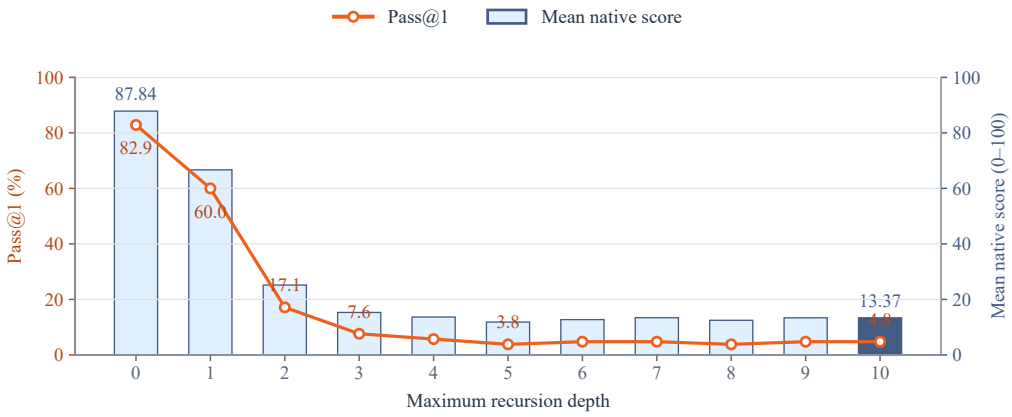

Figure A4: **Performance across recursion-depth caps.** Pass@1 (orange) and mean native score (blue) use a 0–100 scale over 105 seeds, retaining stopped environments. Table 3 selects caps 0, 1, 2, 3, 5, and 10.

D Component Ablation Details

D.1 Settings and Interventions

Task selection. We use 15 complete cases from 20 frozen lineages, selected by endpoint availability and matching planned depths after evaluation, following the original task-group order. Selection does not use target scores. A01 and D01 lack a matching Basic endpoint, C02 requires supplemental independent construction, and RESEARCH_01 and W03 fall outside the 15-task quota. The subset contains 2 file/multi-table, 3 cross-service, 4 finance/personnel, 3 planning, and 3 operations/workspace tasks. Table A5 lists the selected cases and planned depths.

Evaluation and generation budgets. DeepSeek-V4.1-Flash evaluates each component condition with thinking and high reasoning effort, a 50-turn limit, and at most 16,384 output tokens per response. It also supplies the native grade; a weighted score of at least 0.80 counts as a pass. GPT-5.6 Sol proposes transformations with high reasoning effort and a nominal budget of eight outer Codex executions per parent, covering queries, construction, repairs, and retries. We record proposer tokens without imposing a token cap, so matching construction opportunities does not imply equal realized token cost.

The depth analysis is a separate batch of 394 task-environment evaluations with reported outcomes. It uses DeepSeek-V4.1-Flash with thinking and high reasoning effort, a 50-turn limit, and a 16,384-token response cap, following Appendix A.

Removing dependencies. We resolve cross-operator references into directly usable inputs and release cross-resource preparation prerequisites in saved RES environments. Task instructions, underlying facts, and atomic transformation histories are retained. Model-free restoration checks verify environment consistency before target evaluation. This intervention removes both the composed access dependencies and the work needed to satisfy their prerequisites.

Removing recursive inheritance. For task i with planned depth d_i, we compare

$$\begin{aligned} \text{Recursive:} \quad & E_0 \to E_1 \to \cdots \to E_{d_i}, \\ \text{Independent:} \quad & E_1^{(j)} = \mathrm{Enhance}(E_0; j), \quad j = 1, \ldots, d_i. \end{aligned}$$

We evaluate E_{d_i} for the recursive arm and the predeclared last construction $E_1^{(d_i)}$ for the independent arm. Each independent attempt starts from the original environment, and construction order fixes the evaluated endpoint. Planned depths sum to 66 across the 15 tasks, giving both arms 66 construction opportunities with the same nominal per-parent execution budget. Neither arm uses target feedback during construction. The inheritance and library comparisons share a rebuilt RES control; dependency removal uses the saved RES control.

Transformation library. Full contains 17 predefined Claw operators. Basic retains six: sharding, pagination, summary-to-detail access, field references, numeric re-encoding, and export preparation. The additional eleven provide evidence dispersion, namespace bridges, source routing, candidate intersection, dependent lookups, multi-source exports, external-schema views, archive views, version selection, index reconstruction, and staged release. The proposer selects applicable operators from resource capabilities and composes them through explicit source bindings and prerequisite relations. The library is fixed during construction. Basic and Full use the same runtime, verifier, proposer settings, and planned task depths; their comparison includes both operator availability and the proposer's choices among supported compositions.

D.2 Per-task Results

Table A5 reports native scores for all 15 selected tasks. Figure A5 pairs each intervention with its corresponding RES control, keeping the task order fixed across comparisons.

Table A5: **Per-task native scores for the 15-task component ablations.** DeepSeek-V4.1-Flash receives one 50-turn attempt per environment; scores use a 0–100 scale, with success at 80 or above. Original is the unmodified task. Saved and Rebuilt denote full RES controls; w/o deps. removes dependencies from Saved, w/o rec. independently enhances Original and evaluates the predeclared last construction, and Basic rebuilds with six operators. The rebuilt variants share planned depth d_i.

Task	d_i	Original	Saved	w/o deps.	Rebuilt	w/o rec.	Basic
A04	10	93.00	0.00	89.00	0.00	97.00	95.00
D02	4	96.00	0.00	96.00	98.00	99.00	98.00
COMM_13	5	100.00	4.00	82.00	3.00	100.00	82.00
DATA_09	8	86.00	7.00	93.00	3.00	88.00	88.00
PRODAPP_01	3	92.00	14.00	99.00	15.00	99.00	97.00
DATA_07	3	96.00	0.00	96.00	3.00	100.00	98.00
DATA_20	4	84.00	7.00	85.00	5.00	90.00	5.00
HR_04	4	98.00	3.00	97.00	0.00	3.00	73.00
FIN_29	3	100.00	0.00	88.00	3.00	90.00	90.00
OPS_02	4	92.00	0.00	99.00	18.00	92.00	77.00
PRODAPP_17	3	99.00	3.00	98.00	0.00	98.00	97.00
SALES_02	4	93.00	3.00	84.00	3.00	89.00	93.00
SEC_01	4	98.00	0.00	97.00	0.00	95.00	93.00
SHELL_03	4	100.00	20.00	91.00	10.00	91.00	100.00
SHELL_06	3	100.00	55.00	100.00	55.00	100.00	100.00

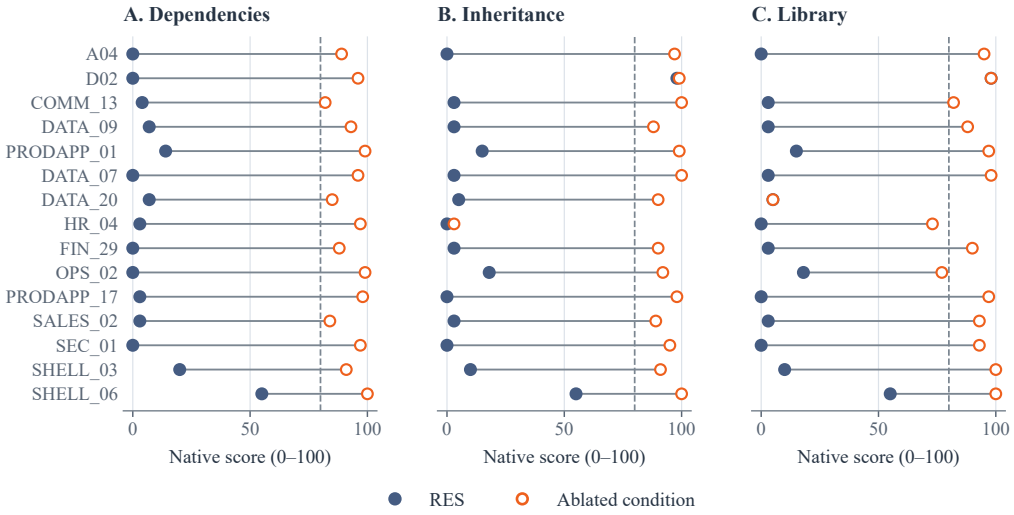

Figure A5: **Per-task effects of the component interventions.** Filled blue markers show the corresponding full RES control; hollow orange markers show the ablation. Segments connect the same task, and dashed lines mark the 80-point success threshold. All panels use the task order in Table A5.

D.3 Execution Cost and Score Sensitivity

Token and interaction cost. Table A6 aggregates input and output tokens across each target rollout, including cached input and reasoning tokens once in the reported totals. Relative token cost is the ratio of task means. A model turn may contain several tool calls, and these measurements include retries and unfinished trajectories. Every saved RES task reaches the 50-turn limit; removing its dependencies reduces the mean to 19.1 turns.

Table A6: **Target execution cost for the 15-task component comparisons.** Values are task means for DeepSeek-V4.1-Flash with one 50-turn attempt per environment, including failed and turn-limited attempts. Saved RES is the dependency-removal control; Rebuilt RES is shared by the independent-construction and basic-library comparisons. Tokens sum target input and output; k denotes thousands. Relative tokens divides the condition's mean by the Original mean before rounding. Proposer construction and grading costs are excluded.

Condition	Turns	Tool calls	Tokens (k)	Relative tokens (×)
Original	2.87	6.67	8.43	1.00
Saved RES	50.00	226.93	3115.42	369.40
w/o dependencies	19.07	85.60	723.76	85.82
Rebuilt RES	49.73	163.93	1644.20	194.96
w/o inheritance	21.20	90.40	651.09	77.20
Basic library	18.53	39.60	227.01	26.92

Table A7: **Success counts under alternative native-score thresholds on Claw-Eval-Live.** DeepSeek-V4.1-Flash uses one 50-turn attempt per environment. Each entry counts tasks with score $\geq \tau$ over the cohort size. Component rows use 15 paired tasks; Saved RES controls dependency removal, and Rebuilt RES controls inheritance and library changes. Depth rows use 105 separately reevaluated seeds, retaining stopped environments at higher budgets. Counts use saved scores.

Condition	$\tau = 0.70$	$\tau = 0.80$	$\tau = 0.90$
Original	15/15	15/15	13/15
Saved RES	0/15	0/15	0/15
w/o dependencies	15/15	15/15	10/15
Rebuilt RES	1/15	1/15	1/15
w/o inheritance	14/15	14/15	12/15
Basic library	14/15	12/15	10/15
Depth cap 0	92/105	87/105	68/105
Depth cap 1	68/105	63/105	45/105
Depth cap 2	18/105	18/105	15/105
Depth cap 3	8/105	8/105	4/105
Depth cap 5	4/105	4/105	1/105
Depth cap 10	6/105	5/105	2/105

Sample size and thresholds. The component results comprise 90 rollouts: one attempt for each of 15 tasks in six conditions, including the original environment. Inheritance and library comparisons reuse the rebuilt RES control. Table A7 recomputes success counts from these scores under alternative thresholds. All comparisons are descriptive and paired by task.