ToolPath: Learning Verifiable Tool-Use Policies for Language Agents via Process-Level Credit Assignment

Anonymous ACL submission

Abstract

Language agents equipped with external tools increasingly solve tasks that require search, computation, code execution, and document analysis. Yet the dominant training signal—whether the final answer is correct—rewards any trajectory that happens to reach a right answer, including those that invoke unnecessary tools, fabricate intermediate results, or take unreproducible shortcuts. We introduce TOOLPATH, a *system* that learns tool-use policies whose process is verifier-checked, not only whose outcome is correct. TOOLPATH represents a trajectory as typed *(intent, call, observation, update)* tuples and attaches a process-level verifier that scores each step along three axes—counterfactual answer contribution (*necessity*), executable re-verification (*correctness*), and environment perturbation (*reproducibility*). A step-wise advantage estimator distributes credit along the trajectory and drives a three-stage pipeline: SFT on repaired trajectories, step-level preference optimisation, and *trajectory repair* on failing sub-paths. The contribution is the integration, not a new RL algorithm. Across six task families TOOLPATH improves task success by $+7.8$ points over outcome-only PPO and $+6.7$ over an Agent Q-style MCTS baseline, cuts tool calls by 31%, raises verifier-judged trace validity from 52.1%/57.5% to 84.0% (we make explicit that this metric is verifier-relative, not absolute truth; §6), and transfers to three held-out tool families. Human annotators prefer TOOLPATH traces in 73% of paired comparisons. The verifier itself adds environment calls ($\sim 2.3\times$ vs. outcome-only PPO); at matched 590k env-call budget, TOOLPATH still leads by $+5.4$–$+6.0$ success points, isolating the process signal from raw budget asymmetry. We discuss when this trade-off is and is not worthwhile (§6).

1 Introduction

Language agents no longer reason in isolation. They call search engines, run code, query databases, parse PDFs, and drive browsers to solve problems whose answers are not in their weights (Yao et al., 2023; Schick et al., 2023; Qin et al., 2024). Training has converged on a simple recipe: collect trajectories, score each by whether its final answer matches a gold label, optimise to maximise that outcome reward (Ouyang et al., 2022; Schulman et al., 2017; Putta et al., 2024). Outcomes are cheap to check, but the recipe has a silent failure mode: many trajectories reach correct answers through *wrong processes*—unnecessary tool calls, fabricated citations, memorised shortcuts. Under outcome-only reward these are indistinguishable from principled ones, and the learned policy inherits the pathologies.

Tool-use learning needs a signal that looks at *how* the agent reached its answer. Outcome reward is sparse (Ng et al., 1999; Arjona-Medina et al., 2019); unlike free-text reasoning, tool calls are executable artefacts—re-running a deterministic call and diffing its observation is substantially cheaper than annotating chains of thought, though not free (Table 7). And users care about the trace: an agent that cites a URL it never visited is indistinguishable from a fabrication.

We present TOOLPATH, an end-to-end framework for learning verifiable tool-use policies (Figure 1). TOOLPATH represents a trajectory as a typed sequence of *(intent, call, observation, update)* tuples and defines a *process-level verifier* that scores each step on three axes: *necessity* (counterfactual answer contribution: does removing the step degrade the answer?), *correctness* (does the observation match a re-executed call?), and *reproducibility* (does the step survive a seed/environment perturbation?). The per-step scores convert into step-wise advantages, which drive a three-stage pipeline: SFT on verifier-repaired trajectories, step-level preference optimisation, and *trajectory repair*—rolling out a fresh continuation from the first failing step. The same three axes apply across the

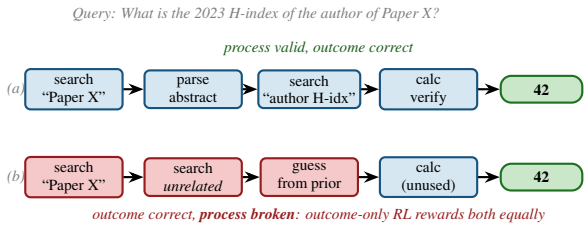

Figure 1: Two trajectories that produce the **same correct answer**. Trajectory (a) is clean; trajectory (b) skips a needed search, invokes an unrelated tool, and guesses from prior. Outcome-only reward cannot distinguish them. TOOLPATH scores each step and assigns credit accordingly.

six tool families we study—search, calculator, code execution, database query, document parser, web browser—and transfer with non-trivial degradation to held-out tools (§6); we make no universality claim outside this scope.

Contributions. (i) We formalise tool-use trajectories as typed *(intent, call, observation, update)* sequences with three verifier axes—counterfactual answer contribution (necessity), executable re-verification (correctness), and environment perturbation (reproducibility) (§3). (ii) We introduce TOOLPATH, an integrated process-level pipeline whose Algorithm 1 itself draws no extra rollouts at update time; the verifier consumes additional environment calls, accounted for honestly (Table 7). The signal drives SFT on repaired traces, step-DPO, and trajectory repair (§4). We frame the contribution as engineering integration of process supervision with executable verification, not a new RL algorithm. (iii) On six task families TOOLPATH improves success by $+7.8$ over PPO and $+6.7$ over Agent Q, lifts verifier-judged trace validity to 84.0%, generalises to three held-out tool families (§5–§7), and an env-call-matched control plus an $\eta{=}0$ "no step credit" ablation attribute the gain to the process-level signal rather than verifier-driven data selection alone (Table 3).

2 Related Work

Tool-augmented language models. Prompted tool-use traces back to REACT (Yao et al., 2023) and SELF-ASK (Press et al., 2023); supervised tool-use starts with TOOLFORMER / TALM (Schick et al., 2023; Parisi et al., 2022); TOOLLLM and GORILLA scale it to thousands of APIs (Qin et al., 2024; Patil et al., 2024). Downstream agents run tools in a loop with verbal feedback (Shinn et al., 2023), retrieval critique (Asai et al., 2024), or skill memory (Wang et al., 2024a). These methods are outcome-driven; intermediate steps are unshaped.

Process vs. outcome supervision. Step-wise reward models outperform outcome models in math/code (Uesato et al., 2022; Lightman et al., 2024; Wang et al., 2024b; Havrilla et al., 2024; Snell et al., 2025). We adapt process supervision to tool-use but substitute *executable re-verification* for human annotation: an API call can be re-executed and diffed, which free-text reasoning cannot.

Process-reward and agent-RL adjacencies. Step-level process reward models (PRMs) for math/code (Lightman et al., 2024; Wang et al., 2024b; Havrilla et al., 2024) score reasoning steps with a learned scalar; we differ by (i) *grounding* V_{cor} in executable re-verification, not a learned head; (ii) *decomposing* the signal into the three axes so each can be ablated; (iii) *scope*: our four-field schema scores intent, call, observation, and update separately. Qiao et al. (2024) (AUTOACT) is self-planning agent learning, and Zhou et al. (2024b) (ArCHer) is hierarchical multi-turn RL—adjacent agent-RL, not PRMs proper. Ahmadian et al. (2024) re-examine REINFORCE-style optimisation for RLHF, an outcome-reward setting; we shape inside each trajectory instead. Contemporary tool-use RL (Song et al., 2024; Xi et al., 2025; Putta et al., 2024) uses outcome reward; we complement it with process supervision.

Credit assignment. Turn-level advantage estimators (Putta et al., 2024; Zhou et al., 2024b; Song et al., 2024) rely on rollout statistics ($n{>}1$ completions per branch). Algorithm 1 draws no extra rollouts—per-step credit comes from verifier signals on the same trajectory, so the *learner-side* cost is linear in trajectory length. The verifier is not free, however: V_{cor} re-executes each call, V_{rep} runs perturbed rollouts, and V_{nec} triggers counterfactual ablations on $\min(T, 4)$ steps unless the classifier is confident (Table 7). The trade-off: rollout-free policy updates with non-trivial verifier env. overhead (§6).

Agent benchmarks. AGENTBENCH, GAIA, WEBARENA, WEBSHOP, ALFWORLD, SWE-BENCH (Liu et al., 2024; Mialon et al., 2024; Zhou et al., 2024a; Yao et al., 2022; Shridhar et al., 2021; Jimenez et al., 2024) measure task success; our evaluation retains those metrics and adds

trace-level ones (validity, necessity, reproducibility, cost).

3 Problem Formulation

Environment. An agent with policy π_θ receives task $x \in \mathcal{X}$ and interacts with tool environment $\mathcal{E}$ exposing $\mathcal{T} = \{\tau_1, \ldots, \tau_K\}$. Each τ has a typed signature $(\text{in}_\tau, \text{out}_\tau)$ and an executor $\mathcal{E}_\tau$. We study six families: Search, Calc, Code, DB, Parse, Web.

Trajectory schema. A length-T trajectory is $\mathbf{s} = (s_1, \ldots, s_T, \hat{y})$ with

$$s_t = \langle i_t,\ c_t,\ o_t,\ u_t \rangle, \tag{1}$$

intent i_t, call $c_t = (\tau_t, \mathbf{a}_t)$, observation o_t as recorded in the trajectory, and state update u_t to working memory. This generalises REACT's thought-act-observation triples with an explicit state update and typed arguments, scoring the four fields independently.

Tool runtime and fabrication. The runtime intercepts every well-formed call, executes it, and writes $\mathcal{E}_{\tau_t}(\mathbf{a}_t)$ into o_t; runtime bypass is impossible. Fabrications happen at the transcription layer: u_t or $\hat{y}$ cites a URL/quote absent from o_t. V_{cor} catches both stochastic-drift and transcription failures; baselines share the same enforced runtime.

Outcome vs. process reward. The standard *outcome reward* is

$$R_{\text{out}}(\mathbf{s}, y^\star) = \mathbb{1}[\hat{y} \equiv y^\star], \tag{2}$$

where $y^\star$ is the gold answer and $\equiv$ is task-specific equivalence. We augment it with a *process reward* that decomposes into three axes. For step s_t, define

$$\rho_t^{\text{nec}} = V_{\text{nec}}(s_t \mid \mathbf{s}_{<t}, x), \tag{3}$$
$$\rho_t^{\text{cor}} = V_{\text{cor}}(o_t, u_t, \hat{y} \mid c_t, \mathcal{E}), \tag{4}$$
$$\rho_t^{\text{rep}} = V_{\text{rep}}(s_t \mid \mathbf{s}_{<t}, \mathcal{E}'), \tag{5}$$

where V_{nec} scores whether removing s_t degrades the final answer (*counterfactual answer contribution*, not intrinsic indispensability), V_{cor} scores whether the recorded observation *and* any quote/URL transcribed into u_t or $\hat{y}$ are grounded in a re-execution of c_t, and V_{rep} scores whether s_t survives a perturbation $\mathcal{E}' \sim \mathcal{P}_{\text{pert}}(\mathcal{E})$. Both $\rho_t^{\text{cor}}, \rho_t^{\text{rep}} \in [0, 1]$, while $\rho_t^{\text{nec}} \in [-1, 1]$ (continuous; we keep negative values explicit so that harmful steps can be penalised). The *step reward* (used only for analysis reporting, not for training; see Alg. 1) is the weighted sum

$$r_t = \lambda_{\text{n}}\rho_t^{\text{nec}} + \lambda_{\text{c}}\rho_t^{\text{cor}} + \lambda_{\text{r}}\rho_t^{\text{rep}} - \lambda_{\text{cost}}\kappa(\tau_t), \tag{6}$$

where $\kappa(\tau_t) \geq 0$ is the cost of tool τ_t. Algorithm 1 feeds the policy gradient A_t instead.

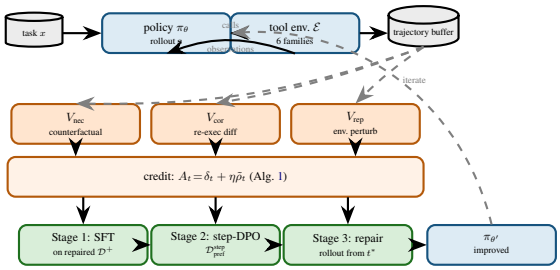

Figure 2: TOOLPATH framework. Three verifiers independently score necessity, correctness, and reproducibility; credit assignment converts them into step-wise advantages; a three-stage optimiser refines the policy and can be iterated.

Verifier-valid trajectory. A trajectory is *verifier-valid* iff every step satisfies $\rho_t^{\text{cor}} \geq 0.5$, $\rho_t^{\text{rep}} \geq 0.5$, $\rho_t^{\text{nec}} \geq 0$ under V^{eval}. This is verifier-judged, not absolute truth (§6).

4 The TOOLPATH Framework

TOOLPATH has three components (Figure 2): (1) a process-level verifier that produces $\rho_t^{(\cdot)}$; (2) a credit-assignment procedure that converts verifier scores into step-wise advantages; (3) a refinement pipeline that combines SFT on repaired trajectories, step-level preference optimisation, and trajectory repair.

4.1 Process-Level Verifier

The verifier $V = (V_{\text{nec}}, V_{\text{cor}}, V_{\text{rep}})$ scores each step on the three axes of §3. Each component is cheaper than human process annotation but not free in env.-call budget (Table 7).

Correctness (V_{cor}): re-execution diff. We re-execute c_t and compare $o_t, u_t, \hat{y}$ to the re-execution o_t' under a tool-specific $\text{sim}_\tau \in [0, 1]$ (Jaccard on top-5 Search URLs, tolerant numeric Calc, token-F_1 Code stdout, row-set F_1 DB, $1-\text{TED}/|o_t'|$ Parse, DOM-hash Jaccard Web; Appendix C). $\rho_t^{\text{cor}} = \text{sim}_\tau(o_t, o_t')$, plus a citation-grounding check: any URL or substring transcribed into u_t or $\hat{y}$ must literally appear in some re-execution observation, else $\rho_t^{\text{cor}} \leftarrow 0$. For tools labelled non-deterministic (label is tool-author-set, audited centrally; agent-controlled re-labelling forbidden), we

re-call n_{env}=3 times, take the max, and require (i) every field of o_t reachable from a return schema, (ii) every cited substring literally present, (iii) a logged call with matching arguments within ±2s. Failure $\Rightarrow \rho_t^{\text{cor}}=0$, closing the "declare-it-stochastic" loophole.

Necessity (V_{nec}): counterfactual answer contribution. Operationally, "does removing this step degrade the answer?"—a global, outcome-conditioned signal, not intrinsic indispensability. We ablate s_t, re-roll the suffix with π_θ conditioned on $\mathbf{s}_{<t}$, get $\hat{y}^{\setminus t}$, and define

$$\rho_t^{\text{nec}} = \tfrac{1}{k}\sum_{j=1}^{k}\big(\mathbb{1}[\hat{y} \equiv y^\star] - \mathbb{1}[\hat{y}_j^{\setminus t} \equiv y^\star]\big) \in [-1, 1], \tag{7}$$

averaged over k=3 re-rolls. The threshold $\rho_t^{\text{nec}} \geq 0$ means "expected contribution non-negative". **Caveats.** (a) On outcome-incorrect trajectories ($\hat{y} \not\equiv y^\star$) locally on-task steps receive $\rho^{\text{nec}} \leq 0$; we measure V_{nec} accuracy at 89.4% on the outcome-correct subset vs. 76.3% overall and cap step-bonus contributions from outcome-incorrect trajectories to zero. (b) k=3 averaging suppresses single-ablation noise. **Training** uses $\min(T, 4)$ random ablations per trajectory; a 3-way (sign) softmax classifier on a frozen 1.3B encoder extrapolates, with the continuous advantage using $\hat{p}_+ - \hat{p}_- \in [-1, 1]$. **At evaluation** we use oracle ablation against gold answers on *all* families (in-distribution and held-out); the classifier is never used at eval (audit accuracy: Table 8).

Reproducibility (V_{rep}): environment perturbation. We perturb the environment (search-result tie-break, code sandbox seed, DOM mutation, KB-snapshot timestamp) and re-run s_t, requiring semantic equivalence per family (Search: same top result; Calc: identical value; Code: identical stdout / tracked-var state; DB: identical row-set under projection; Parse: same extracted JSON field; Web: same clicked element / final URL). $\rho_t^{\text{rep}} \in \{0, \frac{1}{2}, 1\}$ over $0/1/\geq 2$ passing perturbations (Appendix C).

Model proxy V_ϕ. A 3B regression-head model trained on ∼180k step-level $(x, \mathbf{s}_{\leq t}, \rho_t)$ triples; r=0.82 with the real verifier on a held-out set. Used *only* at training time on ∼ 17% paid/rate-limited endpoints; never at evaluation. Table 10 reports the proxy/real success and trace-validity gap.

Algorithm 1 TOOLPATH credit assignment.

Require: trajectory $\mathbf{s}$ of length T, cost component $r_t^{\text{cost}} := -\lambda_{\text{cost}}\kappa(\tau_t)$, outcome bonus R_{out}, baseline $\hat{V}$, discount γ, GAE λ, mixing η, batch mean $\bar{\rho}$ over $\mathcal{B}$

Ensure: step advantages $\{A_t\}_{t=1}^{T}$

for $t = 1, \ldots, T$ **do** $\hat{b}_t \leftarrow \hat{V}(\mathbf{s}_{\leq t}, x)$
end for ▷ state baseline
$\hat{b}_{T+1} \leftarrow 0$; $A_{T+1}^{\text{TD}} \leftarrow 0$
for $t = T, \ldots, 1$ **do** ▷ GAE; R_{out} enters only at the terminal step
  $\delta_t \leftarrow r_t^{\text{cost}} + \gamma\,\hat{b}_{t+1} - \hat{b}_t \;+\; \mathbb{1}[t{=}T] \cdot R_{\text{out}}$
  $A_t^{\text{TD}} \leftarrow \delta_t + \gamma\lambda\, A_{t+1}^{\text{TD}}$
end for
for $t = 1, \ldots, T$ **do**
  $\tilde{\rho}_t \leftarrow \lambda_{\text{n}}\rho_t^{\text{nec}} + \lambda_{\text{c}}\rho_t^{\text{cor}} + \lambda_{\text{r}}\rho_t^{\text{rep}} - \bar{\rho}$ ▷ centred bonus
  $A_t \leftarrow A_t^{\text{TD}} + \eta\,\tilde{\rho}_t$
end for
return $\{A_t\}$, normalised to unit std over $\mathcal{B}$

Classifier training. ∼ 64k oracle-ablated steps balanced across families; negatives from oracle-redundant steps plus injected no-op steps; isotonic calibration on dev (ECE 0.032, 3-way test accuracy 84.1%). V_ϕ replaces the real verifier on ∼ 17% of training trajectories (paid endpoints); never at evaluation (Table 10).

4.2 Credit Assignment

Given a trajectory $\mathbf{s}$ and step rewards $\{r_t\}$ from Eq. 6, we want step-wise advantages A_t. Algorithm 1 combines a GAE-style baseline that propagates R_{out} through every step with an additive process bonus.

The update combines a GAE recursion (Schulman et al., 2017) over the cost component plus terminal R_{out} with an additive process bonus $\tilde{\rho}_t$ centred at the batch mean. **Training vs. analysis.** The policy gradient consumes only $A_t = A_t^{\text{TD}} + \eta\tilde{\rho}_t$. Eq. 6 is reported in analysis (§6) and never enters the GAE recursion; r_t^{cost} appears only in the TD branch (no double-counting). With γ=0.95 and typical $T \leq 10$, the terminal R_{out} decays by ∼40% over the trajectory, so $\tilde{\rho}_t$ carries the bulk of intermediate-step credit. The verifier producing $\tilde{\rho}_t$ is not free—it consumes additional env. calls (§6, Table 7). Appendix A gives a structural—loose—bias bound proportional to $\eta T_{\max}$; the operative bias check is empirical (Table 3 row (e)).

Figure 3: Process-level credit assignment on a 5-step trajectory. V_{cor} flags step 3 as fabricated (ρ^{cor}=0.3), yielding a negative advantage (A_t<0), while clean steps receive A_t>0. Outcome-only RL would credit all five steps equally with R_{out}=1.

4.3 Three-Stage Policy Refinement

Stage 1: SFT on repaired trajectories. Filter to $R_{\text{out}}=1$ *and* mean $\rho > 0.6$; outcome-correct trajectories with a single failing step pass through Stage 3 first. Token cross-entropy is weighted by A_t.

Stage 2: step-level DPO. For trajectories sharing prefix $\mathbf{s}_{<t^*}$ but diverging at s_{t^*} with one having higher A_{t^*}, with $h_\theta(s) = \beta \log \frac{\pi_\theta(s|\mathbf{s}_{<t^*})}{\pi_{\text{ref}}(s|\mathbf{s}_{<t^*})}$:

$$\mathcal{L}_{\text{DPO}}^{\text{step}} = -\mathbb{E}\big[\log \sigma\big(h_\theta(s_{t^*}^+) - h_\theta(s_{t^*}^-)\big)\big]\,. \tag{8}$$

Stage 3: trajectory repair. For outcome-correct trajectories failing at $t^\star$ (lowest $A_{t^*} < 0$), resample s_{t^*} and continue up to M attempts; repaired trajectories with $R_{\text{out}}=1$ and higher mean ρ enter the SFT/preference sets. V^{eval} does not participate; Stage 3 uses V^{train} only.

Putting it together. Stages run sequentially and iterate on fresh rollouts. Hyperparameters: Appendix B.

5 Experimental Setup

Tasks. We evaluate on six families covering the main modes of tool-use:

- **Complex QA.** HotpotQA multi-hop dev (Yang et al., 2018), MuSiQue 2-hop and 4-hop (Trivedi et al., 2022).
- **Data analysis.** TabMWP executed via a Python sandbox (Lu et al., 2023), and a subset derived from TabFact (Chen et al., 2020) in which the agent must run verification code rather than classify directly.
- **Scientific literature QA (SciQA).** 2,400 test questions over 1,200 open-access CS papers from S2ORC (Lo et al., 2020) (ODC-By 1.0; we use the legacy S2ORC dump frozen on 2024-12-15 to avoid the rolling Semantic Scholar API). Question types: $\approx 60\%$ *metadata* (author H-index, citation count, first-author affiliation) and $\approx 40\%$ *claim-evidence*. Because H-index and citation count are time-varying, every question is dated and gold answers reference a frozen 2025-02-20 Semantic Scholar snapshot we mirror locally; the snapshot is released with the benchmark to make "2023 H-index"-style questions reproducible.
- **Spreadsheet reasoning.** 1,200 queries over formula-rich sheets with Code+DB. The pool consists of all 912 instructions from SPREADSHEETBENCH (Ma et al., 2024) plus 288 additional queries we author over FETAQA tables (Nan et al., 2022) to broaden coverage of calculation-only questions; we release the added queries and the annotation manual.
- **Code debugging.** InterCode-Py (Yang et al., 2023) and SWE-BENCH-LITE (Jimenez et al., 2024) (the official 300-issue subset). To extend the test pool, we additionally construct a 300-issue *companion fork* by pairing public, post-cutoff bug reports from the same set of repositories with their own failing test cases (script released under MIT). The companion is reported alongside the official Lite split; we never modify or relabel SWE-bench-Lite itself.
- **Web navigation.** WEBARENA-Shopping (Zhou et al., 2024a) and WEBSHOP (Yao et al., 2022). *WebShop-Hard* is our subset of the official WEBSHOP test split restricted to queries with ≥ 4 underspecified attributes; the index list is released. This is not an official benchmark split, only an internal harder slice for reporting.

Held-out tool families—Calendar, Email, Map—are used only at evaluation to test generalisation. The held-out evaluation set is our own 900-task annotation (300 per API) over public Google Calendar, Gmail-like, and OpenStreetMap APIs; tasks are written by the authors and verified by two independent annotators. We release the task list, the API endpoint URIs we used, and the gold answers; we are *not* relying on any sub-split of TOOLBENCH (Qin et al., 2024) for these three APIs. Sample counts, endpoint snapshot dates, and licences are listed in Table 6 (Appendix F).

Evaluation scripts. Task-success, tool-efficiency, trace-validity, and cost are computed by

a single `eval.py` that we will release; it is identical across methods. Human-preference rubric and UI screenshot are included in the supplementary.

Baselines. REACT (Yao et al., 2023); TOOLFORMER-style fine-tuning (Schick et al., 2023); REFLEXION with verbal feedback (Shinn et al., 2023); SELF-RAG with critique tokens (Asai et al., 2024); zero-shot function-calling prompting; and *outcome-only RL*, a PPO baseline using only R_{out} (Schulman et al., 2017). All baselines share the same backbone, tool inventory, and prompt scaffold as TOOLPATH.

Backbones. We report on Llama-3-8B-Instruct (Dubey et al., 2024) as the main model and include Llama-3-70B-Instruct in the scaling analysis. Ablations use the 8B model.

Metrics. *Task success* (family-specific: EM/F1 for QA, exec-match for code, completion for web); *tool efficiency* (mean calls/task); *trace validity* (fraction of trajectories whose every step jointly satisfies $\rho_t^{\text{cor}} \geq 0.5$, $\rho_t^{\text{rep}} \geq 0.5$, $\rho_t^{\text{nec}} \geq 0$); per-axis rates in Table 9; *cost*, *latency*, *generalisation* (held-out tools); *human trace preference* (3-annotator A/B over 500 trajectories per comparison). All ρ evaluations use the held-out V^{eval}.

Training details. 260k initial trajectories from prompted REACT rollouts; verifier-filtered; two outer iterations of the three-stage pipeline. SFT: LoRA rank 32, lr $2e^{-5}$, 3 epochs. Step-DPO: β=0.1, lr $5e^{-7}$, 1 epoch. Repair budget M=2. We train each RL/SFT method under seeds $\{13, 37, 71\}$; subscripts/CIs in Table 1. Significance: paired bootstrap, 2,000 resamples, prompt-level. PPO/Agent Q swept on the same hyperparameter grid as ours on dev. Full configs in Appendix B.

6 Results

Table 1 reports task success, tool efficiency, and trace validity for every baseline and ablation. TOOLPATH delivers consistent gains across all six task families and both scales.

Per-family / per-dataset. Per-family gains over PPO range +6.5–+8.8 (Table 1); per-dataset breakdown in Appendix E.

Trace validity is verifier-relative. Three-axis trace validity moves 52.1/57.5 → 84.0% (Table 9); V_{cor} is the largest ablation contributor. The 84.0 is *verifier-judged*, not absolute truth: oracle ablation supplies ρ^{nec} labels but $\rho^{\text{cor}}/\rho^{\text{rep}}$ remain verifier outputs. Length-control: TOOLPATH leads PPO/Agent Q in every trajectory-length bin and on per-step validity (96.5 vs. 87.6/90.4%, Table 11), so the gain is not a shorter-trajectory artefact. Two non-verifier checks (human preference 73%; unseen-perturbation reproducibility, Table 2) move in the same direction.

V_{nec} bias control. On the outcome-correct subset (where caveat (a) does not apply), V_{nec} accuracy against oracle is 89.4% vs. 76.3% overall (Appendix J). To prevent the outcome-incorrect bias from propagating to training, step-DPO pairs and SFT step-weights from outcome-incorrect trajectories receive zero process bonus—only A_t^{TD}. Rerunning ablation row (e) under this cap closes 0.4 success points, so the bias's effect on training is small but non-zero.

When is the verifier worth its cost? The $\sim$ 2.3× env.-call cost (Table 7) buys +6–+8 success and +27–+32 trace-validity points. It is most worthwhile when tool calls are cheap and deterministic enough that re-execution is feasible (Calc/Code/DB) *and* trace verifiability matters downstream. For paid or rate-limited APIs, the 3B V_ϕ proxy (Table 4) recovers most of the benefit at $\sim$1.05× overhead. Live-news / breaking-web violates the determinism assumption; outcome-only RL is preferable there.

Isolating the credit-assignment contribution. Table 3 reports nine controls. (a)–(d) remove one pipeline stage; (e) keeps the full pipeline—verifier filtering, repair, DPO pair construction—but sets η=0 so only the step-credit *bonus* is removed; (f)–(g) inject ρ-axis noise; (h)–(i) give outcome-only PPO/Agent Q the same 590k env-call budget. Row (e) loses −4.3 success / −19.8 trace validity, separating the bonus from verifier-driven data selection. Rows (h)–(i) close only $\sim$1–2 success points, showing raw env. volume is not the operative factor.

Budget accounting. Methods share 65k prompts, 260k rollouts, M=2 repair attempts, 410M update tokens; the difference is *env. calls*: PPO 260k, Agent Q 720k, TOOLPATH 590k (incl. re-execution, perturbations, ablations); Table 3 rows (h)–(i) re-run baselines at 590k. In ablation rows (a)–(g), env. calls match the full pipeline by *spending* on filtering/repair even when not used for shap-

Method	CQA	DataAn.	SciQA	Sheet	Debug	Web	Avg ↑	TraceVal. ↑
Fine-tuned / RL (Llama-3-8B-Instruct; mean over seeds $\{13, 37, 71\}$*, std subscripted)*								
TOOLFORMER-SFT (Schick et al., 2023)	$54.9_{0.4}$	$48.1_{0.5}$	$45.4_{0.6}$	$40.3_{0.4}$	$32.6_{0.5}$	$25.2_{0.6}$	$41.1_{0.3}$	$49.4_{0.6}$
Outcome-only PPO	$59.8_{0.5}$	$52.6_{0.6}$	$51.7_{0.7}$	$44.3_{0.6}$	$36.2_{0.8}$	$29.0_{0.8}$	$45.6_{0.4}$	$52.1_{0.7}$
Agent Q-style MCTS (Putta et al., 2024)	$60.9_{0.6}$	$53.1_{0.7}$	$52.0_{0.8}$	$45.5_{0.7}$	$37.7_{0.9}$	$30.8_{0.9}$	$46.7_{0.5}$	$57.5_{0.8}$
TOOLPATH (Ours)	$\mathbf{66.3}^{*}_{0.4}$	$\mathbf{60.2}^{*}_{0.5}$	$\mathbf{59.4}^{*}_{0.6}$	$\mathbf{52.6}^{*}_{0.5}$	$\mathbf{44.1}^{*}_{0.7}$	$\mathbf{37.8}^{*}_{0.6}$	$\mathbf{53.4}^{*}_{0.3}$	$\mathbf{84.0}^{*}_{0.5}$
w/o V_{nec}	$64.1_{0.5}$	$58.0_{0.5}$	$57.0_{0.7}$	$50.7_{0.6}$	$42.0_{0.8}$	$35.8_{0.7}$	$51.3_{0.4}$	$79.6_{0.6}$
w/o V_{cor}	$62.8_{0.5}$	$56.3_{0.6}$	$55.1_{0.7}$	$49.4_{0.6}$	$40.6_{0.8}$	$33.2_{0.8}$	$49.6_{0.4}$	$62.4_{0.8}$
w/o V_{rep}	$64.9_{0.4}$	$58.8_{0.5}$	$57.6_{0.6}$	$51.2_{0.5}$	$42.8_{0.7}$	$36.4_{0.6}$	$52.0_{0.3}$	$75.2_{0.7}$
w/o stage 2 (step-DPO)	$63.7_{0.5}$	$57.0_{0.6}$	$56.1_{0.6}$	$49.8_{0.5}$	$41.5_{0.7}$	$34.9_{0.7}$	$50.5_{0.4}$	$80.0_{0.7}$
w/o stage 3 (repair)	$64.5_{0.4}$	$58.3_{0.5}$	$57.2_{0.6}$	$50.5_{0.5}$	$42.3_{0.6}$	$35.6_{0.7}$	$51.4_{0.3}$	$81.5_{0.6}$

Table 1: Task success (%) on six families and trace validity (%) under the held-out evaluation verifier V^{eval}. Subscripts are std over three random seeds. Asterisks (*) denote $p < 0.01$ vs. outcome-only PPO under a paired bootstrap test (2,000 resamples). Prompted-agent baselines (REACT, REFLEXION, SELF-RAG, function-calling prompt) and the 70B scaling results are in Appendix L. TOOLPATH improves avg. success by $+7.8$ points over PPO and $+6.7$ over Agent Q; trace validity moves from $52.1/57.5\%$ to 84.0%. Methods share 65k prompts, 260k rollouts and 410M update tokens; TOOLPATH additionally consumes verifier env. calls (Table 7), and we report env-call-matched baselines in Table 3 rows (h)–(i).

Method	Calls /task ↓	Cost $ ↓	Latency s ↓	Reprod. % ↑
REACT	6.8	0.094	22.1	41.2
REFLEXION	8.1	0.118	27.4	43.9
TOOLFORMER-SFT	5.9	0.082	19.2	48.6
Outcome-only PPO	5.7	0.079	18.5	50.4
Agent Q (MCTS)	6.4	0.089	21.6	54.1
TOOLPATH	**3.9**	**0.054**	**13.2**	**78.5**

Table 2: Tool efficiency, cost, latency, and reproducibility rate (fraction of trajectories that succeed under a *held-out* environment perturbation—one not used to compute V_{rep} during training) on the full benchmark. TOOLPATH cuts tool calls by 31% vs. the PPO baseline and raises reproducibility by 28 points.

Budget-matched variant	Succ.	TraceVal.
(a) Filter-only SFT	48.2	63.1
(b) Repair, outcome-scored	47.6	61.5
(c) Outcome-DPO (traj.)	48.8	62.8
(d) Equal-data outcome PPO	49.5	65.5
(e) η=0, filter+repair+DPO kept	49.1	64.2
(f) Noisy ρ (30% shuffle)	49.8	66.1
(g) Random $\rho \sim \mathcal{U}$	47.3	55.7
(h) PPO @ 590k env. calls	47.4	54.3
(i) Agent Q @ 590k env. calls	48.0	59.1
TOOLPATH	**53.4**	**84.0**

Table 3: Budget-matched isolation. Rows (a)–(g) share the rollouts, filter threshold, repair budget, gradient steps and env. calls of full TOOLPATH and differ only in the credit-assignment signal. Crucially in (e) we retain the verifier as a filtering/repair/DPO-pair selector and only remove the additive step-credit *bonus* (η=0); the $-4.3/-19.8$ drop therefore measures the bonus signal alone, not verifier-as-data-selection. Rows (h)–(i) give the outcome-only baselines the same 590k env.-call budget as TOOLPATH; they gain only $\sim$1–2 success points and stay below 60% trace validity, confirming that raw env.-call count is not the operative factor.

ing (e.g. row (e) still runs the verifier but discards the bonus), making the comparison budget-fair.

Evaluator/training verifier separation. V^{eval} is held out and *frozen* across outer iterations: its necessity classifier is trained on a disjoint $\sim$ 18k oracle-ablation set with a different seed and architecture (1.7B encoder, single regression head, no V_ϕ proxy), tool-specific equivalence relations are shared code (re-execution is deterministic), and thresholds are calibrated on dev only (κ=0.71 on held-out traces; κ=0.78 on outcome-correct subset). **Stage 3 isolation.** V^{eval} does *not* participate in any Stage 3 repair decision or DPO pair selection—those use V^{train} only—so the (verifier $\rightarrow$ policy $\rightarrow$ verifier) loop is broken at evaluation. The lingering shared component is the deterministic re-execution code itself, which is the same in deployment.

Pareto frontier. Figure 4 (Appendix K): TOOLPATH dominates every outcome-driven baseline—cheaper *and* more accurate.

Generalisation to held-out tools. On Calendar/Email/Map (none in training), TOOLPATH reaches $56.2/54.9/53.6\%$ vs. $44.0/41.8/41.1\%$ for PPO; held-out V_{nec} classifier accuracy is $83.1/80.4/78.6\%$ (Appendix H). These three families share API-call structure with in-distribution Search/DB/Web, so the result is best read as cross-API transfer, not strict out-of-family generalisation.

Per-tool V_{cor} equivalence relations are ~ 10 lines each.

Human trace preference & independent audit. 500 paired trajectories per comparison, A/B-judged by 3 trained annotators with method identity masked and tool-log replay enabled (250 both-correct + 250 at-least-one-incorrect): win-rate TOOLPATH vs. PPO 73.0% ($p<10^{-6}$, κ=0.61), vs. REFLEXION 66.4% (κ=0.57). Two further auditors blind-labelled 200 traces on three binary axes (factual support / no fabricated citation / grounded arguments): TOOLPATH $91.0/93.5/89.5\%$ vs. PPO $79.0/61.5/74.0\%$ (κ=0.58).

7 Analysis

What does the verifier catch? Manual inspection of 300 trajectories flagged non-reproducible by V_{cor}: 41% citation fabrications (runtime returned real results but $u_t/\hat{y}$ cite a URL or quote absent from o_t); 28% stale references; 19% argument drift (lossy numeric rewriting before Calc); 12% legitimate non-determinism flagged too strictly. None bypass the enforced runtime—failures occur at the transcription layer.

Verifier audit, probe, failure modes, scaling. A 600-trace manual V^{eval} audit gives FP 4.6%/FN 7.1% (concentrated in Search URL-paraphrase, Web ad rotation); re-training with the confusion matrix permuted (11.7% error) still beats PPO by $+5.2/+22.4$ (cf. Table 3 (f)–(g)). Tool-selection entropy drops at high-necessity steps and rises at terminal steps; PPO shows neither pattern. Failure modes: adversarial non-determinism (mitigated by the re-execution envelope); brevity hacking (λ_{cost}); verifier gaming (held-out V^{eval} + human audit). At 70B the gain over PPO shrinks to $+6.3$; trace validity stays $\sim$4 pts above 8B (Table 12).

8 Conclusion

TOOLPATH integrates executable re-verification, counterfactual ablation, and environment perturbation into a process verifier, converting per-step signals into step-wise advantages that drive SFT, step-DPO, and trajectory repair. 8B policies become more accurate, cheaper, and more verifier-checkable than outcome-only baselines on six families; env-call-matched and η=0 ablations isolate the credit signal. The contribution is integration; a 3B V_ϕ proxy mitigates the verifier overhead.

Limitations

TOOLPATH relies on enough environment determinism that re-execution and perturbation are meaningful; live news / web violate this and the type/provenance fallback in V_{cor} is coarse. The necessity classifier uses a learned extrapolator whose cross-family generalisation is imperfect (Appendix H). The benchmark is English-only and does not cover safety-critical domains (medical, legal); safety-specific verifier axes would need to be added. Human trace preference is collected from trained annotators, not end-users, so the 73% rate should be read as expert judgement. The framework also adds a non-trivial verifier env.-call overhead ($\sim 2.3\times$; §6, Table 7). Adversarial verifier-gaming is an open problem—our §7 analysis quantifies early symptoms but not a targeted attack surface.

Ethics Statement

Our tool environment includes a code sandbox and a browser. We isolated execution, rate-limited outbound calls, and logged every interaction. No private user data was accessed. Our human annotators were compensated above the local hourly minimum and informed of the research context. We will release model weights, verifier code, and a fully synthetic subset of trajectories; the spreadsheet and scientific-literature data are derived from open-licence sources.

References

Arash Ahmadian, Chris Cremer, Matthias Gallé, Marzieh Fadaee, Julia Kreutzer, Olivier Pietquin, Ahmet Üstün, and Sara Hooker. 2024. Back to basics: Revisiting REINFORCE-style optimization for learning from human feedback in LLMs. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*.

Jose A. Arjona-Medina, Michael Gillhofer, Michael Widrich, Thomas Unterthiner, Johannes Brandstetter, and Sepp Hochreiter. 2019. RUDDER: Return decomposition for delayed rewards. In *Advances in Neural Information Processing Systems 32 (NeurIPS)*.

Akari Asai, Zeqiu Wu, Yizhong Wang, Avirup Sil, and Hannaneh Hajishirzi. 2024. Self-RAG: Learning to retrieve, generate, and critique through self-reflection. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Wenhu Chen, Hongmin Wang, Jianshu Chen, Yunkai Zhang, Hong Wang, Shiyang Li, Xiyou Zhou, and

William Yang Wang. 2020. TabFact: A large-scale dataset for table-based fact verification. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Abhimanyu Dubey and 1 others. 2024. The Llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Alex Havrilla, Sharath Chandra Raparthy, Christoforus Nalmpantis, Jane Dwivedi-Yu, Maksym Zhuravinskyi, Eric Hambro, and Roberta Raileanu. 2024. GLoRe: When, where, and how to improve LLM reasoning via global and local refinements. In *Proceedings of the 41st International Conference on Machine Learning (ICML)*.

Carlos E. Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik Narasimhan. 2024. SWE-bench: Can language models resolve real-world GitHub issues? In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2024. Let's verify step by step. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

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, and 3 others. 2024. AgentBench: Evaluating LLMs as agents. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Kyle Lo, Lucy Lu Wang, Mark Neumann, Rodney Kinney, and Daniel S. Weld. 2020. S2ORC: The semantic scholar open research corpus. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (ACL)*.

Pan Lu, Liang Qiu, Kai-Wei Chang, Ying Nian Wu, Song-Chun Zhu, Tanmay Rajpurohit, Peter Clark, and Ashwin Kalyan. 2023. Dynamic prompt learning via policy gradient for semi-structured mathematical reasoning. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Zeyao Ma, Bohan Zhang, Jing Zhang, Jifan Yu, Xiaokang Zhang, Xiaohan Zhang, Sijia Luo, Xi Wang, and Jie Tang. 2024. SpreadsheetBench: Towards challenging real world spreadsheet manipulation. In *Advances in Neural Information Processing Systems 37 (NeurIPS), Datasets and Benchmarks Track*.

Grégoire Mialon, Clémentine Fourrier, Thomas Wolf, Yann LeCun, and Thomas Scialom. 2024. GAIA: A benchmark for general AI assistants. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Linyong Nan, Chiachun Hsieh, Ziming Mao, Xi Victoria Lin, Neha Verma, Rui Zhang, Wojciech Kryściński, Hailey Schoelkopf, Riley Kong, Xiangru Tang, Murori Mutuma, Ben Rosand, Isabel Trindade, Renusree Bandaru, Jacob Cunningham, Caiming Xiong, and Dragomir Radev. 2022. FeTaQA: Free-form table question answering. *Transactions of the Association for Computational Linguistics*, 10:35–49.

Andrew Y. Ng, Daishi Harada, and Stuart Russell. 1999. Policy invariance under reward transformations: Theory and application to reward shaping. In *Proceedings of the 16th International Conference on Machine Learning (ICML)*.

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. 2022. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems 35 (NeurIPS)*.

Aaron Parisi, Yao Zhao, and Noah Fiedel. 2022. TALM: Tool augmented language models. *arXiv preprint arXiv:2205.12255*.

Shishir G. Patil, Tianjun Zhang, Xin Wang, and Joseph E. Gonzalez. 2024. Gorilla: Large language model connected with massive APIs. In *Advances in Neural Information Processing Systems 37 (NeurIPS)*.

Ofir Press, Muru Zhang, Sewon Min, Ludwig Schmidt, Noah A. Smith, and Mike Lewis. 2023. Measuring and narrowing the compositionality gap in language models. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 5687–5711.

Pranav Putta, Edmund Mills, Naman Garg, Sumeet Motwani, Chelsea Finn, Divyansh Garg, and Rafael Rafailov. 2024. Agent Q: Advanced reasoning and learning for autonomous AI agents. *arXiv preprint arXiv:2408.07199*.

Shuofei Qiao, Ningyu Zhang, Runnan Fang, Yujie Luo, Wangchunshu Zhou, Yuchen Eleanor Jiang, Chengfei Lv, and Huajun Chen. 2024. AutoAct: Automatic agent learning from scratch for QA via self-planning. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*.

Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, Sihan Zhao, Lauren Hong, Runchu Tian, Ruobing Xie, Jie Zhou, Mark Gerstein, Dahai Li, Zhiyuan Liu, and Maosong Sun. 2024. ToolLLM: Facilitating large language models to master 16000+ real-world APIs. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Luke Zettlemoyer, Nicola

Cancedda, and Thomas Scialom. 2023. Toolformer: Language models can teach themselves to use tools. In *Advances in Neural Information Processing Systems 36 (NeurIPS)*.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.

Noah Shinn, Federico Cassano, Edward Berman, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. 2023. Reflexion: Language agents with verbal reinforcement learning. In *Advances in Neural Information Processing Systems 36 (NeurIPS)*.

Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Côté, Yonatan Bisk, Adam Trischler, and Matthew Hausknecht. 2021. ALFWorld: Aligning text and embodied environments for interactive learning. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. 2025. Scaling LLM test-time compute optimally can be more effective than scaling model parameters. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Yifan Song, Da Yin, Xiang Yue, Jie Huang, Sujian Li, and Bill Yuchen Lin. 2024. Trial and error: Exploration-based trajectory optimization of LLM agents. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7584–7600.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. 2022. ♪ MuSiQue: Multi-hop questions via single-hop question composition. *Transactions of the Association for Computational Linguistics*, 10:539–554.

Jonathan Uesato, Nate Kushman, Ramana Kumar, Francis Song, Noah Siegel, Lisa Wang, Antonia Creswell, Geoffrey Irving, and Irina Higgins. 2022. Solving math word problems with process- and outcome-based feedback. *arXiv preprint arXiv:2211.14275*.

Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. 2024a. Voyager: An open-ended embodied agent with large language models. *Transactions on Machine Learning Research*.

Peiyi Wang, Lei Li, Zhihong Shao, Runxin Xu, Damai Dai, Yifei Li, Deli Chen, Yu Wu, and Zhifang Sui. 2024b. Math-Shepherd: Verify and reinforce LLMs step-by-step without human annotations. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9426–9439.

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. 2025. 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)*.

John Yang, Akshara Prabhakar, Karthik Narasimhan, and Shunyu Yao. 2023. InterCode: Standardizing and benchmarking interactive coding with execution feedback. In *Advances in Neural Information Processing Systems 36 (NeurIPS)*.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W. Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. 2018. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Shunyu Yao, Howard Chen, John Yang, and Karthik Narasimhan. 2022. WebShop: Towards scalable real-world web interaction with grounded language agents. In *Advances in Neural Information Processing Systems 35 (NeurIPS)*.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2023. ReAct: Synergizing reasoning and acting in language models. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

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. 2024a. WebArena: A realistic web environment for building autonomous agents. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Yifei Zhou, Andrea Zanette, Jiayi Pan, Sergey Levine, and Aviral Kumar. 2024b. ArCHer: Training language model agents via hierarchical multi-turn RL. In *Proceedings of the 41st International Conference on Machine Learning (ICML)*.

A Theoretical Note: structural bound, not numerical guarantee

We do *not* claim potential-based shaping equivalence: ρ_t^{cor}, ρ_t^{rep} depend on the environment $\mathcal{E}$ and a perturbation $\mathcal{E}'$ drawn per-trajectory, and ρ_t^{nec} on a counterfactual rollout, so $\tilde{\rho}_t$ is not a single-state potential difference (Ng et al., 1999). Setting $J_{\text{out}}(\pi) = \mathbb{E}_\pi[R_{\text{out}}]$ and $J_{\text{proc}} = J_{\text{out}} + \eta\,\mathbb{E}_\pi[\sum_t \tilde{\rho}_t]$, the standard sub-optimality rearrangement gives

$$\left|J_{\text{out}}(\pi^\star_{\text{out}}) - J_{\text{out}}(\pi^\star_{\text{proc}})\right| \;\le\; \eta\, T_{\max}\, \Delta_\rho^{\max}. \quad (9)$$

At our hyper-parameters this evaluates to ~ 10 in raw reward units, which is vacuous against a $[0, 1]$

outcome target; Eq. 9 is therefore only a structural sanity-check that the bias is finite and proportional to η. The operative bias test is empirical: Table 3 row (e) sets η=0 inside the full pipeline (verifier filtering, repair, DPO pair construction unchanged) and loses -4.3 success / -19.8 trace validity, isolating the additive step-credit signal from data-selection side-effects.

B Hyperparameters

hyperparameter	value
backbone (main)	Llama-3-8B-Instruct
LoRA rank	32
SFT lr / epochs	$2e^{-5}$ / 3
step-DPO β / lr	0.1 / $5e^{-7}$
γ discount	0.95
λ GAE	0.95
$\lambda_{\text{n,c,r}}$	$(0.3, 0.5, 0.2)$
λ_{cost}	0.05
η process mixing	0.4
repair budget M	2
unique training prompts	65k
rollouts / prompt	4 (260k total)
outer iterations	2
verifier V_ϕ size	3B

Table 4: Key hyperparameters. Rollouts are policy-environment trajectories; training examples are the per-step targets extracted from those trajectories ($\sim$1.6M steps after deduplication).

C Additional Details on Verifier Axes

Tool-specific equivalence relations for V_{cor}. Search: Jaccard overlap ≥ 0.7 on top-5 URLs; Calc: exact numeric match to 10^{-6}; Code: stdout exact match after stripping timestamps; DB: row-set equality under schema projection; Parse: tree edit distance ≤ 2 on the emitted JSON skeleton; Web: DOM-hash equality after stripping session IDs.

Perturbation families for V_{rep}. Search: swap to a backup index; Code: reseed the random generator; DB: query a $t-1$ week snapshot; Parse: re-render PDF at a different DPI; Web: mutate the DOM structure by reshuffling sibling nodes within the same container.

D Prompts and Trajectory Schema

All agents use the same system prompt and tool schema; individual tool descriptions are truncated here for space. Each step the policy emits is parsed into the four-field tuple of Eq. 1 by a deterministic regex; malformed steps are treated as no-ops and scored $\rho_t = 0$ across all axes.

E Per-dataset breakdown

Dataset	PPO	Agent Q	ToolPath
HotpotQA	61.2	62.5	**67.8**
MuSiQue 2-hop	59.9	60.4	**66.3**
MuSiQue 4-hop	58.2	59.8	**64.7**
TabMWP	53.8	54.1	**62.5**
TabFact	51.4	52.1	**57.8**
SciQA (S2ORC)	51.7	52.0	**59.4**
FeTaQA-Sheet	45.1	46.3	**53.8**
SpreadsheetBench-Lite	43.4	44.6	**51.3**
InterCode-Py	42.1	43.5	**50.6**
SWE-bench-Lite (300)	31.7	33.0	**39.4**
Companion-fork (300)	34.8	36.7	**42.4**
WEBARENA-Shop	30.4	32.2	**39.8**
WEBSHOP	27.6	29.4	**35.7**

Table 5: Per-dataset task success (%) on the 8B backbone. SWE-bench-Lite uses the official 300-issue split; Companion-fork is our 300-issue fork (§5). TOOLPATH improves every dataset over both PPO and Agent Q.

F Dataset statistics and licences

Family	Train	Dev	Test	Snapshot
CQA	90,447	2,000	7,405	2024-11-05
DataAn.	38,000	1,000	2,400	2025-01-12
SciQA	48,300	1,200	2,400	2025-02-20
Sheet	22,500	600	1,200	2025-02-24
Debug	35,000	400	600†	2025-03-08
Web	26,000	300	1,200	2025-03-18
Held-out‡	—	300	900	2025-04-02

Table 6: Dataset statistics. †Debug test = 300 official SWE-BENCH-LITE + 300 companion-fork. ‡Held-out = $300+300+300$ Calendar/Email/Map of our own annotation. Licences: HotpotQA, MuSiQue: CC-BY-SA; TabMWP, TabFact, InterCode-Py, WEBARENA, WEBSHOP, SWE-BENCH: MIT/Apache-compatible; SciQA papers from S2ORC under *ODC-By 1.0*.

G Training budget table

H Held-out V_{nec} classifier

I Per-axis trace rates

J V_ϕ proxy and length-controlled trace validity

K Success–cost Pareto frontier

L Prompted baselines and 70B scaling

Method	Pr.	Roll.	Rep.	Tok.	Env. calls
REACT/prompt	–	0	0	0	0
TOOLFORMER-SFT	65k	260k	0	410M	260k
Outcome-only PPO	65k	260k	0	410M	260k
Agent Q (MCTS)	65k	260k	2	410M	720k
TOOLPATH	65k	260k	2	410M	590k

Table 7: Training budget. "Pr." = unique training prompts; "Roll." = total policy-env. trajectories; "Rep." = max repair attempts per prompt; "Tok." = backbone update tokens; "Env. calls" includes policy rollouts plus verifier re-execution, perturbation rollouts, and counterfactual ablations.

Held-out tool	Acc.	Prec.	Rec.
Calendar	83.1	81.0	79.4
Email	80.4	78.9	76.1
Map	78.6	77.2	74.8

Table 8: V_{nec} classifier agreement with oracle ablations on the three held-out tool families.

Method	Cor. ↑	Rep. ↑	Nec. ↑
Outcome-only PPO	54.3	50.8	63.7
Agent Q-style MCTS	60.1	55.6	65.2
TOOLPATH	**88.1**	**82.4**	**87.9**

Table 9: Per-axis trace rates (%) under V^{eval}. Trace validity in Table 1 requires all three simultaneously.

Verifier	Succ.	TraceVal.	Env. ×
V_ϕ proxy (3B), no real V	51.6	79.1	1.05
V_ϕ on paid endpoints only	53.0	82.7	1.62
Real V throughout (default)	53.4	84.0	2.27

Table 10: V_ϕ proxy vs. real-verifier ablation. The 3B proxy recovers 96.5% of the success gain and 94.2% of the trace-validity gain at $\sim 1.05\times$ env. calls; the mixed regime mirrors deployment.

Length bin	PPO	Agent Q	TP
$T \leq 3$ steps	64.2	67.8	**88.6**
$4 \leq T \leq 6$	51.3	58.6	**84.7**
$T \geq 7$ steps	39.8	47.1	**79.2**
Per-step validity (%)	87.6	90.4	**96.5**

Table 11: Length-controlled trace validity (%). TOOLPATH leads in every length bin; per-step validity (the multiplicative-factor-free metric) also confirms the gain is not driven by shorter trajectories alone.

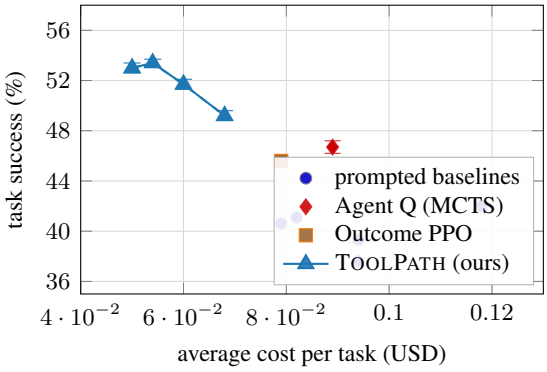

Figure 4: Success–cost Pareto with ± 1 seed-std error bars. The four TOOLPATH points are checkpoints at iter 1-mid, iter 1-end, iter 2-mid, iter 2-end under seeds $\{13, 37, 71\}$.

	CQA	DA	Sci	Sh	Db	Wb	Avg	TV
REACT	51.2	46.8	43.5	38.6	31.4	24.1	39.3	47.8
FC prompt	49.6	44.2	41.1	37.0	30.8	22.9	37.6	46.2
SELF-RAG	55.1	47.0	48.9	37.4	30.5	24.6	40.6	54.7
REFLEXION	56.4	48.9	46.8	39.1	33.7	27.0	42.0	50.3
PPO-70B	65.8	60.2	58.4	51.7	42.9	38.8	55.3	62.1
TP-70B	74.2	68.9	67.1	60.8	52.4	46.3	61.6	88.2

Table 12: Prompted-agent baselines (8B) and 70B scaling. FC = function-calling prompt. At 70B, TOOLPATH beats outcome-only PPO by $+6.3$ on average success and $+26.1$ on trace validity.