

RiskGuard: Hypergraph-Based Pre-Execution Risk Control for Long-Horizon Tool Use

Yandong Yan¹ Zesen Cai² Daming Shi³
Yifei He² Ruiting Dai^{4,*} Yu Huang^{2,*}

¹Key Laboratory of High Confidence Software Technologies,
School of Computer Science, Peking University

²National Engineering Research Center for Software Engineering, Peking University

³Department of Automation, Tsinghua University

⁴School of Information and Software Engineering,

University of Electronic Science and Technology of China

2501111974@stu.pku.edu.cn, zesen.cai@std.uestc.edu.cn

shidm18@tsinghua.org.cn, yfhe0929@gmail.com

rtdai@uestc.edu.cn, hy@pku.edu.cn

Abstract

In long-horizon tool use, early-step defects passing per-call checks often propagate silently across dependency chains to cause irreversible late-stage failures—a propagation class accounting for 73.0% of diagnosed Vanilla FC failures. Because these defects are locally valid (*stepwise legality*) yet manifest differently across tool boundaries (*semantic heterogeneity*), they easily evade standard per-call guardrails and post-hoc reflection, necessitating proactive pre-execution risk anticipation. To this end, we formalize the pre-execution risk propagation problem and propose **RiskGuard**, a training-free, dev-calibrated runtime controller. To capture the multi-source, conjunctive dependencies inherent in tool composition, RiskGuard models execution traces via a typed dependency hypergraph. Meanwhile, a Referential/Constraint/Operational (R/C/O) kernel tracks risk transitions across semantic boundaries. Replacing rigid binary blocking, a five-mode intervention rule selects the lowest-commitment feasible response to mitigate risks smoothly. Evaluated on ToolSandbox-hard, pooled τ^2 -Bench, and ToolHop, RiskGuard achieves absolute gains of +0.133 Sim (0.281 $\rightarrow$ 0.414), +9.2 pp pass@1, and +4.85 pp accuracy over the baseline, with depth diagnostics showing stronger gains on harder dependency slices.

1 Introduction

In long-horizon tool-use chains, failures can emerge even when individual tool calls appear locally valid at the step where they are made. In

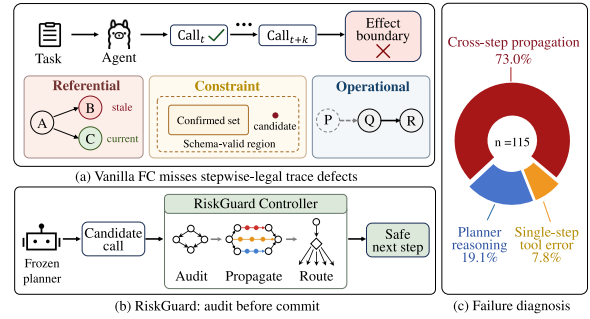


Figure 1: Motivation and overview of RiskGuard. (a) Vanilla FC can miss stepwise-legal referential, constraint, and operational trace defects that become risky only after downstream reuse. (b) RiskGuard audits each candidate before commitment by propagating R/C/O risk over the candidate-conditioned trace and routing to a safe next step. (c) In 115 diagnosed Vanilla FC failures, cross-step propagation accounts for 73.0%, dominating planner reasoning and single-step tool errors.

a long-horizon session, an early tool call may introduce a wrong entity binding, stale evidence, an ambiguous requirement, or a missing prerequisite. Although locally well-formed, this defect may be reused as confirmed evidence until a later effect-boundary action, such as a database write or financial transaction, consumes it and commits an irreversible effect. Long-horizon safety therefore requires auditing whether unresolved trace defects can propagate through dependencies to an effect-boundary action.

Figure 1 illustrates both the failure class and the control interface. As shown in Figure 1a, we group stepwise-legal trace defects into three recurring types: referential defects, where later calls reuse stale or ambiguous entity bindings; constraint de-

*Corresponding authors.

fects, where declared user, schema, policy, scope, capability, or explicit-confirmation constraints are violated or left unresolved; and operational defects, where required prerequisite calls, runtime-state checks, or verification observations are missing before commitment. These defects can enter the trace through locally valid calls, but become risky once downstream calls depend on them. Figure 1b shows how RiskGuard uses the same R/C/O taxonomy as a runtime interface: it audits the candidate-conditioned trace, propagates typed risk, and routes the candidate to a safe next step before commitment.

Using the annotation protocol summarized in Appendix B.1, we diagnose 115 failure trajectories from a no-intervention function-calling baseline (Vanilla FC) across ToolSandbox-hard and τ^2 -Bench. As shown in Figure 1c, cross-step propagation is the dominant origin, accounting for 73.0% of failures. Crucially, in every cross-step case, the error-introducing call passes the per-call validity checks available at the step where it occurs. These failures therefore have two structural properties. First, they are stepwise legal: validating the candidate call in isolation does not expose the defect. Second, they are semantically heterogeneous: the same upstream defect can manifest as referential, constraint, or operational risk depending on which downstream tool consumes it. Stepwise legality demands trace-level detection; semantic heterogeneity demands typed, non-binary intervention.

Existing reliability mechanisms are complementary, but they do not directly target locally legal defects that become risky only through trace-level propagation. Planner-side methods improve the quality of proposed actions; per-call guards inspect the current candidate; and graph- or taint-based defenses model dependencies under specific threat models. RiskGuard operates at a complementary interface: rather than improving planner deliberation or checking the candidate in isolation, it propagates typed R/C/O risks over the accumulated trace and, when needed, routes candidates to non-binary repair before commitment.

We formalize this setting as candidate-conditioned pre-execution risk propagation: given a typed trace prefix and a proposed action, the controller determines whether unresolved trace defects can reach an effect boundary through the candidate’s dependencies and selects a lowest-commitment safe response. RiskGuard is a training-free deterministic runtime controller that

keeps the base planner frozen; its decision-layer scalar constants are fixed by lightweight dev calibration, with no gradient-based optimization or backbone fine-tuning. It temporarily inserts each candidate into a typed dependency hypergraph, propagates referential, constraint, and operational risk with cross-channel transport, and routes the candidate through a non-binary intervention interface before any effect-boundary commitment. To our knowledge, RiskGuard is the first pre-execution layer that decomposes long-horizon tool-use risk into typed R/C/O channels and propagates these risks through a typed representation that transfers across benchmarks without per-task retuning.

Our contributions are as follows:

- We identify a trace-level failure taxonomy for long-horizon tool use. By diagnosing 115 Vanilla FC failure trajectories, we find that cross-step propagation accounts for 73.0% of failures; it also exposes stepwise legality and semantic heterogeneity, two properties that remain systematically invisible to per-call validity checks.
- We formulate candidate-conditioned pre-execution risk propagation as the problem of deciding, before commitment, whether unresolved trace defects can reach an effect boundary through a proposed action. We instantiate this formulation with a candidate-augmented typed dependency hypergraph that preserves conjunctive prerequisite relations flattened by binary graphs.
- We propose RiskGuard, a training-free, dev-calibrated deterministic controller that keeps the base planner frozen. It combines R/C/O risk diffusion with a five-mode intervention rule, routing each candidate to execution, constrained execution, clarification, probing, or safe fallback without modifying the base planner.
- Across three long-horizon runtime benchmarks, RiskGuard improves the primary metric over Vanilla FC by +0.133 Sim on ToolSandbox-hard, +9.2 pp pass@1 on pooled τ^2 -Bench, and +4.85 pp ToolHop Acc. We also verify static R/C/O contract transfer on ToolBH.

2 Related Work

Agentic planning and self-correction. Agentic planners connect reasoning to action through chain-of-thought, self-critique, trajectory search, multi-stage control, or learned clarification (Yao et al., 2023b; Wei et al., 2022; Shinn et al., 2023; Madaan et al., 2023; Yao et al., 2023a; Zhou et al., 2024; You et al., 2025; Zhang et al., 2025). These methods improve planner deliberation and the quality of proposed actions. RiskGuard is complementary: it keeps the base planner frozen and acts only after a candidate action has been proposed, auditing whether the accumulated trace supports that action before commitment.

Safety guards and pre-execution verification. Recent guards and verifiers simulate risky actions, inject constitutions, check safety policies, propagate information-flow labels, restrict tool-invocation graphs, or detect belief-misaligned actions (Ruan et al., 2024; Hua et al., 2024; Xiang et al., 2025; Chen et al., 2025; Wang et al., 2025; An et al., 2025; Jiang et al., 2026; Yuan et al., 2026; Fang et al., 2025). These systems provide complementary guarantees for injection robustness, policy compliance, information flow, tool availability, and action-level safety. RiskGuard additionally models trace-internal pre-execution risk propagation: it attributes R/C/O contamination from the accumulated trace, propagates typed risks toward effect boundaries, and routes risky candidates to clarification, constrained execution, probing, or safe fallback rather than allow/block alone. Closest to RiskGuard, AskToAct (Zhang et al., 2025) learns self-correcting clarification for ambiguous tool-use requests, while InferAct (Fang et al., 2025) detects intent-misaligned actions before execution using belief reasoning. RiskGuard differs by modeling how typed R/C/O contamination accumulates across the trace and propagates through multi-step dependencies toward effect boundaries.

Tool-use evaluation. Tool-use evaluation has moved from API retrieval and single-call generation (Li et al., 2023; Qin et al., 2024; Patil et al., 2025; Huang et al., 2024) to stateful dialogue, multi-hop tool use, plan-audit, and adversarial settings (Lu et al., 2025; Barres et al., 2025; Ye et al., 2025; Zhang et al., 2024; Debenedetti et al., 2024; Yuan et al., 2024). We evaluate RiskGuard on three long-horizon runtime benchmarks and report ToolBH as an offline plan-audit transfer; an easy-

dev-pool check verifies that the controller does not interfere with already-safe trajectories under the calibrated setting.

3 Methodology

3.1 Problem Setup and Action Space

We consider long-horizon tool use with a frozen base planner π_0 . At step t , the interaction history is a typed trace prefix $\tau_{\leq t}$ obtained from environment logs and tool-interface records, and the planner proposes a candidate tool call a_t . A tool call is an *effect-boundary* action when its effects are state-changing or externally visible and cannot be retracted—writes, sends, payments, cancellations, or updates to persistent state. Effect-boundary status is a property of the tool, fixed per tool rather than per scenario. Two failure properties shape the controller: stepwise legality motivates trace-level attribution beyond the candidate call, while semantic heterogeneity motivates graded risk severity, typed failure semantics, and mode-specific interventions.

RiskGuard inserts a deterministic rule between the planner and executor, $\phi(\tau_{\leq t}, a_t) \mapsto m_t \in \mathcal{M}$. The five modes ask for a missing user-resolvable binding (CLARIFY), commit unchanged (EXECUTE), commit a constrained safe template (CONSTRAINED), gather non-committing evidence (PROBE), or stop safely (FALLBACK). Only CLARIFY and PROBE consume explicit intervention costs $\kappa_{\text{clr}}, \kappa_{\text{prb}}$ from $\rho_t = \mathcal{R} - \text{Usage}(\tau_{\leq t})$; Appendix A.7 gives the success–resource objective, and Fig. 2 summarizes the three-stage controller.

3.2 Candidate-Augmented Hypergraph

Given $\tau_{\leq t}$ and a_t , we build a candidate-augmented dependency hypergraph

$$G_t^+ = (V_t, E_t, \mathcal{U}_t), \quad (1)$$

where the superscript $+$ marks augmentation with a_t ; all sets below are evaluated on G_t^+ . V_t contains execution-relevant trace units and a provisional node for a_t ; E_t contains typed data-flow and schema-derived effect-outlet edges; and $\mathcal{U}_t$ contains conjunctive prerequisite hyperedges. A hyperedge $(U, v) \in \mathcal{U}_t$ requires every prerequisite $u \in U$ to be satisfied before v commits, such as ownership plus confirmation or authentication plus capability checks. The boundary set $\mathcal{B}_t \subseteq V_t$ marks state-changing or externally visible calls, and G_t^+ is constructed deterministically from the typed trace

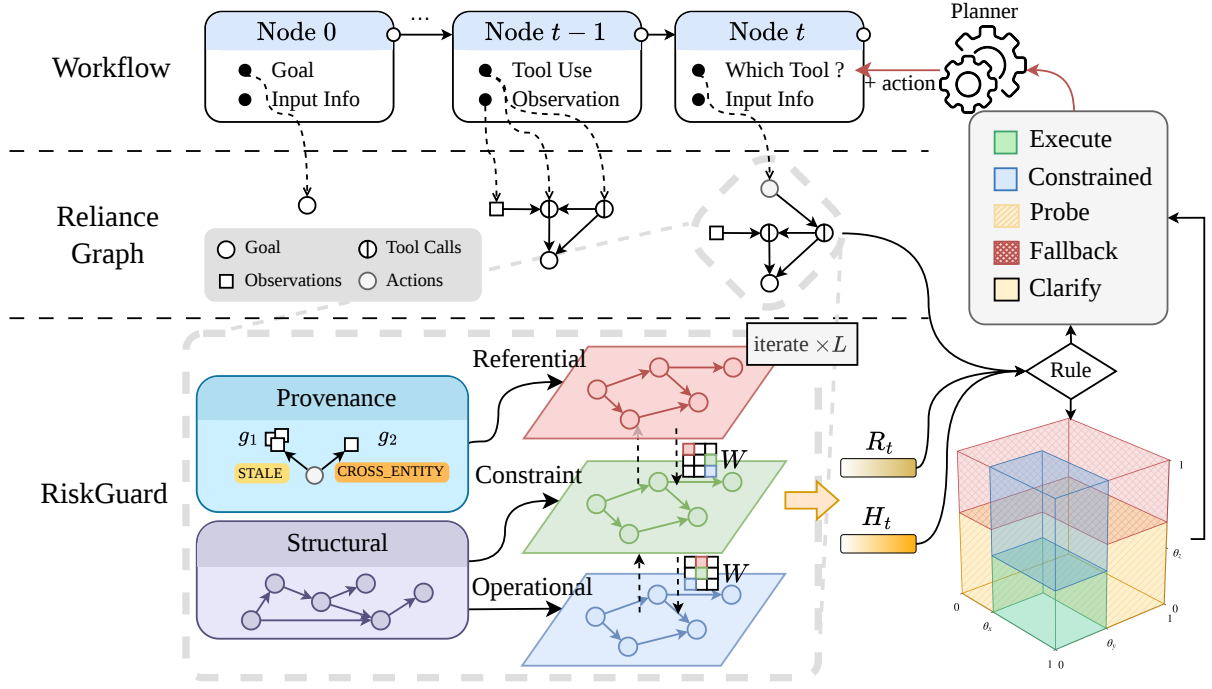


Figure 2: RiskGuard controller. For each candidate a_t , attribution inserts a provisional hypergraph node and assigns $R/C/O$ scores; diffusion combines cross-channel reach with smoothed channel fields into (R_t, H_t) over the unresolved frontier; the mode rule maps (x_t, y_t, z_t, ρ_t) to one of five modes before any effect-boundary commitment.

and tool-interface contract without additional LLM inference.

Integration metadata is adapter- or registry-level: schemas provide roles and effectfulness, the shared contract layer generates safe fallback or constrained templates, and provenance plus parameter overlap induces data-flow and boundary-proximity edges. Evidence hyperedges group co-referential arguments and tool results; in schema-poor read-only settings such as ToolHop, RiskGuard uses provenance tracking and non-committing chain-completion recovery over read-only calls.

Let $\sigma_t(u) \in [0, 1]$ be a soft satisfaction score and $r(v, \mathcal{B}_t) \in [0, 1]$ a bounded reach score from v to a candidate-reachable boundary. For $\mathcal{P}(a_t) = \{U : (U, a_t) \in \mathcal{U}_t\}$ (prerequisite groups for the candidate), we define a graded prerequisite-coverage score

$$c_t^{\text{pre}} = \frac{1}{|\mathcal{P}(a_t)|} \sum_{P \in \mathcal{P}(a_t)} \prod_{u \in P} \sigma_t(u), \quad (2)$$

with $c_t^{\text{pre}} = 1$ when $\mathcal{P}(a_t) = \emptyset$, treating c_t^{pre} as a graded coverage—not a logical permission condition.

The boundary proximity $b_t^{\text{crit}} := \max_{v \in \text{Desc}_{G_t^+}(a_t)} r(v, \mathcal{B}_t)$ is the strongest path from a_t to a boundary, and the candidate-

conditioned unresolved frontier $F_t(a_t) := \{v \in V_t : \sigma_t(v) < 1, v \in \text{Anc}_{G_t^+}(a_t) \cup \{a_t\}, r(v, \mathcal{B}_t) > 0\}$ is the set on which all global aggregation is performed. Reachability of a frontier node is computed through a_t and its declared effect outlets, so reach is conditioned on the candidate rather than reported as a global alarm, and Eq. (3) is well-defined on $F_t(a_t)$. A missing or unsatisfied required group is also injected into the operational channel O below, so it can block execution through the risk state.

Risk attribution decomposes pre-execution risk into three interacting failure semantics. At node v , R captures non-unique, stale, or conflicting entity bindings; C captures schema, user, policy, scope, or confirmation violations; and O captures unsatisfied prerequisites, runtime-state inconsistencies, or missing verification events.

The candidate a_t is scored under the same rule as any other node in V_t . The three channels are typed entry points rather than mutually exclusive labels: a referential defect at v can manifest as a constraint or operational defect downstream when the affected argument is consumed by a guarded tool, which motivates the cross-channel transport in Sec. 3.3.

For each critical argument we collect its support

set from prior trace units and assign one of four fixed verdicts (CONFIRMED, STALE, CROSSENTITY, UNGROUNDED); a fixed monotone table maps the verdicts to an evidential contamination score $q_t^{\text{evid}} \in [0, 1]$ that feeds the channel scores $q_v^R, q_v^C, q_v^O \in [0, 1]$ at v . We use UNGROUNDED for any argument that lacks a unique current binding, including both unsupported arguments and cases with multiple user-resolvable candidate bindings. Construction rules, schema-vocabulary coverage, the soft-satisfaction map, the verdict table, and the per-channel update are catalogued in Appendix A.

3.3 Risk Diffusion

A defect should affect the controller only if it can still reach an effect boundary and the trace still contains active evidence for it. Cross-channel reach estimates whether a defect can manifest at a boundary, while channel smoothing summarizes whether unresolved candidate-relevant evidence still supports the defect.

A failure can change type as it propagates: a stale entity reference (R) can become a constraint violation (C) when consumed by a guarded tool. We therefore index reach over two axes, a node axis along the dependency graph and a channel axis governed by a fixed transport kernel W_ℓ per edge type. With per-edge weight $w_{u \rightarrow v}$,

$$\begin{aligned} r_{u \rightarrow \beta}^{\alpha} &= 1 - \prod_{(u \rightarrow v, \ell) \in E_t} (1 - w_{u \rightarrow v} m_{v, \ell}^{\alpha \rightarrow \beta}), \\ m_{v, \ell}^{\alpha \rightarrow \beta} &:= \sum_{\gamma \in \{R, C, O\}} W_\ell[\alpha \rightarrow \gamma] r_v^{\gamma \rightarrow \beta}, \end{aligned} \quad (3)$$

with terminal clause $r_c^{\alpha \rightarrow \beta} = \mathbb{K}[\alpha = \beta]$ for $c \in \mathcal{B}_t$. The product acts as a bounded soft-union over successor paths rather than as an independence assumption over calibrated probabilities. Channel transport is tied to edge type rather than to individual tools; setting $W_\ell = I$ disables type conversion and recovers same-channel reach.

To incorporate currently observed evidence, we apply a smoothing operator to each channel over the normalized dependency adjacency:

$$p_t^{\alpha, (k+1)} = \eta (\tilde{A}_t p_t^{\alpha, (k)}) \odot (1 - \bar{\sigma}_t) + (1 - \eta) q_t^\alpha, \quad (4)$$

where $q_t^\alpha = (q_v^\alpha)_{v \in V_t}$, $\tilde{A}_t$ is row-normalized adjacency induced by G_t^+ , and $\bar{\sigma}_t[v] = \sigma_t(v)$ damps flow into resolved nodes. The teleport term anchors each node to its local evidence; we initialize

$p_t^{\alpha, (0)} = q_t^\alpha$ and iterate to the unique fixed point for $\eta < 1$.

Let p_v^α be the fixed-point value at node v and $\tilde{r}_v^\alpha := \max_\beta r_v^{\alpha \rightarrow \beta}$ the worst-case manifestation reach. We aggregate the propagated field over $F_t(a_t)$ as

$$\begin{aligned} R_t &= \max_{v \in F_t(a_t)} \max_\alpha p_v^\alpha \tilde{r}_v^\alpha, \\ H_t &= \frac{1}{\max(|F_t(a_t)|, 1)} \sum_{v \in F_t(a_t)} \sum_\alpha H_b(p_v^\alpha) \tilde{r}_v^\alpha, \end{aligned} \quad (5)$$

where $H_b(p) = -p \log_2 p - (1 - p) \log_2 (1 - p)$ is the binary entropy on $[0, 1]$ and both quantities vanish when $F_t(a_t) = \emptyset$. We use H_b as an entropy-shaped scoring function rather than as a calibrated probabilistic entropy: p_v^α is a contamination score in $[0, 1]$, not a posterior probability.

R_t uses $\max$ because a single boundary-reaching contaminated channel is enough to treat a_t as risky; H_t uses a length-normalized sum so that mid-confidence ambiguity scales with the share, not the count, of unresolved candidate-relevant nodes. Per-edge weights, kernel parameterization, cyclic-graph iteration, and the contraction proof for Eq. (4) are in Appendix A.

3.4 Five-Mode Selection Rule

We reduce the risk state to three decision variables: obligation closure $x_t := \min(c_t^{\text{pre}}, 1 - q_t^{\text{evid}})$, boundary risk $y_t := R_t + \lambda_H H_t$, and effect exposure

$$z_t := \lambda_{\text{eff}} \mathbb{K}[a_t \in \mathcal{B}_t] + \lambda_{\text{crit}} b_t^{\text{crit}},$$

with $\lambda_{\text{eff}} + \lambda_{\text{crit}} = 1$.

Thresholds $\theta_x^*, \theta_y^*, \theta_z^*$ form gates $\text{Rdy}_t = (x_t \geq \theta_x^*)$, $\text{LRisk}_t = (y_t \leq \theta_y^*)$, $\text{LExp}_t = (z_t \leq \theta_z^*)$, and three contract-derived admissibility predicates. Ask_t marks user-resolvable binding gaps, including missing support and multiple candidate bindings; Tpl_t marks bounded safe templates such as argument narrowing, confirmation preservation, read-before-write, or a declared safe option; and Recov_t marks risks reducible by non-committing read-only or verification actions.

We further define two paired admissibility gates,

$$\begin{aligned} \text{SafeExec}_t &:= \text{Rdy}_t \wedge \text{LRisk}_t \wedge \text{LExp}_t, \\ \text{SafeTpl}_t &:= \text{Rdy}_t \wedge \text{LRisk}_t \wedge \text{Tpl}_t, \end{aligned}$$

where SafeExec_t holds when the original candidate is committable as-is, and SafeTpl_t holds when a

contract-declared safe template is committable in its place.

We also use the probe-eligibility predicate $\text{NPrb}_t := (\neg \text{Rdy}_t \vee \neg \text{LRisk}_t) \wedge \text{Recov}_t$.

RiskGuard selects the first admissible mode:

$$m_t = \begin{cases} \text{CLARIFY} & \text{Ask}_t \wedge \neg \text{SafeExec}_t \wedge \rho_t \geq \kappa_{\text{clr}}, \\ \text{EXECUTE} & \text{SafeExec}_t, \\ \text{CONSTRAINED} & \neg \text{SafeExec}_t \wedge \text{SafeTpl}_t, \\ \text{PROBE} & \text{NPrb}_t \wedge \rho_t \geq \kappa_{\text{prb}}, \\ \text{FALLBACK} & \text{otherwise.} \end{cases} \quad (6)$$

The priority order resolves askable binding gaps first, executes only when all gates pass, substitutes a safe template when available, probes recoverable risks under budget, and otherwise falls back to safe non-execution. These modes are not additional planning rollouts; they restrict how the proposed candidate may proceed before commitment.

3.5 Calibration

RiskGuard uses no gradient-based optimization or backbone fine-tuning and keeps the base planner π_0 frozen. Its constants have two tiers.

Decision-layer constants—thresholds, aggregation weights, per-mode costs, and η —are fixed by lightweight dev calibration from standard starting values (e.g., all $\theta^*=0.5$, $\lambda=0.5$, $\eta=0.7$), verified on a small disjoint split, and frozen for all benchmarks. Structural constants—per-edge weights w_ℓ , kernels W_ℓ , retention factors, and node-type priors—are fixed before evaluation from tool-interface semantics and runtime conventions.

Appendix A.7 lists every constant and its tier; at runtime, decisions are deterministic given the typed trace and contract.

4 Experiments

We evaluate whether RiskGuard improves long-horizon tool-use reliability by controlling typed trace-level risk before irreversible commitments. We ask four questions: (i) does RiskGuard improve runtime success under controlled budgets? (ii) do gains concentrate on deeper traces with more accumulated dependencies? (iii) which components of typed risk diffusion drive the gains? and (iv) does the controller avoid over-intervention on easy cases? We answer through main results (§4.2), component ablations (§4.3), and diagnostic analyses (§4.4).

4.1 Experimental Setup

Long-horizon slices. We evaluate three long-horizon runtime benchmarks—ToolSandbox-hard, τ^2 -Bench, and ToolHop 4–7-hop—and one offline plan-audit transfer, ToolBH. ToolSandbox-hard (Lu et al., 2025) ($n=67$, Sim) follows a deterministic hard cut: scenarios with official function-calling similarity < 0.8 and at least one multi-tool dependency edge. τ^2 -Bench (Barres et al., 2025) contributes Telecom ≥ 4 faults ($n=63$) and Retail ≥ 4 actions ($n=68$), pooled at the (domain, scenario) level for $n=131$ pass@1; both share protocol, backbone, user simulator, and metric, which makes the pooled paired bootstrap admissible under the official protocol. ToolHop 4–7-hop (Ye et al., 2025) ($n=103$, Acc) instantiates dense prerequisite chains. ToolBH (Zhang et al., 2024) ($n=190$, L2/L3-U) is an offline plan-audit benchmark that does not exercise the runtime loop and is reported only as a transfer of the static $R/C/O$ contract.

Baselines. All baselines share the same backbone and tool inventory. Vanilla FC has no intervention. Prompting baselines ReAct (Yao et al., 2023b) and Reflexion (Shinn et al., 2023) strengthen planner deliberation. LATS (Zhou et al., 2024) adds tree search, evaluated on ToolBH as its natural offline plan-audit setting where tree expansion and roll-back are well defined over static candidate plans. SafeHarbor (Liu et al., 2026), GuardAgent (Xiang et al., 2025), ShieldAgent (Chen et al., 2025), and Rule-based are runtime guardrails organized around block-or-execute decisions.

Protocol and backbones. All rows use gpt-4.1-mini at temperature 0; RiskGuard rows use the frozen dev-calibrated controller in Sec. 3.5. Interactive slices use matched step budget $K=80$, except ReAct on τ^2 -Telecom uses $K=120$ to offset its two-call protocol; cells are three-seed means over identical scenario IDs. Full benchmark, baseline, and reproducibility details are in Appendices B.2–B.5.

4.2 Main Results

Overall. RiskGuard improves the primary metric on every long-horizon runtime slice. ToolSandbox-hard rises by +0.133 Sim on $[0, 1]$ (paired bootstrap 95% CI $[+0.075, +0.200]$, +47% vs. Vanilla). The pooled τ^2 -Bench cell gains +9.2 pts p@1 (CI $[+0.8, +17.6]$, two-sided $p = 0.041$). The single-domain CIs include zero because

Method	ToolSandbox-hard		τ^2 -Tel. $\geq 4F$		τ^2 -Ret. $\geq 4A$		τ^2 -Pooled		ToolHop 4-7 Hop	
	Sim $\uparrow$	Turn $\downarrow$	p@1 $\uparrow$	Cost $\downarrow$	p@1 $\uparrow$	Cost $\downarrow$	p@1 $\uparrow$	Cost $\downarrow$	Acc $\uparrow$	Turn $\downarrow$
Vanilla FC	0.281	15.73	<u>23.8</u>	<u>0.081</u>	47.1	0.027	35.9	<u>0.053</u>	40.78	<u>4.50</u>
ReAct	0.291	15.95	19.0	0.133	37.1	0.026	28.4	0.077	41.75	4.95
Reflexion	0.304	16.42	20.6	0.102	50.0	0.029	35.9	0.064	40.78	5.58
Rule-based	0.271	15.28	17.5	0.089	52.1	<u>0.025</u>	35.5	0.056	39.81	4.28
GuardAgent	0.282	15.68	21.8	0.098	42.1	0.026	32.3	0.061	40.78	4.72
ShieldAgent	0.283	15.74	15.9	0.105	32.4	0.021	24.4	0.061	37.86	4.68
SafeHarbor	<u>0.373</u>	<u>14.96</u>	15.9	0.091	<u>54.4</u>	0.027	<u>37.4</u>	0.058	<u>42.72</u>	4.86
RiskGuard (ours)	0.414*	14.81	33.3	0.076	55.9	0.030	45.0*	0.052	45.63	7.37
Δ_{rel} vs. Vanilla	+47%	-5.9%	+40%	-6.2%	+19%	+11%	+26%	-1.9%	+12%	+64%

Table 1: Main runtime results across three long-horizon benchmark families, reported as three-seed means. The τ^2 -Pooled column merges Telecom $\geq 4F$ and Retail $\geq 4A$ at the scenario level ($n=131$). Resource proxies are slice-local: turns for ToolSandbox/ToolHop and agent USD cost for τ^2 -Bench. * denotes paired-bootstrap 95% CI excluding zero. Cost-normalized τ^2 -Pooled p@1/USD: Vanilla 35.9/0.053=677, RiskGuard 45.0/0.052=865 (+28%); ToolHop turn-budget control is in Appendix C.4. ToolHop Acc uses the official ToolHop evaluator on the reported 4-7-hop subset; corrected diagnostic scores on the same trajectories are reported in Appendix C.7.

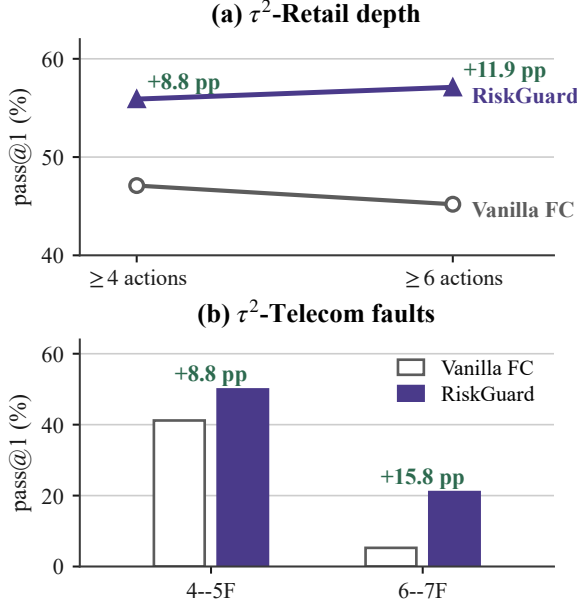


Figure 3: Task-depth diagnostics on τ^2 -Retail and τ^2 -Telecom. Retail curves compare Vanilla FC with RiskGuard as required actions increase; Telecom bars compare p@1 across injected-fault-count bands.

each domain has at most 68 evaluation scenarios (Appendix B.6). ToolHop rises by +4.85 pts Acc under the official evaluator (40.78→45.63, +12%). The corrected diagnostic ToolHop evaluator gives +8.74 pts Acc (38.83→47.57, +23%; Appendix C.7), and the pooled τ^2 -Bench cell improves p@1 per USD by +28% (Table 1).

Task-depth diagnostics. Gains concentrate on deeper traces (Fig. 3). On τ^2 -Retail, tightening the action-count cut from $\geq 4A$ to $\geq 6A$ raises the

absolute p@1 gain from +8.8 to +11.9 pts. On τ^2 -Telecom, grouping injected faults into 4-5F and 6-7F bands gives gains of +8.8 and +15.8 pp, respectively; these within-domain trends support the pooled headline despite sample-size-limited per-domain CIs (Appendix B.6).

Baseline comparison. Prompting methods add deliberation but no typed pre-execution representation, and on τ^2 -Telecom they sit 3.2-4.8 pts below Vanilla. Block-or-execute guardrails are brittle when the safer response is clarification, constrained execution, or prerequisite probing; ShieldAgent trails Vanilla by 14.7 pts on Retail $\geq 4A$. SafeHarbor, the strongest non-RiskGuard baseline on ToolSandbox-hard, reaches 0.373 Sim but remains 0.041 Sim below RiskGuard. Its improvement does not transfer to τ^2 -Telecom, where it scores 15.9 p@1 versus 23.8 for Vanilla FC, consistent with a policy-compliance-oriented boundary mechanism rather than trace-level dependency propagation.

Offline transfer. ToolBH does not execute tools or return observations; we use it only to test whether the static $R/C/O$ contract transfers to plan-audit (Table 2). With the runtime loop inactive, the static contract reaches L2/L3-U progress rates of 0.515/0.617, above LATS (0.401/0.458) and GuardAgent (0.451/0.508), at lower tokens than both; Appendices C.1 and C.6 give implementation details and subtask breakdowns.

Resource trade-off. On ToolHop, turns rise 4.50→7.37 (+64%). Under the corrected diagnostic evaluator, a longer-budget Vanilla control

Method	ToolBH-Unsolvable		
	L2-U $\uparrow$	L3-U $\uparrow$	Tok $\downarrow$
Vanilla FC	0.371	0.428	3,401
ReAct	0.391	0.448	3,911
Reflexion	0.411	0.468	4,081
LATS	0.401	0.458	11,902
GuardAgent	0.451	0.508	5,101
ShieldAgent	0.441	0.498	5,101
Rule-based	0.401	0.458	3,401
RiskGuard (ours)†	0.515*	0.617*	4,604

Table 2: Offline $R/C/O$ contract transfer on ToolBH (runtime loop not exercised). † marks the static-contract variant; * marks paired-bootstrap CI excluding zero.

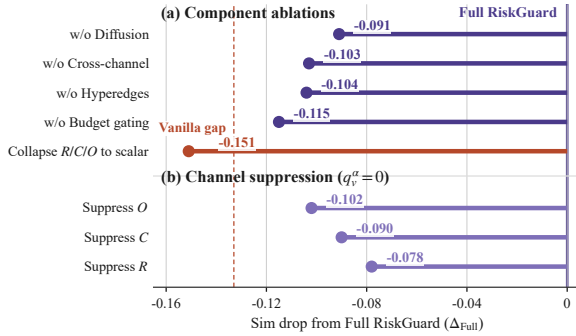


Figure 4: ToolSandbox-hard ablations ($n=67$, three-seed paired). Bars plot $\Delta_{\text{Full}} = \text{Sim}_{\text{ablated}} - \text{Sim}_{\text{Full}}$ (Full Sim=0.414); the dotted line marks the Vanilla horizon (-0.133). Panel (a) isolates each diffusion subsystem; panel (b) suppresses one q_v^α source at a time. Numeric Sim/Turn rows are in Table 10.

at $K=125$ closes only about one third of the RiskGuard gain (+2.92 pts vs. +8.74 pts), indicating that the gain is not attributable to extended horizon alone (Appendix C.4). Other slices are cost-neutral or cheaper except for a \$0.003 Retail increase. Together, the cost pattern localizes the main overhead to ToolHop prerequisite materialization rather than broad over-intervention.

4.3 Ablation Study

We run component ablations on ToolSandbox-hard because it exercises the full runtime loop and all three intervention channels.

Hypergraph and cross-channel diffusion. Each diffusion subsystem in Fig. 4 contributes a distinguishable share of the 0.133-Sim Vanilla gap, consistent across three independent seeds.

Three-channel diagnostic. Fig. 4b shows that suppressing any channel causes a substantial drop, with O largest. Because O contains unsatisfied prerequisites and runtime-state inconsistencies, its larger drop is consistent with the PROBE-heavy

Variant	Tel. $\geq 4F$	Ret. $\geq 4A$	Pooled τ^2
Full RiskGuard	33.33%	55.88%	45.04%
w/o CLARIFY	23.81%	45.59%	35.11%
w/o CONSTRAINED	26.98%	41.18%	34.35%
w/o PROBE	1.59%	2.94%	2.29%

Table 3: Three-seed ablation results of disabling individual decision-layer intervention modes on τ^2 -Bench.

Backbone	V	RiskGuard	Δ	IV
GPT-4.1-mini	.281	.414	+0.133	87
Claude-Haiku-4.5	.329	.362	+0.032	12
DeepSeek-V3.1	.428	.466	+0.037	32
Qwen3-32B	.165	.250	+0.085	28

Table 4: Cross-model ToolSandbox-hard results ($n=67$, identical scenario IDs, three-seed paired). V is Vanilla FC, Δ is the absolute Sim gain, and IV is the number of recorded controller-mode events. The controller and user simulator are frozen at the GPT-4.1-mini anchor.

behavior observed on hard traces. The nontrivial R/C drops show that grounding and constraint semantics remain active diffusion inputs, not merely labels used after a scalar risk score is computed.

Decision-layer modes. Table 3 separates the contribution of each intervention mode on pooled τ^2 -Bench. Removing CLARIFY or CONSTRAINED reduces pooled p@1 by roughly 10 points, while removing PROBE collapses success to 2.29%. This indicates that non-committing read-only checks for entity state, ownership, order status, or prior evidence are the load-bearing recovery mechanism for long-horizon dialogue tasks. Routing-vs-input details are in Appendix B.8.

4.4 Analysis

Controller selectivity. On the 67 ToolSandbox-hard scenarios over three seeds, the event log contains 87 recorded controller-mode events: approximately 70 PROBE (80%), 8 FALLBACK (9%), 5 allowed EXECUTE decisions (6%), 3 CLARIFY decisions (4%), and 1 CONSTRAINED decision (1%). The profile is workload-specific rather than a globally aggressive policy: on the ToolSandbox easy dev pool ($n=66$), Vanilla and RiskGuard both reach Sim=1.000, and no intervention fires. Thus, the controller preserves already-safe trajectories while concentrating non-committing probes on traces with missing operational evidence.

Cross-model transfer. Table 4 shows positive paired gains on all four backbones without threshold retuning. Excluding the GPT-4.1-mini anchor’s

Failure mode	Vanilla	RiskGuard	Ablated
Ungrounded args.	.000	1.000	.111 ($-R$)
Vague reference	.333	1.000	.500 ($-C$)
Abandoned prereq.	.486	.933	.517 ($-O$)

Table 5: Representative ToolSandbox-hard case outcomes (Sim, three-seed means). “Ablated” removes the corresponding channel and eliminates most of the repair gain.

+0.133 gain, the three transfer-backbone gains span +0.032 to +0.085 Sim, while turns fall by 1.5–3.2%. Recorded controller-mode events remain PROBE-dominant ($\geq 75\%$), matching the anchor profile. On the pooled τ^2 -Bench cell, Claude-Haiku-4.5 and Qwen3-32B improve p@1 by +20.9 and +22.1 points, respectively (Appendix B.9). These results isolate transfer of the typed-trace controller from backbone-specific adaptation.

Budget and latency controls. The ToolHop gain is not explained by a larger interaction budget alone. Increasing Vanilla’s turn cap from 9 to 125 raises corrected accuracy only from 38.83 to 41.75 (+2.92 points), whereas RiskGuard reaches 47.57 (+8.74 points over the original control). The residual gain over Vanilla@ $K=125$ is +5.82 points, so only about one third of the improvement is attributable to a longer horizon; the remainder is consistent with PROBE materializing prerequisite hops that Vanilla skips. On ToolSandbox, mean wall-clock time rises from 13.50 to 21.32 seconds per scenario, but subtracting median probe-call latency leaves less than 50 ms of structural computation per decision.

Case-wise repair pathways. Table 5 connects the aggregate gains to the three typed channels. For ungrounded search coordinates, the referential channel rejects fabricated latitude/longitude arguments and CLARIFY obtains a grounded location before the second call. For a vague “next reminder” instruction, the constraint channel detects that the selected entity has no provenance and CONSTRAINED first resolves the current schedule. For GPS access under disabled location services and low-battery mode, the operational channel identifies repairable prerequisites; PROBE reads device state, enables the required services, and only then commits the location call. The corresponding $-R$, $-C$, and $-O$ ablations sharply reduce Sim, showing that these modes implement distinct repairs rather than cosmetic labels on a shared scalar

score.

5 Conclusion

RiskGuard defines long-horizon tool use as a pre-execution risk-control problem: before committing a proposed call, the controller verifies grounding, constraints, and operational prerequisites against the typed trace and propagates typed contamination through the dependency hypergraph toward irreversible effects. A three-channel risk state, cross-channel diffusion, and budget-aware mode selection give the frozen runtime controller enough structure to repair failures that prompting, search, and block-or-execute guardrails leave unresolved. Runtime claims are supported by three long-horizon slices on which it improves the primary metric while making resource trade-offs explicit; ToolBH provides an auxiliary offline check that the static $R/C/O$ contract is also useful when the runtime loop does not apply. Composing RiskGuard with retrieval-side verifiers, dynamic tool catalogs, and multi-agent settings are natural extensions of the typed-trace interface.

Limitations

RiskGuard assumes typed traces and a registry-level tool-interface contract that exposes prerequisite, confirmation, ownership, and capability information. This is an integration cost shared per tool registry. When such metadata is absent, the hyperedge primitive falls back to typed data-flow control (Appendix A.3); on ToolHop, $\mathcal{U}_t^+ = \emptyset$ and gains come from PROBE-mode prerequisite recovery rather than hyperedge diffusion. Incorrect prerequisite annotations could shift the controller’s O -channel behavior. RiskGuard addresses decision-time risk after evidence has entered the trace, but does not guarantee retrieval-side correctness, tool-observation truthfulness, or external database validity. Defending against adversarial prompt injections falls under a separate threat model, while achieving absolute success under ultra-deep dependencies remains an open challenge. Following the official τ^2 -Bench protocol, the main runs use the same backbone for the agent and user simulator; to probe this same-model bias, Appendix B.9 varies the agent backbone while keeping the user simulator and controller frozen.

Acknowledgments

This work was supported by the National Defense Basic Scientific Research Program of China.

References

- Hengyu An, Jinghui Zhang, Tianyu Du, Chunyi Zhou, Qingming Li, Tao Lin, and Shouling Ji. 2025. [IPI-Guard: A novel tool dependency graph-based defense against indirect prompt injection in LLM agents](#). In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.
- Victor Barres, Honghua Dong, Soham Ray, Xujie Si, and Karthik Narasimhan. 2025. [\$\tau^2\$ -bench: Evaluating conversational agents in a dual-control environment](#). *Preprint*, arXiv:2506.07982.
- Zhaorun Chen, Mintong Kang, and Bo Li. 2025. [Shield-Agent: Shielding agents via verifiable safety policy reasoning](#). In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pages 8313–8344. PMLR.
- Edoardo DeBenedetti, Jie Zhang, Mislav Balunovic, Luca Beurer-Kellner, Marc Fischer, and Florian Tramèr. 2024. [Agentdojo: A dynamic environment to evaluate prompt injection attacks and defenses for llm agents](#). In *Advances in Neural Information Processing Systems*, volume 37, pages 82895–82920. Curran Associates, Inc.
- Haishuo Fang, Xiaodan Zhu, and Iryna Gurevych. 2025. [Preemptive detection and correction of misaligned actions in LLM agents](#). In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.
- Wenyue Hua, Xianjun Yang, Mingyu Jin, Zelong Li, Wei Cheng, Ruixiang Tang, and Yongfeng Zhang. 2024. [TrustAgent: Towards safe and trustworthy LLM-based agents](#). *Preprint*, arXiv:2402.01586.
- Yue Huang, Jiawen Shi, Yuan Li, Chenrui Fan, Siyuan Wu, Qihui Zhang, Yixin Liu, Pan Zhou, Yao Wan, Neil Zhenqiang Gong, and Lichao Sun. 2024. [Meta-Tool benchmark for large language models: Deciding whether to use tools and which to use](#). In *The Twelfth International Conference on Learning Representations*.
- Yan Jiang, Hao Zhou, Lizhong GU, Tianlong Li, Ruinan Jin, Wanqi Zhou, and Ai Han. 2026. [NaviAgent: Graph-driven bilevel planning for scalable tool orchestration](#). *Preprint*, arXiv:2506.19500.
- Minghao Li, Yingxiu Zhao, Bowen Yu, Feifan Song, Hangyu Li, Haiyang Yu, Zhoujun Li, Fei Huang, and Yongbin Li. 2023. [API-bank: A comprehensive benchmark for tool-augmented LLMs](#). In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 3102–3116, Singapore. Association for Computational Linguistics.
- Zhe Liu, Zonghao Ying, Wenxin Zhang, Quanchen Zou, Deyue Zhang, Dongdong Yang, Xiangzheng Zhang, and Hao Peng. 2026. [SafeHarbor: Defining precise decision boundaries via hierarchical memory-augmented guardrail for LLM agent safety](#). In *Proceedings of the 43rd International Conference on Machine Learning*, volume 306 of *Proceedings of Machine Learning Research*. PMLR.
- Jiarui Lu, Thomas Holleis, Yizhe Zhang, Bernhard Aumayer, Feng Nan, Haoping Bai, Shuang Ma, Shen Ma, Mengyu Li, Guoli Yin, Zirui Wang, and Ruoming Pang. 2025. [ToolSandbox: A stateful, conversational, interactive evaluation benchmark for LLM tool use capabilities](#). In *Findings of the Association for Computational Linguistics: NAACL 2025*, pages 1160–1183, Albuquerque, New Mexico. Association for Computational Linguistics.
- Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegrefe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. 2023. [Self-Refine: Iterative refinement with self-feedback](#). In *Advances in Neural Information Processing Systems*, volume 36.
- Shishir G. Patil, Huanzhi Mao, Fanjia Yan, Charlie Cheng-Jie Ji, Vishnu Suresh, Ion Stoica, and Joseph E. Gonzalez. 2025. [The Berkeley Function Calling Leaderboard \(BFCL\): From tool use to agentic evaluation of large language models](#). In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pages 48371–48392. PMLR.
- 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 *International Conference on Learning Representations*.
- Yangjun Ruan, Honghua Dong, Andrew Wang, Silviu Pitis, Yongchao Zhou, Jimmy Ba, Yann Dubois, Chris J. Maddison, and Tatsunori Hashimoto. 2024. [Identifying the risks of LM agents with an LM-emulated sandbox](#). In *The Twelfth International Conference on Learning Representations*.
- Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. 2023. [Reflexion: language agents with verbal reinforcement learning](#). In *Advances in Neural Information Processing Systems*, volume 36, pages 8634–8652. Curran Associates, Inc.

- Peiran Wang, Yang Liu, Yunfei Lu, Yifeng Cai, Hongbo Chen, Qingyou Yang, Jie Zhang, Jue Hong, and Ye Wu. 2025. [AgentArmor: Enforcing program analysis on agent runtime trace to defend against prompt injection](#). *Preprint*, arXiv:2508.01249.
- Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. 2022. [Chain-of-thought prompting elicits reasoning in large language models](#). In *Advances in Neural Information Processing Systems*, volume 35.
- Zhen Xiang, Linzhi Zheng, Yanjie Li, Junyuan Hong, Qinbin Li, Han Xie, Jiawei Zhang, Zidi Xiong, Chulin Xie, Carl Yang, Dawn Song, and Bo Li. 2025. [GuardAgent: Safeguard LLM agents by a guard agent via knowledge-enabled reasoning](#). *Preprint*, arXiv:2406.09187.
- Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. 2023a. [Tree of thoughts: Deliberate problem solving with large language models](#). In *Advances in Neural Information Processing Systems*, volume 36.
- Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R Narasimhan, and Yuan Cao. 2023b. [ReAct: Synergizing reasoning and acting in language models](#). In *The Eleventh International Conference on Learning Representations*.
- Junjie Ye, Zhengyin Du, Xuesong Yao, Weijian Lin, Yufei Xu, Zehui Chen, Zaiyuan Wang, Sining Zhu, Zhiheng Xi, Siyu Yuan, Tao Gui, Qi Zhang, Xu-anjing Huang, and Jiecao Chen. 2025. [ToolHop: A query-driven benchmark for evaluating large language models in multi-hop tool use](#). In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2995–3021, Vienna, Austria. Association for Computational Linguistics.
- Ziming You, Yumiao Zhang, Dexuan Xu, Yiwei Lou, Yandong Yan, Wei Wang, Huaming Zhang, and Yu Huang. 2025. [DatawiseAgent: A notebook-centric LLM agent framework for adaptive and robust data science automation](#). *Preprint*, arXiv:2503.07044.
- Aojie Yuan, Zhiyuan Su, and Yue Zhao. 2026. [AEGIS: No tool call left unchecked—a pre-execution firewall and audit layer for AI agents](#). *Preprint*, arXiv:2603.12621.
- Tongxin Yuan, Zhiwei He, Lingzhong Dong, Yiming Wang, Ruijie Zhao, Tian Xia, Lizhen Xu, Binglin Zhou, Fangqi Li, Zhuosheng Zhang, Rui Wang, and Gongshen Liu. 2024. [R-judge: Benchmarking safety risk awareness for LLM agents](#). In *Findings of the Association for Computational Linguistics: EMNLP 2024*. Association for Computational Linguistics.
- Xuan Zhang, Yongliang Shen, Zhe Zheng, Linjuan Wu, Wenqi Zhang, Yuchen Yan, Qiuying Peng, Jun Wang, and Weiming Lu. 2025. [AskToAct: Enhancing LLMs tool use via self-correcting clarification](#). In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.
- Yuxiang Zhang, Jing Chen, Junjie Wang, Yaxin Liu, Cheng Yang, Chufan Shi, Xinyu Zhu, Zihao Lin, Hanwen Wan, Yujiu Yang, Tetsuya Sakai, Tian Feng, and Hayato Yamana. 2024. [ToolBeHonest: A multi-level hallucination diagnostic benchmark for tool-augmented large language models](#). *Preprint*, arXiv:2406.20015.
- Andy Zhou, Kai Yan, Michal Shlapentokh-Rothman, Haohan Wang, and Yu-Xiong Wang. 2024. [Language agent tree search unifies reasoning, acting, and planning in language models](#). In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 62138–62160. PMLR.

A Method Reference

A.1 Notation and Pseudocode

Table 6 lists the composite symbols used by the main text; atomic symbols ($\tau_{\leq t}$, a_t , σ_t , q_v^α , $w_{u \rightarrow v}$, W_ℓ , etc.) are defined where they first appear in §3 and not duplicated here. Throughout the result tables, **bold** / underline mark the best / second-best entry per metric column; * marks paired-bootstrap CIs that exclude zero (Appendix B.6); † marks the offline plan-audit setting in Table 2. The pre-execution stage then runs the five-step procedure below.

Symbol	Role
G_t^+	Candidate-augmented dependency hypergraph (§3.2).
$F_t(a_t)$	Candidate-conditioned unresolved frontier; the aggregation domain.
$c_t^{\text{pre}}, b_t^{\text{crit}}$	Prerequisite coverage and boundary proximity for a_t .
R_t, H_t	Boundary severity and ambiguity mass over $F_t(a_t)$.
x_t, y_t, z_t	Closure, boundary risk, and effect exposure (§3.4).
SafeExec _t	Safe-execution gate; admits EXECUTE.
SafeTpl _t	Safe-template gate; admits CONSTRAINED.
$m_t \in \mathcal{M}$	Selected intervention mode (Eq. 6).

Table 6: Composite symbols used in the main text.

For each candidate a_t , RiskGuard (i) builds G_t^+ from $\tau_{\leq t}$, (ii) computes $c_t^{\text{pre}}, b_t^{\text{crit}}, F_t(a_t)$ and the four-class provenance verdicts on critical arguments, (iii) maps signals to $q_v^{R/C/O}$ via Eq. (7) and applies cross-channel reach (Eq. (3)) and channel smoothing (Eq. (4)) to obtain p_v^α , (iv) aggregates (R_t, H_t) over $F_t(a_t)$ and forms (x_t, y_t, z_t) , and (v) selects m_t by Eq. (6). All steps are deterministic functions of the typed trace and tool-interface contract.

A.2 Dependency Hypergraph Construction

Nodes. A trace element $u_i \in \tau_{\leq t}$ enters V_t when it changes external state, introduces a value later consumed, verifies/invalidates a prerequisite, or blocks/enables a pending commitment; the candidate a_t is added as a provisional node together with its schema-derived downstream effect nodes.

Edges. A pairwise edge $(u, v) \in E_t$ is added when v consumes a field written by u , references an entity introduced by u , or uses u as a validation witness. A hyperedge $(U, v) \in \mathcal{U}_t$ encodes a joint prerequisite group whose every element must hold before v may commit.

Prerequisite groups. $\mathcal{U}_t$ is read from four schema fields treated as a closed vocabulary: `auth_required`, `confirmation_required`, `precondition_calls`, and `capability_check`. Each declaration emits one hyperedge whose witness set $U_G \subseteq V_t$ contains the latest valid trace witness; an absent witness is inserted as a $\sigma_t=0$ sentinel so the prerequisite is marked unsatisfied rather than vacuous. When a_t declares no group, $\mathcal{P}(a_t) = \emptyset$ and $c_t^{\text{pre}}=1$ by Eq. (2).

A.3 Schema-Vocabulary Coverage

		auth required	confirmation required	precondition calls	capability check
ToolSandbox	$ \mathcal{T} =34$	✓	✓	✓	+
τ^2 -Telecom	$ \mathcal{T} =13$	✓	✓	✓	✓
τ^2 -Retail	$ \mathcal{T} =16$	✓	✓	✓	✓
ToolHop	$ \mathcal{T} =178$	—	—	—	—
ToolBH	$ \mathcal{T} \approx 70$	—	—	—	—

✓ Native (declared in official schema) + Single registry-side rule
 — Read-only registry ($\mathcal{P}(a_t)=\emptyset$)

Figure 5: Schema-vocabulary coverage per registry across the four schema fields used by hyperedge construction. Native = declared in the official schema; + = a single registry-side rule (not per-scenario); — = read-only registry ($\mathcal{P}(a_t)=\emptyset$ everywhere), in which case $\mathcal{U}_t=\emptyset$ and Eq. (2) reduces to flat data-flow control; the hyperedge ablation in Table 10 quantifies this regime.

Figure 5 details the schema-vocabulary coverage across the evaluated benchmark registries. The four fields are declared per tool, not per scenario, so the integration cost scales with $|\mathcal{T}|$ rather than with task volume. The only registry that required author-added coverage in this paper is ToolSandbox, where one registry-side rule fills `capability_check` for the entire registry.

A.4 Soft Satisfaction, Provenance, and Channel Estimation

Soft satisfaction σ_t . $\sigma_t(u)=1$ on a successful tool call or a current observation; $\sigma_t(u)=0$ on a pending or hard-error event; $\sigma_t(u) \in (0, 1)$ for retryable errors (0.5) and stale observations (0.3). Effect-carrying nodes reach $\sigma_t=1$ only after both commit and post-condition observation.

Provenance verdicts. For each critical argument $g \in \mathcal{G}(a_t)$, let $\mathcal{S}_t(g)$ be the support set under exact reference, typed alias, or validated derivation. The verdict $e_t(g)$ is computed by deterministic

precedence: CONFIRMED (unique current support consistent with the active entity context), STALE (support superseded by a later write), CROSSENTITY (best support for a different entity), UNGROUNDED (no traceable source, or no unique user-resolvable binding among multiple candidate entities). Verdicts are mapped to local contamination by a fixed monotone table: $q_g=0$ for CONFIRMED, $\rho_{\text{stale}} < \rho_{\text{cross}} < \rho_{\text{unk}} \leq 1$ for the others. The action-level evidential contamination is the noisy-OR over evidence groups, $q_t^{\text{evid}} = \max_h (1 - \prod_{g \in h} (1 - q_g))$.

Channel contamination update. Each node $v \in V_t$ has type $\text{typ}(v)$ (tool call, observation, verification event, effect-carrying action) and a fixed prior $\pi_{\text{typ}}^\alpha \in (0, 1)$. The signal s_v^α aggregates trace evidence per channel: $s_v^R = q_t^{\text{evid}}$ when v corresponds to a candidate carrying provenance-scored arguments; s_v^C is a noisy-OR over schema, type, range, capability, and bounded-reference violations attached to v ; s_v^O is a noisy-OR over failed prerequisite checks, prior-tool errors, rate-limit warnings, and session-state inconsistencies. The channel update is the standard log-odds form

$$q_v^\alpha = \sigma(\text{logit}(\pi_{\text{typ}(v)}^\alpha) + \gamma_\alpha s_v^\alpha), \quad (7)$$

with σ the logistic sigmoid and $\gamma_\alpha > 0$ a fixed channel-specific sensitivity; priors and sensitivities are task-independent constants.

A.5 Reachability Recurrences and Edge Weights

The dual-index reach $r_v^{\alpha \rightarrow \beta}$ in Eq. (3) reduces to the scalar reach $r(u, \mathcal{B}_t) = 1 - \prod_{(u,v)} (1 - w_{u \rightarrow v} r(v, \mathcal{B}_t))$ when $W_\ell = I$ for all ℓ , with terminal $r(c, \mathcal{B}_t) = 1$ for $c \in \mathcal{B}_t$. On acyclic G_t^+ a single reverse sweep from $\mathcal{B}_t$ solves it in $O(|E_t|)$; on cyclic graphs the same sweep is a monotone bounded fixed-point iteration. Edge weights are tied to edge type: prereq 1.00, data_dep 0.90, validation 0.80, alias 0.70, soft_evid 0.60.

A.6 Cross-Channel Transmission Kernels and Diffusion

Each edge type carries a row-stochastic 3×3 kernel W_ℓ parameterized by a single retention coefficient ρ_ℓ : $W_\ell[\alpha, \alpha] = \rho_\ell$, off-diagonal entries $\epsilon_\ell/2 = (1 - \rho_\ell)/2$. We use ρ_ℓ values prereq 0.80, validation 0.85, data_dep 0.70, soft_evid 0.70, alias 0.60 (alias leaks the

most because typed aliases couple R and C by construction; validation the least because witnesses are channel-targeted). Setting $\rho_\ell = 1$ recovers same-channel reach; $\rho_\ell = \frac{1}{3}$ gives uniform mixing.

The smoothing fixed point of Eq. (4) exists and is unique because the operator $T_\alpha(p) = \eta(\tilde{A}_t p) \odot (1 - \bar{\sigma}_t) + (1 - \eta)q_t^\alpha$ is an η -contraction in ℓ_∞ : $\|T_\alpha(p) - T_\alpha(p')\|_\infty \leq \eta \|\tilde{A}_t(p - p')\|_\infty \leq \eta \|p - p'\|_\infty$ since $\tilde{A}_t$ is row-normalized and $(1 - \bar{\sigma}_t)_i \in [0, 1]$. Banach’s theorem gives a unique fixed point in $[0, 1]^{|V_t|}$, reached at linear rate η . We project the hypergraph into $\tilde{A}_t$ by adding $1/|U|$ from each prerequisite witness in a hyperedge (U, v) on top of the pairwise adjacency, then row-normalize.

A.7 Budget-Aware Intervention and Calibration

Design criterion. With π_0 frozen and $\pi = \pi_0 \circ \phi$, we want

$$\max_{\phi} \mathbb{E}_\pi [\text{Succ}(\tau)] \quad \text{s.t.} \quad \text{Usage}(\tau) \leq \mathcal{R}, \quad (8)$$

where $\text{Usage}(\tau) = \sum_t [c^{\text{exec}}(a_t) + c^{\text{interv}}(m_t)]$ counts tool and intervention costs. Stepwise, RiskGuard estimates (R_t, H_t) from $\tau_{\leq t}$ and selects the least-cost admissible mode under remaining budget $\rho_t = \mathcal{R} - \text{Usage}(\tau_{\leq t})$.

Calibration: two-tier partition. Constants partition into two tiers (Table 7). *Tier 1* (dev-calibrated decision constants) contains the nine scalars $\theta = (\theta_x^*, \theta_y^*, \theta_z^*, \lambda_H, \lambda_{\text{eff}}, \lambda_{\text{crit}}, \kappa_{\text{prb}}, \kappa_{\text{clr}}, \eta)$. We use lightweight dev calibration to fix a standard configuration: $\theta_x^* = \theta_y^* = \theta_z^* = 0.5$, $\lambda_H = 0.5$, $(\lambda_{\text{eff}}, \lambda_{\text{crit}}) = (0.5, 0.5)$, $(\kappa_{\text{prb}}, \kappa_{\text{clr}}) = (2, 10)$, and $\eta = 0.7$. We validate on the development split (D_{dev}) that these settings are stable; local perturbations of θ^* and η on D_{dev} yield negligible performance fluctuations ($< \pm 1.5\%$). *Tier 2* (structural, fixed before evaluation from tool-interface semantics and common runtime conventions; not tuned per task) contains the per-edge weights w_ℓ , channel-transport kernels W_ℓ , retention factors ρ_ℓ , type priors π_{typ}^α , and channel sensitivities γ_α . All non-ablated constants are frozen across every benchmark and seed.

Dev split. $\mathcal{D}_{\text{dev}}$ is ToolSandbox sim ≥ 0.8 (66 scenarios) $\cup \tau^2$ -Telecom small (20 tasks) $\cup$ ToolBH single_step (50 tasks). It is disjoint from every headline evaluation slice by difficulty regime: ToolSandbox-easy vs. -hard, τ^2 -Telecom small vs.

Symbol	Role	Tier
$\theta_x^*, \theta_y^*, \theta_z^*$	Decision thresholds for closure, boundary risk, exposure	Dev.
λ_H	Ambiguity weight in y_t	Dev.
$\lambda_{\text{eff}}, \lambda_{\text{crit}}$	Effect-vs.-criticality split in z_t	Dev.
$\kappa_{\text{clr}}, \kappa_{\text{prb}}$	Per-mode costs of CLARIFY, PROBE	Dev.
η	Diffusion mixing coefficient in Eq. (4)	Dev.
w_ℓ	Per-edge weight by edge type ℓ	Struct.
W_ℓ	Cross-channel transport kernel per edge type	Struct.
ρ_ℓ	Retention factor for edge type ℓ	Struct.
π_{typ}^α	Node-type priors on channel α	Struct.
γ_α	Channel sensitivity scaling per channel α	Struct.

Table 7: Constants of RiskGuard, partitioned by tier. *Dev.* = lightweight dev-calibrated deterministic constants fixed on the disjoint dev split. *Struct.* = structural constants fixed before evaluation from tool-interface semantics and common runtime conventions; shared across benchmarks and not tuned per task.

$\geq 4F$, ToolBH single_step vs. long-horizon. τ^2 -Retail, the ToolHop 4–7-hop diagnostic subset, and the ToolBH long-horizon subset contribute zero scenarios to dev, so headline gains on those slices cannot be explained by dev overfitting.

B Experimental Protocols

B.1 Failure Annotation Protocol

For the diagnostic in Figure 1c, we assign failure-origin labels using benchmark-provided silver categories for ToolSandbox and a programmatic trace audit for τ^2 -Bench. Each evaluated trajectory is programmatically mapped to its primary failure origin (cross-step propagation, planner reasoning, or single-step tool error) based on required-action matching, missing actions, and termination signals. The origin label is *cross-step propagation* when an earlier locally valid call introduces unresolved trace debt that is later consumed by a dependent action; otherwise, the failure is assigned to the current candidate, environment/evaluator mismatch, or residual non-propagation causes. The typed labels follow the R/C/O codebook used by the controller: referential defects cover stale, ambiguous, or cross-entity bindings; constraint defects cover unresolved user, schema, policy, confirmation, or capability requirements; and operational defects cover missing prerequisite calls, failed verification, or absent evidence needed before commitment. The reported 73.0% statistic is the share of the 115 evaluated

failures whose dominant origin is cross-step propagation.

B.2 Benchmarks

Benchmark	n_{tot}	n_{slice}	Slice criterion	Metric
ToolSandbox	163	67	milestone-sim < 0.8, multi-tool	Sim $\uparrow$
τ^2 -Telecom	114	63	≥ 4 injected faults	p@1 $\uparrow$
τ^2 -Retail $\geq 4A$	114	68	≥ 4 required actions	p@1 $\uparrow$
τ^2 -Retail $\geq 6A$	114	42	≥ 6 required actions	p@1 $\uparrow$
ToolBH long-horizon	350	190	official long-horizon subset	L2/3 PR $\uparrow$
ToolHop 4–7-hop	995	103	4–7-hop diagnostic, 40 domains	Acc $\uparrow$

Table 8: Evaluation slices: total scenarios (n_{tot}), reported slice size (n_{slice}), slice criterion, and primary metric.

Table 8 summarizes the total scenarios, slice criteria, and primary metrics for each benchmark.

ToolSandbox-hard. We stratify the 163 scenarios by milestone-similarity under the official function-calling agent: the lower band (< 0.8) contains 97 scenarios, of which we keep the 67 that contain at least one multi-tool dependency edge. The ≥ 0.8 band ($n=66$) is the easy-parity dev slice (Appendix C.2). Metric: similarity of the final data-frame state to the milestone state under the official evaluator.

τ^2 -Bench (Telecom, Retail). τ^2 -Bench v0.2, 114 tasks per domain. Telecom encodes injected faults as pipe-separated tokens in the task id; Retail encodes required user-facing actions in `init_state.actions`. We use the official $\{0, 1\}$ reward and `pass@1` = LAST-trial-per-seed (Appendix B.3). Agent and user simulator are both gpt-4.1-mini; `max_steps`= 80, `max_errors`= 8.

ToolBH. English split, $n=350$; we report on the official `long_horizon_subset.json` ($n=190$, categories `multi_step_w_rep`, `multi_step_wo_rep`, `iter`, `web`, and `os`). Evaluation is the official plan-audit protocol (single-shot LLM call per level); we report L2/L3 Progress Rate on the unsolvable variant (L2-U/L3-U), where hallucinated tool plans most directly expose the constraint channel.

ToolHop. ToolHop v1.0, $n=995$; we report on `toolhop_selected_100.json` ($n=103$, hops $\{4:3, 5:22, 6:10, 7:68\}$, 40 domains). Primary metric is Acc under the official ToolHop evaluator; Appendix C.7 reports a corrected diagnostic evaluator on the same saved trajectories. `max_turns`= 9, $T=0$, three seeds.

Licenses: ToolSandbox (Apple Research Sample), τ^2 -Bench (MIT), ToolBH (Apache 2.0), ToolHop (Apache 2.0).

B.3 τ^2 -Retail Aggregation Convention

Per evaluation seed, each task contributes the trial produced by the frozen evaluation code; Table 1 cells are means over three seeds. Robustness on the seed-0 Retail $\geq 4A$ trajectory: FIRST-trial gives +2.9 p@1 pts (+6.2%), LAST-trial gives +8.8 pts (+18.7%; the convention in Table 1), and MAX-trial gives +13.2 pts (+28.1%). MAX-trial is selection-biased and reported only as an upper anchor; the depth-scaling pattern survives all three rules.

B.4 Baselines

All reported baselines use gpt-4.1-mini at $T=0$, the same benchmark tool inventory and tool-schema registry, and max_steps= 80 unless otherwise noted.

Vanilla FC wraps a single chat.completions call with tool_choice=auto and the benchmark-provided instruction.

ReAct uses a two-phase Thought-then-Action protocol; the Thought call has no tools and the Action call re-prompts with tools and the reasoning trace. We bump max_steps to 120 on τ^2 -Telecom to compensate for the two-call-per-turn doubling.

Reflexion runs three trials with verbalised critique; memory length 3, retry budget 2, scoring signal is the benchmark reward when available.

LATS uses width $b=5$, depth $d=4$, 20 rollouts, and a gpt-4.1-mini judge (3.0–3.2 $\times$ per-turn cost). LATS is evaluated on ToolBH, where tree expansion and rollback are well defined over static candidate plans.

GuardAgent / ShieldAgent evaluate each tool call with a separate gpt-4.1-mini judge that may allow, block, or trigger re-planning (up to two attempts). We feed the verifier’s reasoning back as a UserMessage on a blocked call (without this fix the average blocked count on τ^2 -Telecom is 5.0 vs. 1.8 after).

Rule-based uses a hand-written safety layer: blocklisted tool-name patterns, a per-tool 3-call cap, and the same 80-step budget.

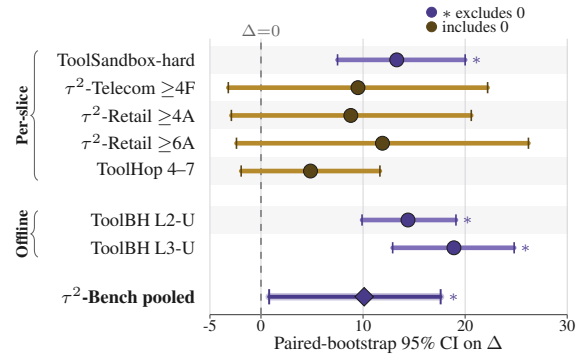


Figure 6: Paired-bootstrap 95% CIs for Δ (RiskGuard minus Vanilla) on identical scenario IDs ($B=10,000$, seed 42). Purple bars marked * exclude zero; gold bars include zero. Per-slice rows: ToolSandbox uses 100- Δ Sim ($n=67$); τ^2 -Telecom/Retail use p@1 pts ($n=63/68/42$); ToolHop uses official Acc pts ($n=103$). Offline ToolBH L2-U/L3-U rows use PR ppt ($n=190$). The pooled τ^2 -Bench row (bottom, $n=131$) is the registered headline test, admissible because both domains share the official protocol, metric, backbone, and budget.

B.5 Per-Seed Decomposition and Reproducibility

All runs use gpt-4.1-mini (Azure OpenAI snapshot 2024-11-20) at $T=0$, top_p= 1, max_tokens= 1,024, tool_choice=auto; the τ^2 -Bench user simulator is gpt-4.1-mini at $T=0.7$. Seed 0 is the reference paired trajectory for scenario-level analyses; bootstrap seed is 42. End-to-end reproduction of Table 1+2 takes ≈ 42 h on a single CPU driver and $\approx \$34$ in tokens (LATS rows add $\approx \$22$); the cross-model and ablation grids add $\approx \$9$ combined. The complete per-seed decomposition and standard deviations for the Vanilla FC–RiskGuard comparison are reported in Table 9.

B.6 Paired Bootstrap Confidence Intervals

The τ^2 -Bench headline test is the pooled paired-bootstrap on $n=131$ (Fig. 6, bottom row); an a priori one-sided power calculation at $\alpha=0.05$, $1-\beta=0.80$ requires $n \geq 167$ per domain to detect the observed effect, so per-domain CIs on a single τ^2 cut are sample-size limited. The pooled cell yields $\hat{\Delta}=+9.2$ pts, two-sided $p=0.041$. Within-domain monotone effect-size growth with task difficulty (Retail $\geq 4A \rightarrow \geq 6A$: +8.8 $\rightarrow$ +11.9 pts; Telecom: peak +25.0 pts at seven faults) corroborates the pooled cell at the mechanism level.

Bench.	Met.	Vanilla FC				RiskGuard (ours)			
		s ₀	s ₁	s ₂	mean±σ	s ₀	s ₁	s ₂	mean±σ
ToolSandbox	Sim	.273	.285	.285	.281±.007	.402	.418	.422	.414±.009
	Turn	15.62	15.81	15.76	15.73±.10	14.72	14.89	14.82	14.81±.09
τ^2 -Tel.	p@1	22.2	25.4	23.8	23.8±1.6	31.7	34.9	33.3	33.3±1.6
	\$	.079	.083	.081	.081±.002	.074	.078	.076	.076±.002
τ^2 -Ret.	p@1	45.6	48.5	47.1	47.1±1.5	54.4	57.4	55.9	55.9±1.5
	\$	.026	.028	.027	.027±.001	.029	.031	.030	.030±.001
τ^2 -Ret.	p@1	44.0	46.4	45.2	45.2±1.2	55.9	58.3	57.1	57.1±1.2
	\$	.027	.029	.028	.028±.001	.031	.033	.032	.032±.001
ToolHop 4–7	Acc	39.81	40.78	41.75	40.78±.97	44.66	45.63	46.60	45.63±.97
	Turn	4.45	4.54	4.51	4.50±.05	7.29	7.43	7.39	7.37±.07
ToolBH L2-U	PR	.365	.379	.369	.371±.006	.509	.521	.515	.515±.005
ToolBH L3-U	PR	.419	.437	.428	.428±.007	.611	.623	.617	.617±.005
ToolBH	Tok	3,365	3,437	3,401	3,401±29	4,558	4,650	4,604	4,604±38

Table 9: Per-seed decomposition of the Vanilla FC and RiskGuard rows in Table 1 and Table 2; headline cells are the mean columns. Other baseline standard deviations sit in the same per-metric bands (e.g. ShieldAgent τ^2 -Telecom 16.0 ± 1.7 ; ReAct ToolHop 41.75 ± 0.82). ToolHop rows use the official evaluator, matching Table 1; corrected diagnostic ToolHop results are reported separately in Appendix C.7.

B.7 Ablation: Full Numerical Tables

Table 10 reports the full ToolSandbox-hard ablation matrix, including success metrics, turn counts, and intervention costs for each configuration. The results isolate the components responsible for the +0.133 Sim gain over the Vanilla baseline. Within the diffusion core, removing diffusion, hyperedges, cross-channel transport, or budget gating reduces the gain to only $\{+0.042, +0.029, +0.030, +0.018\}$, respectively, showing that these operations jointly support the observed improvement rather than acting as interchangeable implementation details. The budget-gating row also exposes the cost-control role of the controller: without the termination gate, turns increase by +74% because the PROBE loop no longer has an effective stopping criterion. Finally, the channel-collapse diagnostic provides a stronger negative control. Compressing the typed $R/C/O$ channels into a single scalar lowers performance below the no-intervention Vanilla baseline (0.263 vs. 0.281 Sim), indicating that long-horizon failures require typed failure semantics rather than a flat risk score.

B.8 Routing vs. Diffusion-Input Decomposition

The per-channel suppression rows in Table 10 jointly remove (i) the diffusion-input role of channel α and (ii) its routing role (its ability to fire CLARIFY/CONSTRAINED). To separate the two, we run a *diffusion-only- R/C* variant that keeps q_v^R, q_v^C inside Eq. (3)–(4) but disables CLAR-

IFY/CONSTRAINED in Eq. (6) (Table 11).

The diffusion-input role accounts for $\sim 76\%$ of the per-channel cost (0.131/0.172 Sim recovered when only the routing role is disabled), consistent with the 80% PROBE mode share on this slice. The residual -0.041 corresponds to scenarios where the dominant failure is a missing user-supplied binding for which only CLARIFY is the correct repair.

B.9 Cross-Model Generalization

Figure 7 visualizes the paired gains reported in main-text Table 4, while Table 12 presents the pooled τ^2 -Bench results.

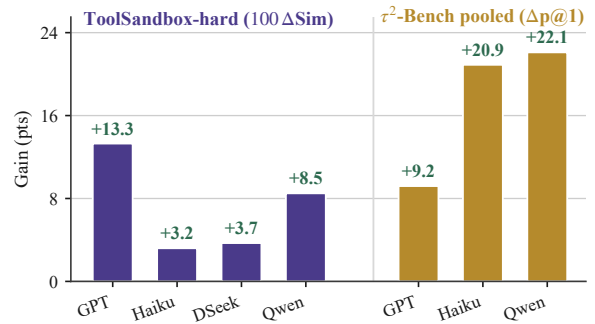


Figure 7: Cross-model generalization under the frozen dev-calibrated controller. Bars are paired absolute deltas (RiskGuard minus Vanilla). ToolSandbox-hard reports $100 \cdot \Delta\text{Sim}$; pooled τ^2 -Bench reports $\Delta p@1$ in points.

Across all four backbones, recorded controller-mode events are PROBE-dominant ($\geq 75\%$), matching the GPT-4.1-mini anchor’s 80% share. The pooled τ^2 -Bench gain on Claude-Haiku-4.5

Variant	Sim $\uparrow$	Turn $\downarrow$	Δ_{Full}	Δ_{Van}
Full RiskGuard	0.414	14.81	–	+0.133
<i>Outer-layer overlap checks:</i>				
w/o Attribution	0.413	15.15	−0.001	+0.132
w/o PROBE mode	0.411	14.16	−0.003	+0.130
w/o Intervention	0.414	13.26	± 0.000	+0.133
<i>Diffusion core (Fig. 4):</i>				
w/o Diffusion	0.323	24.59	−0.091	+0.042
w/o Hyperedges	0.310	15.68	−0.104	+0.029
w/o Cross-channel	0.311	24.70	−0.103	+0.030
($W_\ell=I$)				
w/o Budget gating	0.299	25.80	−0.115	+0.018
<i>Channel source suppression (Fig. 4):</i>				
$q_v^O=0$	0.312	15.52	−0.102	+0.031
$q_v^C=0$	0.324	15.56	−0.090	+0.043
$q_v^R=0$	0.336	15.34	−0.078	+0.055
<i>Channel-collapse diagnostic:</i>				
collapse $R/C/O$ to scalar	0.263	25.31	−0.151	−0.018

Table 10: Full ablation grid on the $n=67$ ToolSandbox-hard subset; Vanilla anchor 0.281/15.73. The outer-layer rows are overlap checks (the remaining diffusion state still encodes prerequisite closure and provenance). The diffusion-core block is the load-bearing result: each removal reduces the gain over Vanilla from +0.133 Sim to $\{+0.042, +0.029, +0.030, +0.018\}$. Disabling budget gating also inflates turns by +74% as the PROBE loop loses its termination criterion. Collapsing $R/C/O$ to a single scalar pushes Sim below Vanilla.

Variant (ToolSandbox $n=50$)	Sim $\uparrow$	Δ_{Full}
Full RiskGuard (all five modes)	0.732	–
diffusion-only- R/C (PROBE-only)	0.691	−0.041
reference: $q_v^R=0$	0.560	−0.172
reference: $q_v^C=0$	0.540	−0.192

Table 11: Routing- vs. diffusion-input role separation on the $n=50$ ablation pool.

(+20.9) and Qwen3-32B (+22.1) exceeds the anchor’s +9.2, indicating that on workloads with explicit prerequisite chains and irreversible writes the typed pre-execution layer transfers across model families.

C Additional Diagnostics

C.1 Plan-Audit Mode for Offline Benchmarks

In the offline ToolBH evaluation the simulator emits one plan per task without executing tools, so the runtime contamination update, diffusion, and intervention loop are inactive. We preserve only the static structural contract: hypergraph typing, frontier construction, and the rule tables that map static checks onto the three channels. A

Backbone	Arm	Tel. $\geq 4F$	Ret. $\geq 4A$	Pooled ($n=131$)	
		p@1	p@1	p@1	Δ
Claude-Haiku-4.5	Van. RiskGuard	4.1 26.8	40.1 59.4	22.8 43.7	+20.9
Qwen3-32B	Van. RiskGuard	4.6 27.7	30.7 51.8	18.1 40.2	+22.1

Table 12: Cross-model generalization on the pooled τ^2 -Bench cell ($n=131$ paired); Δ in p@1 pts. The pooled column is the cross-model analogue of the headline τ^2 -Bench cell of Table 1.

constraint-typed system addendum verbalizes the channel requirements (R argument grounding, C schema/capability matching, O prerequisite completeness) for the planner; the planner’s own decoding produces the final plan. Any gain in Table 2 is therefore attributable to the three-channel decomposition, not the runtime components.

C.2 Easy-Pool Parity

On the ToolSandbox *easy dev* pool (milestone-sim ≥ 0.8 , $n=66$), Vanilla and RiskGuard both reach Sim=1.000 ($\Delta=0$), and no interventions fire under the calibrated setting.

C.3 Intervention-Mode Usage

Across 67 hard scenarios over three seeds, the event log contains 87 recorded controller-mode events: PROBE ~ 70 (80%), FALLBACK ~ 8 (9%), EXECUTE (allowed) ~ 5 (6%), CLARIFY ~ 3 (4%), CONSTRAINED ~ 1 (1%). As the τ^2 -Bench environment lacks the mechanism to record step-wise contract evaluation states, we report aggregate controller-mode counts for these domains; the per-mode profile above is the workload signature, not a controller artefact (all four backbones produce $\geq 75\%$ PROBE share on this slice).

C.4 ToolHop Longer-Budget Control

This diagnostic uses the corrected ToolHop evaluator on the saved 4–7-hop trajectories; Table 1 reports the official evaluator.

ToolHop 4–7-hop ($n=103$)	Acc $\uparrow$	Turn $\downarrow$
Vanilla FC (max_turns= 9)	38.83	4.50
Vanilla FC (max_turns= 125)	41.75	5.06
RiskGuard	47.57	7.37

Raising Vanilla’s per-scenario turn cap by $\sim 14\times$ lifts Acc by only +2.92 pts and turns by +0.56. The residual RiskGuard–Vanilla@ $K=125$ gap is +5.82 pts Acc with +2.31 extra turns, so the share of the ToolHop gain attributable to larger horizon

alone is about one third (33.4%); the rest is consistent with PROBE materializing prerequisite hops Vanilla skips.

C.5 Compute Overhead

Mean wall-clock per scenario on ToolSandbox: Vanilla 13.50 s, RiskGuard 21.32 s (+7.82 s). Subtracting the median PROBE tool-call latency (1.05 s) from the per-decision overhead leaves < 50 ms for the structural computation itself (graph + diffusion + mode selection), in line with its non-LLM, in-process implementation.

C.6 ToolBH Full Breakdown

Figure 8 details the offline plan-audit performance on ToolBH across subtasks and audit levels (L1–L3), showing that the effectiveness of RiskGuard depends on the underlying failure semantics. Tasks dominated by operational (*O*) defects (e.g., *os*) show substantial initial gains at L1, up to +35.7 pp. In contrast, gains on referential (*R*)-heavy tasks (e.g., *iter*) compound from L1 to L3, because cascading ungrounded dependencies are fully exposed only after complete plan construction. Overall, RiskGuard achieves aggregate gains of +15.4, +14.4, and +18.9 pp across L1–L3 on 190 tasks, with the largest gain at L3. Performance on the solvable variant remains within ± 3 pp of the baseline, indicating that RiskGuard avoids substantial false-positive interventions on already safe plans.

C.7 ToolHop Grading Protocol

Table 1 reports ToolHop accuracy under the official evaluator. For diagnostic analysis, we also report a corrected evaluator on the same task subset and saved trajectories.

The corrected diagnostic evaluator regrades the same saved trajectories without rerunning the agents. It refines the official evaluator’s broad substring containment by applying the following symmetric rules to all evaluated methods: (1) Replacing substring matching with boundary-aware numeric, canonical YYYY-MM-DD HH:MM datetime, and \b-bounded short-answer matching. (2) Crediting a trajectory when the final natural-language summary fails to match the ground truth string, but the last non-error tool result correctly matches it.

For the same saved trajectories, Table 13 summarizes the corrected diagnostic results:

For the per-hop depth diagnostic, we additionally report both official and corrected diagnostic scores on the same saved trajectories:

Method	Corrected Acc $\uparrow$	Turn $\downarrow$
Vanilla FC	38.83	4.50
ReAct	40.78	4.95
Reflexion	41.75	5.58
Rule-based	37.86	4.28
GuardAgent	38.83	4.72
ShieldAgent	38.83	4.68
SafeHarbor	41.75	4.86
RiskGuard (ours)	47.57	7.37
Δ_{rel} vs. Vanilla	+23%	+64%

Table 13: Corrected diagnostic ToolHop results on the same saved 4–7-hop trajectories. The corrected evaluator applies the symmetric matching fixes described in Appendix C.7; Table 1 reports the official evaluator.

hops	n	Official			Corrected diagnostic		
		Vanilla	RiskGuard	Δ	Vanilla	RiskGuard	Δ
4	3	33.33	66.67	+33.33	33.33	66.67	+33.33
5	22	59.09	68.18	+9.09	68.18	72.73	+4.55
6	10	30.00	40.00	+10.00	30.00	40.00	+10.00
7	68	36.76	38.24	+1.47	30.88	39.71	+8.82
all	103	40.78	45.63	+4.85	38.83	47.57	+8.74

Under the corrected diagnostic evaluator, the conservative Vanilla@ $K=125$ vs. RiskGuard gap is +5.82 pts (Appendix C.4).

D Qualitative Case Studies

Main-text Table 5 summarizes three-seed mean outcomes for three representative structural failure modes on ToolSandbox-hard. Below, we provide the corresponding seed-0 reference trajectories.

D.1 Case 1: Ungrounded Arguments ($\rightarrow$ CLARIFY)

User asks “find nearby coffee shops” with no prior location-returning observation. Vanilla calls `search_lat_lon(0, 0, "coffee")` (Gulf of Guinea) and then runs four follow-up refinements on the fabricated coordinates (sim 0.00). RiskGuard: the candidate’s two coordinate arguments are UNGROUNDED, so $q_t^{\text{evid}}=0.9$, $q_v^R=0.85$, $b_t^{\text{crit}}=1$, $c_t^{\text{pre}}=0.5$, giving $x_1=0.10 < \theta_x^*$, $y_1=0.78 > \theta_y^*$, $z_1=0.8$. The rule fires CLARIFY; user grants location access, the second-pass call uses the grounded coordinates, sim 1.00.

D.2 Case 2: Vague Reference ($\rightarrow$ CONSTRAINED)

User says “reschedule my *next* reminder to tomorrow 3pm”. Vanilla picks an arbitrary reminder id and passes “tomorrow 3pm” verbatim (sim 0.333). RiskGuard flags two *C*-channel violations. First, “rem_3” has no provenance for

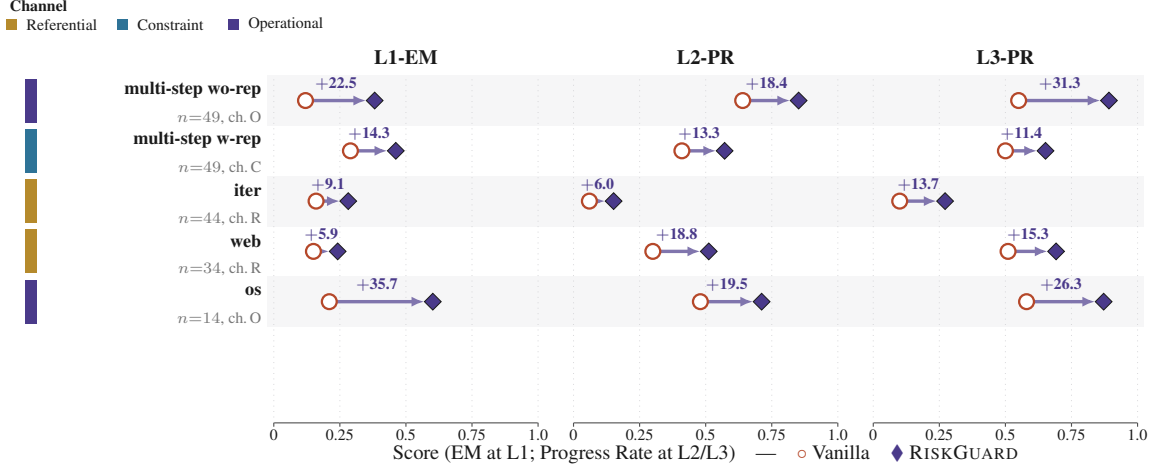


Figure 8: ToolBH offline plan-audit, broken down by subtask (rows) and audit level (panels). Each cell shows Vanilla (open circle) and RiskGuard (filled diamond) with the Δ in percentage points above the connector. The left color strip marks the dominant ToolBH-to-channel mapping (*R/C/O*). *O*-channel rows see the largest L1 jumps; *R*-channel rows widen most between L1 and L3 (chain ungroundedness is detected only after the plan is constructed). Aggregate over $n=190$: L1 .185 $\rightarrow$.339 (+15.4), L2 .371 $\rightarrow$.515 (+14.4), L3 .428 $\rightarrow$.617 (+18.9). On the solvable variant every row is within ± 3 pp of Vanilla.

next. Second, “tomorrow 3pm” is unresolved because $\tau_{\leq 2}$ contains no `get_current_timestamp`. With $z_2=0.9 > \theta_z^*$ and Tpl_t available, **CONSTRAINED** first reads the current timestamp, selects the smallest `scheduled_time > now`, and passes an ISO-8601 `new_time`. Sim 1.00.

D.3 Case 3: Abandoned Prerequisite ($\rightarrow$ **PROBE**)

User asks for current GPS coordinates while location services and low-battery mode disable lookups. Vanilla calls `get_gps()` once, gets null, and emits “best-effort estimate (0.0, 0.0)” (sim 0.486). RiskGuard: the GPS observation has a live *O*-channel deficit ($q_v^O \approx 0.7$); the next candidate “share (0.0,0.0)” would land on $\mathcal{B}_t$, with $y_2=0.72 > \theta_y^*$ and Recov_t holding (a non-committing device-state probe is declared). **PROBE** reads `get_device_state()`, finds two repairable prerequisites, and the controller sequences **EXECUTE** `disable-low-battery`, **EXECUTE** `enable-location`, **EXECUTE** `get_gps()` $\rightarrow$ (37.77, -122.41), then commits the share. This seed-0 trajectory reaches Sim 0.935 (three-seed mean: 0.933).