HARNESS FOR INFERENCE-TIME ALIGNMENT OVER EXECUTION TRAJECTORIES

Anonymous authors
Paper under double-blind review

ABSTRACT

Harness engineering has emerged as an important inference-time technique for large language model (LLM) agents, aiming to improve long-term performance through task decomposition and guided execution. However, **more elaborate harnesses are not uniformly better**: increasing decomposition or guidance can sometimes improve execution, but can also reduce final task success. We study harness design through the lens of inference-time trajectory alignment. This perspective separates harness into two mechanisms: task decomposition, which structures a task into sub-goals, and guided execution, which reshapes local action distributions during execution. This decomposition allows us to quantify how workflow granularity, retry budgets, and guidance-induced action reweighting shape the performance limits of harness design. It further reveals concrete failure modes, including over-decomposition, over-pruning, and hallucinated execution. We validate these predictions through controlled synthetic experiments and real terminal agent benchmarks. Inspired by the theory, we further show that **effective harnesses can be partial**: specifying only the initial steps and leaving the remaining execution to agent can achieve higher pass rate than fully structured workflows.

1 INTRODUCTION

Large language model agents have demonstrated impressive capabilities in solving long-horizon interactive tasks involving complex software engineering (Anthropic, 2026; OpenAI, 2026a), scientific discovery (Qu et al., 2026; Jin et al., 2025), and autonomous tool use (Zeng et al., 2026; Team et al., 2025). A key technique behind is *harnessing*: a scaffolding strategy that injects human priors into the agent's execution process (OpenAI, 2026b; Trivedy, 2026; Anthropic, 2025). By decomposing long-horizon tasks into structured sub-goals and providing guidance for intermediate decisions, a harness enables agents to complete complex tasks more reliably in an autonomous manner (Karpathy, 2026; Foundation, 2026; Harbor Framework Team, 2026; Browser Use, 2026).

This success suggests a tempting intuition: more elaborate harnesses should lead to better agents. From this view, harness design becomes a problem of adding structure: finer-grained decomposition, more detailed instructions, and tighter execution constraints (Erdogan et al., 2025; Wang et al., 2025; Dang et al., 2025). Yet this intuition conflicts with a central lesson from the history of AI (Sutton, 2019): human-designed structure often helps in the short run, but can limit a system's ability to search, adapt, and scale (Yan et al., 2025; Guo et al., 2025; Silver et al., 2017). This raises a basic question: **What should a harness specify, and what should leave for agent to resolve on its own?**

To answer this question, we theoretically model how harnesses shape the inference-time trajectories of LLM agents. We separate a harness into two components: *workflow*, which specifies what the agent should achieve at each stage, and *guidance*, which biases how the agent acts within that stage. This separation yields two findings through stage-level gaps. First, **finer-grained decomposition is not always better**. The optimal granularity must align the required sub-goal scale with the agent's controllable progress scale under the given tolerance and retry budget. Second, **guidance helps only when aligned**. It improves performance by shifting probability mass toward recoverable actions. Moreover, when guidance favors actions that follow the instruction rather than the task evidence, it can instead lead to hallucinated responses. These findings recast harness design as an alignment problem, where imposed structure must match the agent's capability and the available task evidence.

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

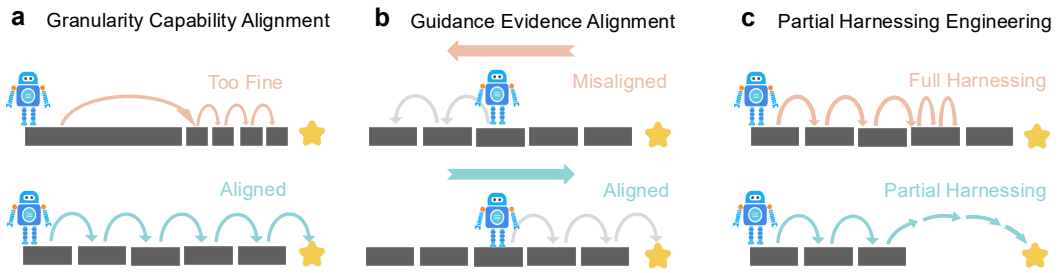


Figure 1: Alignment Principles for Harnessed Agent Execution

Beyond these observations, we find that although a harness is usually designed for the whole task, keeping only its early stages can be more effective. This points to a counterintuitive principle for harness design: effective harnesses need not specify the full execution path. We formalize this idea as **Partial Harnessing**, a new design strategy that specifies only the initial stages and leaves the remaining execution to the agent. Our experiments show that partial harnessing can outperform fully specified workflows, suggesting that harness design should decide not only what structure to add, but also when to stop adding it.

We summarize our contributions as follows:

- We formulate harness design as an inference-time alignment problem by decomposing a harness into workflow and guidance components, yielding two stage-level principles: sub-goal scale should match agent capability, and guidance should match task evidence.
- We introduce partial harnessing as a design strategy that stops scaffolding once its reliability cost outweighs tail-risk reduction, formalized through a marginal stopping rule.
- We empirically validate these predictions on synthetic cumulative-progress tasks and Terminal-Bench v2, where alignment patterns appear and partial harnesses outperform full workflows.

2 Related Work

Long-horizon agent execution. LLM agents have grown from single-step responders into systems capable of executing long-horizon tasks in interactive environments. The starting point is the reasoning-action loop of ReAct (Yao et al., 2022), which later work extends with feedback and memory (Shinn et al., 2023; Packer et al., 2023) and with explicit planning and search (Yao et al., 2023; Zhou et al., 2023a). As these capabilities matured, evaluation moved into realistic long-horizon environments spanning web interaction (Zhou et al., 2023b), software engineering (Jimenez et al., 2023), and computer use (Xie et al., 2024), prompting recent systems to treat the surrounding scaffold itself as a design object and optimize the workflow that orchestrates execution (Hong et al., 2023; Zhang et al., 2024). A common thread across this progression is that every step adds more human-designed structure around the model. Our work asks what such structure should actually specify, and what should be left to the agent.

Harness optimization. A related line of work treats the scaffolds around LLM execution as optimization targets. Early methods optimize prompts directly, by generating, scoring, or refining natural-language instructions (Zhou et al., 2022; Pryzant et al., 2023; Yang et al., 2024; Fernando et al., 2023). This view is extended from single prompts to multi-stage LM programs, where systems such as DSPy, MIPRO, Self-Refine, and TextGrad optimize instructions, demonstrations, or intermediate artifacts across multiple model calls (Khattab et al., 2023; Opsahl-Ong et al., 2024; Madaan et al., 2023; Yuksekgonul et al., 2024). More recent work broadens the search space to agent modules, executable workflows, and full harness implementations (Shang et al., 2024; Hu et al., 2024; Agrawal et al., 2025; Novikov et al., 2025; Lee et al., 2026). These works primarily ask how to discover better scaffolds. In contrast, we ask when additional scaffold should help at all: which decompositions and guidance rules align with the agent's execution, and which ones over-specify the trajectory.

Human priors and agent autonomy. A broader line of work studies how control should be shared between humans and autonomous systems. Mixed-initiative interaction and adjustable autonomy show

that authority need not belong entirely to either side, but can be allocated according to uncertainty, context, and task demands (Horvitz, 1999; Scerri et al., 2002). Recent LLM agent systems revisit this issue as humans provide goals, constraints, feedback, or oversight while agents execute increasingly long-horizon tasks (Feng et al., 2024; Zou et al., 2025; Wang and Lu, 2025). Harness design is a concrete instance of this trade-off: it specifies part of the trajectory through decomposition and guidance, while leaving the rest for the agent to resolve (Pan et al., 2026; Bui, 2026) . Our work studies when such human priors improve execution and when they over-constrain the agent's execution.

3 PRELIMINARY

Intuition. We think of solving a task as proceeding on two interacting timescales, one driven by the harness and one driven by the agent. The *outer* timescale belongs to the harness, which lays out a sequence of sub-goals $g_1 \to g_2 \to \cdots \to g_T$ toward the final answer and advances along this plan one sub-goal at a time. The *inner* timescale belongs to the agent: given the current sub-goal g_t, it repeatedly takes an action and observes the resulting state until g_t is met, at which point control returns to the harness and g_{t+1} is revealed. Crucially, the harness is not silent during this inner loop: even as the agent chooses each step, the harness simultaneously *shapes* the loop, nudging the trajectory toward behaviors it considers promising for g_t. The two timescales thus carry a clean division of labor: the harness decides *what to work on next* and *which trajectories to favor while working on it*, while the agent decides *how to take each step*.

How a Harness Shapes Execution. We now make two timescales precise. Consider tasks $x \sim \mathcal{D}$, each with a unique correct final answer $y^\star(x)$, and define a *harness* as an inference-time scaffold parameterized by $h = (\kappa, \lambda, \psi)$, where κ controls the decomposition granularity, λ controls the guidance strength, and ψ specifies the local guidance rule. These three parameters split cleanly along the two timescales: κ governs the outer one, and λ, ψ govern the inner one.

For the outer timescale, κ applied to a task x induces the ordered sub-goal sequence $\Delta_h(x) = (g_1, \ldots, g_{T_h(x)})$, the coarse plan from the intuition above. This sequence fixes the stage-level structure but leaves each inner trajectory to the agent, so a complete harness-conditioned execution takes the form

$$\tau_h(x) = (g_1, \tau_1, \ldots, g_T, \tau_T), \qquad T = T_h(x),$$

where each $\tau_t = (s_{t,0}, a_{t,0}, \ldots, a_{t,n_t-1}, s_{t,n_t})$ is the inner trajectory generated while pursuing g_t, with states $s_{t,j} \in \mathcal{S}$ and actions $a_{t,j} \in \mathcal{A}$. How each τ_t is actually produced is the job of the inner timescale.

For the inner timescale, let K_{t-1} collect everything observed before stage t, i.e. the task together with all preceding sub-goals and trajectories. Without guidance, the agent unrolls τ_t from K_{t-1} and g_t under its *base trajectory distribution* $\mathbb{Q}_{t,0}(\tau_t \mid K_{t-1}, g_t)$, defined autoregressively by $a_{t,j} \sim q_h(\cdot \mid H_{t,j})$ with $H_{t,j} = (K_{t-1}, g_t, s_{t,0}, a_{t,0}, \ldots, a_{t,j-1}, s_{t,j})$. The base distribution captures the agent acting on its own; the harness's role is to reshape it.

This reshaping is realized through ψ and λ_t, which together determine a non-negative weight $W_{t,\lambda_t}(K_{t-1}, g_t, \tau_t)$ measuring how well a candidate trajectory aligns with the behavior ψ prescribes for g_t. Reweighting the base distribution by this score yields the *guided trajectory distribution*,

$$\mathbb{Q}_{t,\lambda_t}(\tau_t \mid K_{t-1}, g_t) \;\propto\; \mathbb{Q}_{t,0}(\tau_t \mid K_{t-1}, g_t)\, W_{t,\lambda_t}(K_{t-1}, g_t, \tau_t),$$

which governs the agent's actual behavior at stage t. The strength λ_t controls the magnitude of this reweighting: at $\lambda_t = 0$ the weight is uniform and the guided distribution collapses to the base distribution, and as λ_t grows, $\mathbb{Q}_{t,\lambda_t}$ concentrates on trajectories preferred under ψ. Once τ_t terminates, control returns to the harness, g_{t+1} is revealed, and this two-level process repeats until stage T is complete.

From Final Success to Stagewise Recoverability. We now make this decomposition precise. Let $y(\tau_h)$ denote the final answer produced by execution τ_h, define the final success event as $\mathrm{Succ}_x(\tau_h) := \{y(\tau_h) = y^\star(x)\}$, and write the primitive harness-design objective as

$$\max_h \; \mathbb{E}_{x\sim\mathcal{D}} \left[\mathbb{P}_h(\mathrm{Succ}_x(\tau_h) \mid x)\right].$$

To connect this terminal objective with process-level behavior, we introduce the completed prefix after stage t, $K_t := (x, g_1, \tau_1, \ldots, g_t, \tau_t)$ with $K_0 := x$, and let B_t denote the event that K_t is

recoverable, i.e. that some continuation under the remaining plan $(g_{t+1}, \dots, g_T)$ still reaches $y^\star(x)$. The intuition above says that final success is equivalent to recoverability holding throughout, which under goal consistency we write as $\mathrm{Succ}_x(\tau_h) \equiv \bigcap_{t=1}^{T_h(x)} B_t$.

This equivalence turns the terminal objective into a stagewise product. By the chain rule,

$$\mathbb{P}_h(\mathrm{Succ}_x(\tau_h) \mid x) = \prod_{t=1}^{T_h(x)} \bar{p}_t(h; x), \qquad \bar{p}_t(h; x) := \mathbb{P}_h(B_t \mid B_{<t}, x),$$

where $B_{<t} := \bigcap_{s<t} B_s$ and $B_{<1} := \Omega$, so $\bar{p}_t(h; x)$ is the conditional probability of remaining recoverable at stage t given that the run was recoverable through every earlier stage. Taking negative logarithms turns the product into a sum, $-\log \mathbb{P}_h(\mathrm{Succ}_x(\tau_h) \mid x) = \sum_{t=1}^{T_h(x)} -\log \bar{p}_t(h; x)$, which exhibits the process loss as the stagewise decomposition of the primitive final-success objective rather than an auxiliary term added on top of it.

4 Alignment Principles for Harness Design

Building on the recoverability framework of Section 3, we now present three alignment principles, each addressing a distinct lever the harness has over execution: granularity–capability alignment in Section 4.1, guidance–evidence alignment in Section 4.2, and partial harnessing in Section 4.3.

4.1 Granularity - Capability Alignment

A harness's outer timescale specifies sub-goals, but it does not specify them in the abstract: each sub-goal asks the agent to make a definite amount of progress within a finite execution budget. Whether a workflow helps therefore depends on a single relationship—between the progress each stage requests and the progress the agent can reliably realize at that stage. We characterize this relationship precise and show that it controls the final success probability.

Consider stage t in isolation. The harness requests latent progress ℓ_t and grants the agent at most M_t low-level steps in which to deliver it. After $m \le M_t$ such steps, the cumulative progress the agent can reliably control lies in a window $I_{t,m} = [\mu^-_{t,m}, \mu^+_{t,m}]$ with stochastic variation $\sigma_{t,m}$, and a tolerance ϵ_t describes how far the realized progress may drift from ℓ_t before the resulting prefix ceases to be recoverable. The relevant quantity is the smallest standardized gap between the requested progress and any scale the agent can reach within budget,

$$\rho_t^{(M_t)} := \min_{1 \le m \le M_t} \frac{\big(d(\ell_t, I_{t,m}) - \epsilon_t\big)_+^2}{2\sigma_{t,m}^2},$$

which vanishes when some reachable cumulative scale lies within tolerance of ℓ_t and grows quadratically in the gap when no such scale exists. The mismatch $\rho_t^{(M_t)}$ is therefore a stage-local property: it depends only on what the harness asks of stage t and what the agent can do within M_t steps.

Aggregating these stage-local mismatches across the workflow yields the main result.

Theorem 1 (Granularity-capability mismatch bound, informal)**.** *Consider a harness that decomposes a task x into $T = T_h(x)$ sub-goals. At stage t, suppose the harness requires latent progress ℓ_t, and the agent may take at most M_t low-level execution steps. Under the recoverability-tube, concentration, and boundary-contraction conditions in Appendix D.2, the final success probability satisfies*

$$\mathbb{P}_h(\mathrm{Succ}_x(\tau_h) \mid x) \le \exp\left(- \sum_{t=1}^{T_h(x)} \Big[\eta_t + \big(\rho_t^{(M_t)} - \log M_t\big)_+ \Big] \right).$$

The bound exposes two distinct contributions to per-stage loss. The execution cost η_t reflects the intrinsic difficulty of any one stage and is paid regardless of how the workflow is structured. By contrast, the granularity penalty $\big(\rho_t^{(M_t)} - \log M_t\big)_+$ appears only when the requested scale exceeds what the retry budget can absorb, and it precisely captures the cost of misaligned decomposition. A larger M_t enlarges the reachable set and softens this penalty through the $-\log M_t$ term, but cannot eliminate a structural gap between ℓ_t and the agent's execution dynamics.

This decomposition explains why finer workflows are not uniformly better. Specialize to a uniform T-stage workflow on a task with total latent progress L_x, so each stage requests L_x/T. If a single low-level step reliably advances progress by an amount in $[\mu^-, \mu^+]$ and each stage allows at most M steps, the cumulative scales reachable within a stage form the union

$$\bigcup_{m=1}^{M} [m\mu^-, m\mu^+],$$

and useful decompositions place L_x/T near this set. Small T pushes L_x/T above the union and leaves stages unreachable within M steps; large T pushes L_x/T below it and imposes milestones the agent cannot stop at without coordination loss. The reliable regime lies between these extremes, where each stage requires nontrivial progress while remaining reachable under the agent's realizable progress scale. Workflow granularity is thus a design parameter to be chosen against the agent's execution dynamics, not against the task's logical structure alone.

4.2 Guidance - Evidence Alignment

The inner timescale of a harness is shaped by guidance: at stage t, the weight W_{t,λ_t} reweights the agent's base trajectory distribution toward behaviors ψ prescribes for g_t. This reweighting is helpful when it concentrates probability on trajectories that keep the prefix recoverable, and harmful when it concentrates probability on trajectories that look locally preferred but foreclose successful continuations. We make this dichotomy precise through a single stage-local quantity.

Fix a recoverable prefix K_{t-1}, and let $R_t^{\mathrm{stg}}(K_{t-1})$ denote the set of stage trajectories that keep K_t recoverable. Under the base distribution, the average retention weight is $\bar{W}^{\mathrm{rec}}_{t,\lambda_t}(K_{t-1})$ on R_t^{stg} and $\bar{W}^{\mathrm{bad}}_{t,\lambda_t}(K_{t-1})$ on its complement. The retention gap

$$\Gamma_{t,\lambda_t}(K_{t-1}) := \log \bar{W}^{\mathrm{rec}}_{t,\lambda_t}(K_{t-1}) - \log \bar{W}^{\mathrm{bad}}_{t,\lambda_t}(K_{t-1})$$

measures how much more weight guidance places on recoverable trajectories than on non-recoverable ones, in log space. A positive gap means guidance preferentially preserves trajectories that keep a successful continuation available; a negative gap means it preserves trajectories that may look locally acceptable but render the prefix non-recoverable.

This single sign-valued quantity determines whether guidance helps at the prefix.

Theorem 2 (Guidance alignment via retention gaps, informal)**.** *Fix a task x, a stage t, and a recoverable prefix K_{t-1}. Under the regularity conditions in Appendix D.3, the stage recoverability probability after guidance satisfies*

$$Q_{t,\lambda_t}\big(R_t^{\mathrm{stg}}(K_{t-1}) \mid K_{t-1}, g_t\big) = \sigma\big(\omega_t^0(K_{t-1}) + \Gamma_{t,\lambda_t}(K_{t-1})\big)\,,$$

where $\omega_t^0(K_{t-1})$ is the unguided recoverability log-odds and $\sigma(u) = 1/(1+e^{-u})$. Consequently, guidance improves stage recoverability relative to the unguided law at the same prefix if and only if $\Gamma_{t,\lambda_t}(K_{t-1}) > 0$, and harms it if and only if $\Gamma_{t,\lambda_t}(K_{t-1}) < 0$.

The theorem reduces a multi-dimensional design choice—which trajectories to favor, and how strongly—to a one-dimensional diagnostic at each prefix. The unguided log-odds $\omega_t^0(K_{t-1})$ is a property of the agent and the prefix, fixed once both are specified. Guidance enters the recoverability probability only through Γ_{t,λ_t}, additively in log-odds space. Stronger guidance, formalized as a larger λ_t, scales the magnitude of Γ_{t,λ_t} but not its sign, so increasing λ_t amplifies whichever effect ψ already produces at K_{t-1}: helpful guidance becomes more helpful, and misaligned guidance becomes more harmful.

This sign-amplification structure explains why the same guidance can either reduce or amplify hallucination. In evidence-limited settings, a recoverable trajectory is one that stays within what the current observations justify. A guidance rule ψ that rewards evidence checking or uncertainty awareness places its weight on grounded trajectories, producing a positive retention gap; raising λ_t then suppresses ungrounded continuations and reduces hallucination. A guidance rule ψ that rewards detail, confidence, or instruction compliance without conditioning on evidence places weight on trajectories that satisfy ψ regardless of whether they remain grounded, producing a negative retention gap; raising λ_t then drives the agent further from the supported set, amplifying hallucination. Guidance strength is therefore not a generic reliability lever: its effect at a prefix is determined by the alignment between ψ and the evidence available at that prefix.

4.3 Partial Harnessing as a Marginal Reliability Trade-off

A harness's length, the number of stages it specifies before releasing the agent, is itself a design lever. Each additional scaffolded stage shortens the residual task left to the agent, but imposes another recoverability constraint that execution must satisfy. Adding a stage is helpful when its tail-risk reduction exceeds its reliability cost, and harmful when the cost dominates. We make this dichotomy precise through a single marginal comparison, and use it to characterize when *partial harnessing*, which specifies only an initial prefix of the trajectory and leaves the rest to the agent, outperforms full coverage.

Fix a task with total latent progress demand L_x and a scaffold step size $s > 0$. For $m \in \mathcal{J} := \{0, 1, \ldots, \lfloor L_x/s \rfloor\}$, let h_m denote the partial harness that specifies the first m scaffolded stages and leaves the residual length $L_x - ms$ to the autonomous agent. Two scalar quantities determine the trade-off: the *scaffold cost*

$$c_s \;:=\; -\log q_{\mathrm{scaf}}(s; M),$$

the negative log-reliability of executing one scaffolded stage, and the *tail risk*

$$\kappa_{\mathrm{tail}}(d; M) \;:=\; -\log q_{\mathrm{tail}}(d; M),$$

the negative log-reliability of finishing a residual task of length d autonomously. Under the homogeneous slice factorization in Appendix D.4, the negative log-success of h_m separates additively as

$$F(m) \;:=\; -\log \Pr_{h_m}\big(\mathrm{Succ}_x(\tau_h) \mid x\big) \;=\; m\, c_s \;+\; \kappa_{\mathrm{tail}}(L_x - ms; M),$$

where the first term grows linearly with coverage while the second shrinks as the autonomous tail becomes shorter.

The trade-off between these two terms collapses into a one-dimensional diagnostic at the margin. Let

$$\Delta(m; M) \;:=\; \kappa_{\mathrm{tail}}(L_x - ms; M) \;-\; \kappa_{\mathrm{tail}}(L_x - (m{+}1)s; M)$$

denote the reduction in tail risk obtained by adding the $(m{+}1)$-st scaffolded stage. Adding that stage strictly improves reliability if and only if this reduction exceeds its own scaffold cost, $\Delta(m; M) > c_s$. Partial harnessing thus follows a simple marginal principle: keep extending the harness only while the next scaffolded stage saves more residual risk than it introduces.

Theorem 3 (Coverage–autonomy alignment, informal)**.** *Under the homogeneous slice factorization and the diminishing-returns condition on κ_{tail} stated in Appendix D.4, $F(m)$ is discrete-convex in m, and the success probability $\exp(-F(m))$ is log-concave and unimodal over $m \in \mathcal{J}$. The smallest reliability-maximizing coverage is*

$$m_{\mathrm{peak}} \;=\; \min\{m \in \mathcal{J} :\; m + 1 \in \mathcal{J},\; \Delta(m; M) \le c_s\},$$

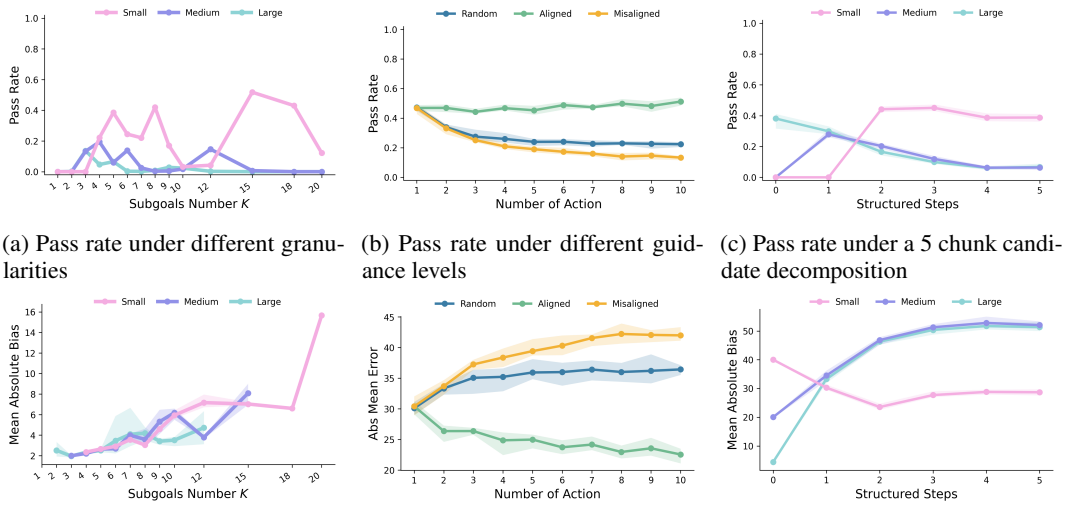
with $m_{\mathrm{peak}} = \max \mathcal{J}$ if the set is empty. For a target reliability $\alpha \in (0, 1)$, the minimum-structure α-reliable coverage is $m_\alpha = \min\{m \in \mathcal{J} : F(m) \le -\log \alpha\}$ whenever this set is nonempty.

The theorem connects harness length to the preceding alignment principles via a single marginal comparison. Far from a free parameter, the scaffold cost c_s is raised by granularity mismatch (Theorem 1) and misaligned guidance (Theorem 2), and lowered by well-aligned scaffolding. The tail risk κ_{tail}, by contrast, is a property of the agent. More capable agents attain smaller κ_{tail} on any residual, so $\Delta(m; M)$ crosses c_s at smaller m and the optimal coverage shifts earlier.

Harness length is therefore not a generic reliability lever, with its effect governed by the alignment between local scaffold quality and the agent's autonomous capability on the residual. The right harness is the smallest one that brings the residual within autonomous reach, and no smaller. The slice rule is a clean stopping principle rather than a universal optimization law. When added stages alter earlier behavior or tail success depends strongly on the realized prefix, the success curve may become multi-modal, and candidate coverages must be compared directly.

5 Experiments

To validate our theoretical predictions on harness design, we conduct **synthetic experiments** for controlled mechanism analysis in Section 5.1 and **real-data experiments** for realistic long-horizon validation in Section 5.2, examining three aspects: (i) *decomposition granularity*, (ii) *guidance alignment*, and (iii) *partial harness specification*.



(a) Pass rate under different granularities

(b) Pass rate under different guidance levels

(c) Pass rate under a 5 chunk candidate decomposition

(d) Final bias under different granularities

(e) Final error under different guidance levels

(f) Final bias under a 5 chunk candidate decomposition

Figure 2: **Three alignment principles for harness design.** (a, d) Granularity–capability: pass rate is non-monotonic in subgoal count K and final bias grows with finer decomposition, with each agent peaking at a different K. (b, e) Guidance–evidence: aligned guidance improves pass rate and lowers final error as the action pool grows, while misaligned guidance does the opposite. (c, f) Partial harnessing: pass rate is unimodal in scaffold count and final bias rises beyond the peak, with stronger agents reaching the peak earlier.

5.1 Synthetic Experiments

Setup. We study a synthetic addition task in which an agent must reach a target sum $G = 100$. A *harness* Δ_h decomposes G into an ordered sequence of stage goals $(g_1, \ldots, g_K)$ satisfying $\sum_{k=1}^{K} g_k = G$; for instance, $\Delta_h = (25, 25, 25, 25)$ when $K = 4$. Beyond decomposition, the harness also provides *guidance* that shapes how the agent acts within each stage. Together, these two roles let the harness serve as a high-level planner that both issues subgoals and constrains the low-level action model used to realize them.

Within stage k, let $s_t = \sum_{\tau=1}^{t} a_\tau$ denote the progress accumulated after t draws (reset to zero at the start of each stage). The agent maintains a pool of candidate truncated Gaussian action distributions

$$\mathcal{P} = \left\{ \mathcal{TN}_{[\ell_j, u_j]}(\mu_j, \sigma_j^2) \right\}_{j \in \mathcal{J}},$$

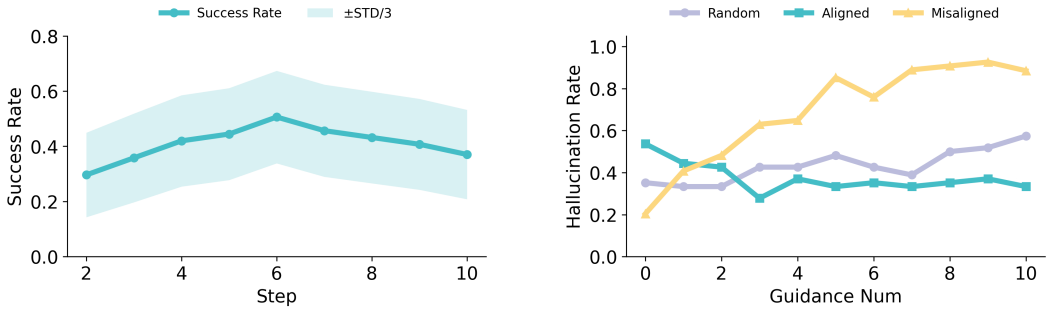
where $\mathcal{TN}_{[\ell,u]}(\mu, \sigma^2)$ denotes a Gaussian with mean μ and variance σ^2 truncated to $[\ell, u]$. The harness's guidance is operationalized in our simplified formulation as a *pruning* of this pool, leaving an admissible index set $\mathcal{J}_{\Delta_h} \subseteq \mathcal{J}$. At each draw, the agent *greedily* selects $j^\star \in \mathcal{J}_{\Delta_h}$ best matched to the residual gap $g_k - s_{t-1}$, samples

$$\tilde{a} \sim \mathcal{TN}_{[\ell_{j^\star}, u_{j^\star}]}(\mu_{j^\star}, \sigma_{j^\star}^2), \qquad a = \mathrm{round}(\tilde{a}) \in \mathbb{Z}.$$

and updates $s_t = s_{t-1} + a$. A stage is deemed successful once $s_t \in [g_k - \epsilon,\ g_k + \epsilon]$ with tolerance $\epsilon = 2$, after which the harness advances to stage $k + 1$.

Based on the synthetic task above, we conduct three controlled experiments on harness execution. First, we study decomposition granularity by varying the number of subgoals K across agents with different action scales. Second, we study guidance quality by varying the available action-pool size under different guidance strategies. In both experiments, we measure pass rate and final bias. We further include additional control experiments on retry budget, completion tolerance, and guidance pruning in Appendices A.2, A.3, and A.4..

I. Granularity–Capability Alignment. We vary the number of subgoals K across three agents with different action scales, sweeping G/K from values too coarse to be reached within the draw budget



(a) Workflow Granularity and Pass Rate on Terminal-Bench v2

(b) Guidance Quantity and Hallucination Rate under Different Alignment Conditions

Figure 3: Harness design trade-offs in real and controlled settings. (a) On Terminal-Bench v2, pass rate first improves and then declines as the workflow is split into more steps. (b) Hallucination rate under increasing amounts of guidance, comparing random, aligned, and misaligned guidance.

to values below the agent's natural action scale. Figures 2a and 2d show that pass rate peaks only when G/K matches the agent's controllable progress, and that final bias grows under overly fine decomposition as local completion errors accumulate. Both effects match the granularity penalty $(\rho^{(M)} - \log M)_+$ in Theorem 1.

II. Guidance–Evidence Alignment. We hold decomposition fixed and vary which action distributions remain available, comparing aligned guidance, misaligned guidance, and uniform random selection on the same pool. Figures 2b and 2e show that pass rate rises with pool size only under aligned guidance, while misaligned guidance produces lower pass rate and larger final error than uniform selection on the same pool. The benefit therefore comes from the sign of the retention gap rather than from additional constraints, matching Theorem 2.

III. Partial Harnessing. We fix a chunk size and sweep the number of scaffolded chunks r, leaving the remaining length to the autonomous agent. Figures 2c and 2f show that pass rate is unimodal in r for every agent, with the peak at smaller r for stronger agents, while final bias rises beyond the peak as additional structure accumulates terminal error rather than reducing it. The same shape persists across alternative chunk sizes (Figure 8). The unimodal shape and agent-dependent peak location match the marginal stopping condition $\Delta(r; M) \leq c_s$ in Theorem 3, with stronger agents reaching this condition at smaller r owing to their lower autonomous tail cost κ_{tail}.

5.2 Real Data

Setup. We conduct three experiments, each isolating one axis of harness design. (i) To probe *decomposition depth*, a granularity sweep on Terminal-Bench-2 (Merrill et al., 2026) varies sub-goal count $k \in \{1, \ldots, 10\}$ in the workflow given to a fixed agent. (ii) To probe *guidance quantity and quality*, a controlled Plotly chart-analysis task varies the number of Aligned, Misaligned, and Random guidance steps and measures the hallucination rate. (iii) To probe partial harnessing, we conduct a case study on a Terminal-Bench-2 task by decomposing the solution into 10 steps and progressively revealing a prefix of length ℓ. Full configurations are deferred to Appendix B.

(a) Granularity-capability alignment on Terminal-Bench v2. Figure 3a shows that pass rate first rises and then declines with sub-goal count, peaking at six steps. The drop on either side reflects the two failure regimes predicted by the theory: coarse workflows leave each step too large to complete reliably, and fine workflows fragment the task into milestones the model cannot meaningfully terminate at.

(b) Guidance-evidence alignment and hallucination. Figure 3b contrasts how hallucination scales with guidance count under three policies that differ only in whether they track task evidence. Under aligned guidance, hallucination stays near 0.35 regardless of count; under misaligned guidance it rises monotonically from 0.20 to nearly 0.90; random guidance, which mixes the two, interpolates in between. The same scaling axis therefore has opposite effects on reliability depending on the sign of

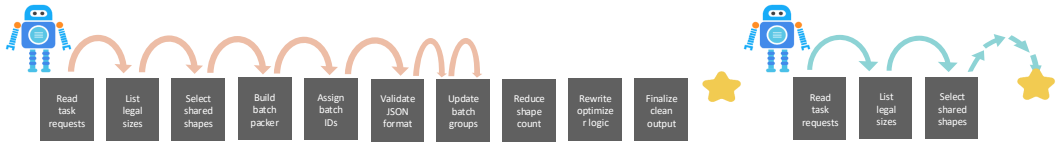


Figure 4: **Partial harnessing on llm-inference-batching-scheduler task.** Left: the fully specified workflow over-constrains execution and the agent gets lost in repeated intermediate revisions before reaching the final stages. Right: the partial workflow provides only an initial 3-step harness, after which the agent completes the remaining task through its own planning.

the retention gap, with stronger guidance amplifying whichever direction this sign points, matching the sign-amplification structure predicted by Theorem 2.

5.3 A Case Study on Partial Harnessing

Beyond the empirical results above, Figure 4 compares a partial harness and a full harness on the `llm-inference-batching-scheduler` task Merrill et al. (2026). The partial harness contains only three initial steps: read the inputs, identify feasible sizes, and select shared shapes. After this initial scaffold, the agent takes over the remaining execution, enters its own iterative refinement loop, and completes the task. In contrast, the full harness specifies the downstream procedure in much greater detail, but the additional structure does not translate into better control. Instead, the agent becomes stuck inside the prescribed scaffold, repeatedly revising intermediate decisions without reaching the final solution. This case illustrates why partial harnessing can outperform full specification: a short scaffold can guide the agent into the right search space while leaving enough autonomy for it to plan, adapt, and finish the task. We further conduct a broader multi-task evaluation of partial harnessing, with results summarized in Table 7.

6 Conclusion

We studied harness engineering as an inference-time alignment problem over agent execution trajectories. Separating a harness into decomposition—which sets the progress scale imposed on the agent—and guidance—which reshapes the local trajectory distribution—reveals that stronger harnesses are not necessarily better: decomposition helps only when its scale matches the agent's controllable progress, and guidance helps only when it preserves trajectories supported by the current evidence. Misalignment in either role turns scaffolding into a reliability bottleneck rather than an aid, with hallucination as one concrete instance of guidance retaining the wrong part of the local trajectory space.

This view recasts harness design from the problem of adding more structure to the problem of choosing what to specify, how strongly to specify it, and when to stop. In particular, partial harnessing shows that a harness need not cover the full execution path: once the remaining task falls within the agent's autonomous capability, continued control adds reliability cost without repaying it. The right harness is the smallest one that keeps the agent on a recoverable trajectory, and no smaller. For clarity, we adopt simplified settings and assumptions to isolate the core mechanisms of harness design, while recognizing that real agent behavior is substantially more complex; extending these principles to richer execution dynamics and real world agent systems remains an important direction for future work.

AI USE STATEMENT

We used generative AI tools to aid and polish writing, assist in drafting portions of the manuscript, support the development and verification of mathematical arguments, and assist in creating selected figures and visual elements. All AI-assisted material was reviewed and revised by the authors. All mathematical claims and proofs were independently checked for correctness, and AI-assisted figures were reviewed to ensure consistency with the underlying results and presentation. We take full responsibility for the final content of this work, including all text, claims, proofs, figures, and other artifacts produced with the aid of generative AI.

REFERENCES

Lakshya A Agrawal, Shangyin Tan, Dilara Soylu, Noah Ziems, Rishi Khare, Krista Opsahl-Ong, Arnav Singhvi, Herumb Shandilya, Michael J Ryan, Meng Jiang, et al. Gepa: Reflective prompt evolution can outperform reinforcement learning. *arXiv preprint arXiv:2507.19457*, 2025.

Anthropic. Effective harnesses for long-running agents. https://www.anthropic.com/engineering/effective-harnesses-for-long-running-agents, 2025. Anthropic Engineering Blog, accessed 2025-11-26.

Anthropic. Claude code overview. https://code.claude.com/docs/en/overview, 2026. Official documentation, accessed 2026-03-10.

Browser Use. Browser harness. https://github.com/browser-use/browser-harness, 2026. Browser Harness | Self-healing harness that enables LLMs to complete any task.

Nghi DQ Bui. Building effective ai coding agents for the terminal: Scaffolding, harness, context engineering, and lessons learned. *arXiv preprint arXiv:2603.05344*, 2026.

Yufan Dang, Chen Qian, Xueheng Luo, Jingru Fan, Zihao Xie, Ruijie Shi, Weize Chen, Cheng Yang, Xiaoyin Che, Ye Tian, et al. Multi-agent collaboration via evolving orchestration. *arXiv preprint arXiv:2505.19591*, 2025.

Lutfi Eren Erdogan, Hiroki Furuta, Sehoon Kim, Nicholas Lee, Suhong Moon, Gopala Anumanchipalli, Kurt Keutzer, and Amir Gholami. Plan-and-act: Improving planning of agents for long-horizon tasks. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=ybA4EcMmUZ.

Xueyang Feng, Zhi-Yuan Chen, Yujia Qin, Yankai Lin, Xu Chen, Zhiyuan Liu, and Ji-Rong Wen. Large language model-based human-agent collaboration for complex task solving. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 1336–1357, 2024.

Chrisantha Fernando, Dylan Banarse, Henryk Michalewski, Simon Osindero, and Tim Rocktäschel. Promptbreeder: Self-referential self-improvement via prompt evolution. *arXiv preprint arXiv:2309.16797*, 2023.

Agentic AI Foundation. agents.md. https://github.com/agentsmd/agents.md, 2026. AGENTS.md — a simple, open format for guiding coding agents, accessed 2026-03-12.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

Harbor Framework Team. Harbor: A framework for evaluating and optimizing agents and models in container environments, January 2026. URL https://github.com/harbor-framework/harbor.

Sirui Hong, Mingchen Zhuge, Jonathan Chen, Xiawu Zheng, Yuheng Cheng, Jinlin Wang, Ceyao Zhang, Zili Wang, Steven Ka Shing Yau, Zijuan Lin, et al. Metagpt: Meta programming for a multi-agent collaborative framework. In *The twelfth international conference on learning representations*, 2023.

Eric Horvitz. Principles of mixed-initiative user interfaces. In *International Conference on Human Factors in Computing Systems*, 1999. URL https://api.semanticscholar.org/CorpusID:8943607.

Shengran Hu, Cong Lu, and Jeff Clune. Automated design of agentic systems. *arXiv preprint arXiv:2408.08435*, 2024.

Carlos E Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik Narasimhan. Swe-bench: Can language models resolve real-world github issues? *arXiv preprint arXiv:2310.06770*, 2023.

Ruofan Jin, Zaixi Zhang, Mengdi Wang, and Le Cong. Stella: Self-evolving llm agent for biomedical research. *arXiv preprint arXiv:2507.02004*, 2025.

Andrej Karpathy. autoresearch. https://github.com/karpathy/autoresearch, 2026. AI agents running research on single-GPU nanochat training automatically, accessed 2026-03-26.

Omar Khattab, Arnav Singhvi, Paridhi Maheshwari, Zhiyuan Zhang, Keshav Santhanam, Sri Vardhamanan, Saiful Haq, Ashutosh Sharma, Thomas T Joshi, Hanna Moazam, et al. Dspy: Compiling declarative language model calls into self-improving pipelines. *arXiv preprint arXiv:2310.03714*, 2023.

Yoonho Lee, Roshen Nair, Qizheng Zhang, Kangwook Lee, Omar Khattab, and Chelsea Finn. Meta-harness: End-to-end optimization of model harnesses. *arXiv preprint arXiv:2603.28052*, 2026.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, et al. Self-refine: Iterative refinement with self-feedback. *Advances in neural information processing systems*, 36:46534–46594, 2023.

Mike A. Merrill, Alexander G. Shaw, Nicholas Carlini, Boxuan Li, Harsh Raj, Ivan Bercovich, Lin Shi, Jeong Yeon Shin, Thomas Walshe, E. Kelly Buchanan, et al. Terminal-bench: Benchmarking agents on hard, realistic tasks in command line interfaces, 2026.

Alexander Novikov, Ngân Vũ, Marvin Eisenberger, Emilien Dupont, Po-Sen Huang, Adam Zsolt Wagner, Sergey Shirobokov, Borislav Kozlovskii, Francisco JR Ruiz, Abbas Mehrabian, et al. Alphaevolve: A coding agent for scientific and algorithmic discovery. *arXiv preprint arXiv:2506.13131*, 2025.

OpenAI. Codex cli. https://developers.openai.com/codex/cli/, 2026a. Official documentation, accessed 2026-03-10.

OpenAI. Harness engineering: leveraging Codex in an agent-first world. https://openai.com/index/harness-engineering/, February 2026b. OpenAI Blog, accessed 2026-02-11.

Krista Opsahl-Ong, Michael J Ryan, Josh Purtell, David Broman, Christopher Potts, Matei Zaharia, and Omar Khattab. Optimizing instructions and demonstrations for multi-stage language model programs. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 9340–9366, 2024.

Charles Packer, Vivian Fang, Shishir G. Patil, Kevin Lin, Sarah Wooders, and Joseph E. Gonzalez. Memgpt: Towards llms as operating systems, 2023.

Linyue Pan, Lexiao Zou, Shuo Guo, Jingchen Ni, and Hai-Tao Zheng. Natural-language agent harnesses. *arXiv preprint arXiv:2603.25723*, 2026.

Reid Pryzant, Dan Iter, Jerry Li, Yin Lee, Chenguang Zhu, and Michael Zeng. Automatic prompt optimization with "gradient descent" and beam search. In *Proceedings of the 2023 conference on empirical methods in natural language processing*, pages 7957–7968, 2023.

Yuanhao Qu, Kaixuan Huang, Ming Yin, Kanghong Zhan, Dyllan Liu, Di Yin, Henry C Cousins, William A Johnson, Xiaotong Wang, Mihir Shah, et al. Crispr-gpt for agentic automation of gene-editing experiments. *Nature Biomedical Engineering*, 10(2):245–258, 2026.

Paul Scerri, David V Pynadath, and Milind Tambe. Towards adjustable autonomy for the real world. *Journal of Artificial Intelligence Research*, 17:171–228, 2002.

Yu Shang, Yu Li, Keyu Zhao, Likai Ma, Jiahe Liu, Fengli Xu, and Yong Li. Agentsquare: Automatic llm agent search in modular design space. *arXiv preprint arXiv:2410.06153*, 2024.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. *Advances in neural information processing systems*, 36:8634–8652, 2023.

David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, Yutian Chen, Timothy P. Lillicrap, Fan Hui, L. Sifre, George van den Driessche, Thore Graepel, and Demis Hassabis. Mastering the game of go without human knowledge. *Nature*, 550:354–359, 2017. URL https://api.semanticscholar.org/CorpusID:205261034.

Rich Sutton. The bitter lesson. http://www.incompleteideas.net/IncIdeas/BitterLesson.html, 2019. accessed: 2025-07-19.

Kimi Team, Yifan Bai, Yiping Bao, Y Charles, Cheng Chen, Guanduo Chen, Haiting Chen, Huarong Chen, Jiahao Chen, Ningxin Chen, et al. Kimi k2: Open agentic intelligence. *arXiv preprint arXiv:2507.20534*, 2025.

Vivek Trivedy. Improving deep agents with harness engineering. https://www.langchain.com/blog/improving-deep-agents-with-harness-engineering, 2026. Langchain Blog, accessed 2026-02-17.

Yanbo Wang, Zixiang Xu, Yue Huang, Xiangqi Wang, Zirui Song, Lang Gao, Chenxi Wang, Xiangru Tang, Yue Zhao, Arman Cohan, Xiangliang Zhang, and Xiuying Chen. Dyflow: Dynamic workflow framework for agentic reasoning. *ArXiv*, abs/2509.26062, 2025. URL https://api.semanticscholar.org/CorpusID:281681609.

Yun Wang and Yan Lu. Interaction, process, infrastructure: A unified framework for human-agent collaboration. *arXiv preprint arXiv:2506.11718*, 2025.

Tianbao Xie, Danyang Zhang, Jixuan Chen, Xiaochuan Li, Siheng Zhao, Ruisheng Cao, Toh J Hua, Zhoujun Cheng, Dongchan Shin, Fangyu Lei, et al. Osworld: Benchmarking multimodal agents for open-ended tasks in real computer environments. *Advances in Neural Information Processing Systems*, 37:52040–52094, 2024.

Chuanhao Yan, Fengdi Che, Xuhan Huang, Xu Xu, Xin Li, Yizhi Li, Xingwei Qu, Jingzhe Shi, Zhuangzhuang He, Chenghua Lin, Yaodong Yang, Binhang Yuan, Hang Zhao, Yu Qiao, Bowen Zhou, and Jie Fu. Re:form – reducing human priors in scalable formal software verification with rl in llms: A preliminary study on dafny, 2025. URL https://arxiv.org/abs/2507.16331.

Chengrun Yang, Xuezhi Wang, Yifeng Lu, Hanxiao Liu, Quoc V. Le, Denny Zhou, and Xinyun Chen. Large language models as optimizers. In *International Conference on Learning Representations*, 2024.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*, 2022.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. *Advances in neural information processing systems*, 36:11809–11822, 2023.

Mert Yuksekgonul, Federico Bianchi, Joseph Boen, Sheng Liu, Zhi Huang, Carlos Guestrin, and James Zou. Textgrad: Automatic" differentiation" via text. *arXiv preprint arXiv:2406.07496*, 2024.

Aohan Zeng, Xin Lv, Zhenyu Hou, Zhengxiao Du, Qinkai Zheng, Bin Chen, Da Yin, Chendi Ge, Chenghua Huang, Chengxing Xie, et al. Glm-5: from vibe coding to agentic engineering. *arXiv preprint arXiv:2602.15763*, 2026.

Jiayi Zhang, Jinyu Xiang, Zhaoyang Yu, Fengwei Teng, Xionghui Chen, Jiaqi Chen, Mingchen Zhuge, Xin Cheng, Sirui Hong, Jinlin Wang, et al. Aflow: Automating agentic workflow generation. *arXiv preprint arXiv:2410.10762*, 2024.

Andy Zhou, Kai Yan, Michal Shlapentokh-Rothman, Haohan Wang, and Yu-Xiong Wang. Language agent tree search unifies reasoning acting and planning in language models. *arXiv preprint arXiv:2310.04406*, 2023a.

Shuyan Zhou, Frank F Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, et al. Webarena: A realistic web environment for building autonomous agents. *arXiv preprint arXiv:2307.13854*, 2023b.

Yongchao Zhou, Andrei Ioan Muresanu, Ziwen Han, Keiran Paster, Silviu Pitis, Harris Chan, and Jimmy Ba. Large language models are human-level prompt engineers. In *The eleventh international conference on learning representations*, 2022.

Henry Peng Zou, Wei-Chieh Huang, Yaozu Wu, Chunyu Miao, Dongyuan Li, Aiwei Liu, Yue Zhou, Yankai Chen, Weizhi Zhang, Yangning Li, et al. A call for collaborative intelligence: Why human-agent systems should precede ai autonomy. *arXiv preprint arXiv:2506.09420*, 2025.

A Synthetic Experiments Details

A.1 Task and Metrics

The cumulative-progress task is parameterized by a total target G, a tolerance ϵ, and a per-stage draw budget R. A harness decomposes G into an ordered subgoal sequence $(g_1, \ldots, g_K)$ with $\sum_k g_k = G$ and provides the agent with a pool of action distributions, each a truncated Gaussian specified by (μ, σ, ℓ, u). At stage k, local progress s_k is initialized to zero. The agent draws $z \sim \mathcal{N}(\mu, \sigma^2)$, rejects samples outside $[\ell, u]$, rounds to the nearest integer, and clips back into $[\ell, u]$; the resulting action a updates $s_k \leftarrow s_k + a$. A stage succeeds once $s_k \in [g_k - \epsilon, g_k + \epsilon]$ after at least one draw, fails by overshoot if $s_k > g_k + \epsilon$, and fails by draw limit if R draws are exhausted before the tolerance window is reached. An episode succeeds if and only if all stages succeed.

We report two metrics throughout. *Pass rate* is the fraction of episodes in which all stages succeed and measures whether the harness keeps the run on a recoverable trajectory. *Absolute final bias* is $|\sum_k s_k - G|$ and measures how much terminal accuracy the harness preserves even when the run completes. The two metrics can dissociate, and reporting both separates "the harness lets the agent finish" from "the harness lets the agent finish accurately."

A.2 Granularity–Capability Experiment

This experiment isolates decomposition by varying K while fixing a single action distribution per agent. We use $G = 100$, $\epsilon = 2$, $R = 4$, and sweep $K \in \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 12, 15, 18, 20\}$. The three agents differ only in action scale: Small $(\mu = 6, \sigma = 2, [4, 8])$, Medium $(\mu = 8, \sigma = 3, [5, 11])$, and Large $(\mu = 10, \sigma = 4, [6, 14])$. Small K produces large subgoals that may exceed what an agent can reach within R draws, while large K produces subgoals below the agent's natural action scale, where rounding-and-clipping introduces residual error at every stage. The three agents therefore peak at different K, sweeping out the granularity–capability frontier predicted by Theorem 1.

A.3 Guidance–Evidence Experiment

This experiment isolates guidance quality from action-pool size. We construct a pool of ten distributions indexed by $i = 0, \ldots, 9$, with

$$\mu_i = 4.0 + 1.2i, \qquad \sigma_i = 1.5 + 0.35i, \qquad \ell_i = \max\{1, \lfloor \mu_i - 2.0 \rfloor\}, \qquad u_i = \lceil \mu_i + 2\sigma_i \rceil.$$

At the start of each episode we randomly retain $N \in \{1, \ldots, 10\}$ distributions, then compare three selection policies on the retained pool: aligned guidance (favoring distributions whose mean is closest to the remaining progress), misaligned guidance (favoring distributions whose mean is farthest), and uniform random selection. We use $G = 100$, $K = 5$, $\epsilon = 4$, and $R = 5$. Holding pool size fixed across the three policies isolates the sign of the retention gap from the size of the local action space, which is the comparison Theorem 2 predicts to be decisive.

A.4 Partial Harnessing Experiment

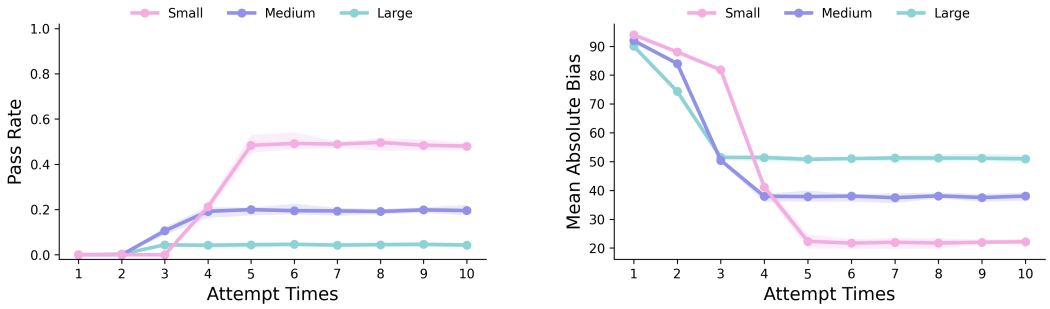
This experiment sweeps a progress slice as in Theorem 3. We fix a chunk size c and a number of scaffolded chunks r, defining the harness

$$\Delta_h = (\underbrace{c, \ldots, c}_{r \text{ chunks}}, G - rc),$$

where $r = 0$ leaves the entire task to the autonomous agent and $r = \lfloor G/c \rfloor$ scaffolds it fully. We use $G = 100$, $\epsilon = 2$, $R = 10$, and the Small, Medium, and Large agents from Appendix A.2. Sweeping r traces the slice objective $F(r)$ for each agent. A stronger agent has lower autonomous tail cost κ_{tail} and reaches $\Delta(r; M) \le c_s$ at smaller r, predicting earlier peaks for stronger agents.

A.5 Additional Control Experiments

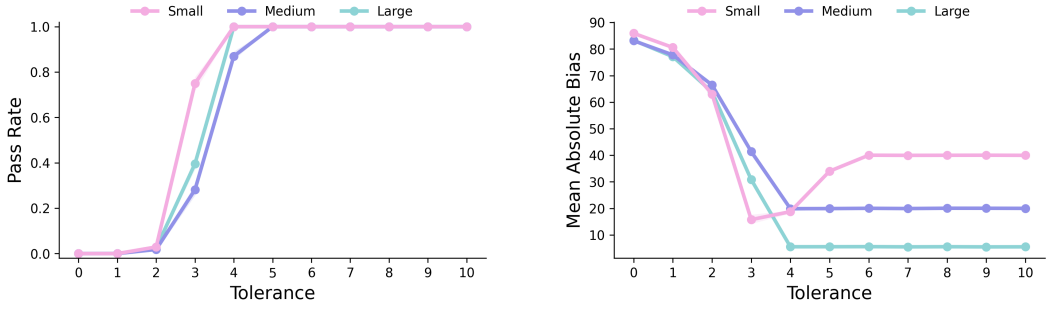
The experiments below vary harness or task parameters that could plausibly explain the patterns above through a simpler mechanism. In each case the result reproduces the alignment predictions rather than displacing them.



(a) Pass rate vs. attempt budget

(b) Final bias vs. attempt budget

Figure 5: **Retry budget improves recoverability but cannot overcome capability mismatch.** Pass rate and final bias both saturate once attempts no longer relax the binding constraint.



(a) Pass rate vs. tolerance

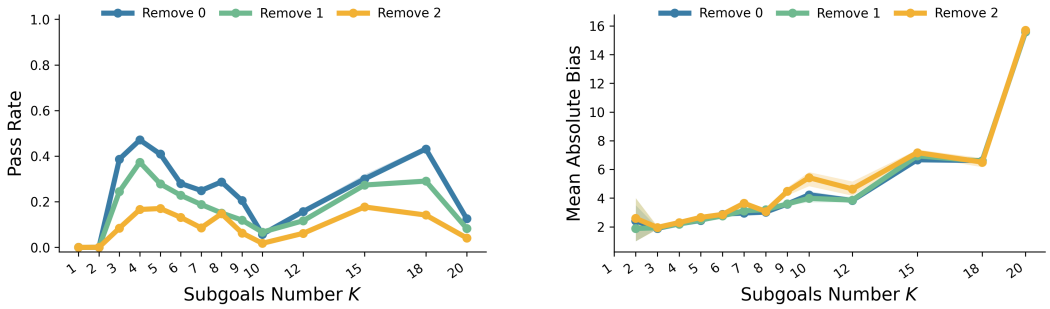
(b) Final bias vs. tolerance

Figure 6: **Tolerance trades recoverability for terminal accuracy.** Pass rate rises sharply with ϵ but final bias grows when tolerance exceeds the agent's natural per-stage variation.

Retry budget. Holding $K = 4$ and the granularity setup of Appendix A.2 fixed, we sweep the per-stage draw budget $R \in \{1, \ldots, 10\}$. Increasing R raises pass rate and lowers final bias, but the curves saturate quickly once the binding failure mode shifts from draw-limit exhaustion to action-scale mismatch and overshoot (Figure 5). Retry budget is therefore a recoverability resource that interacts with granularity rather than substituting for it: more attempts cannot fix a stage whose requested progress lies outside the agent's reachable scales, matching the $-\log M_t$ correction in Theorem 1.

Completion tolerance. Fixing $G = 100$, $K = 10$, and $R = 4$, we sweep $\epsilon \in \{0, 1, \ldots, 10\}$. Larger tolerance raises pass rate by widening each stage's acceptance window, but for agents whose action scale undershoots the subgoal size it accepts premature stage completion and accumulates systematic terminal under-progress (Figure 6). Tolerance therefore trades recoverability against terminal accuracy, and the right setting depends on the agent's action scale rather than functioning as a universal robustness knob.

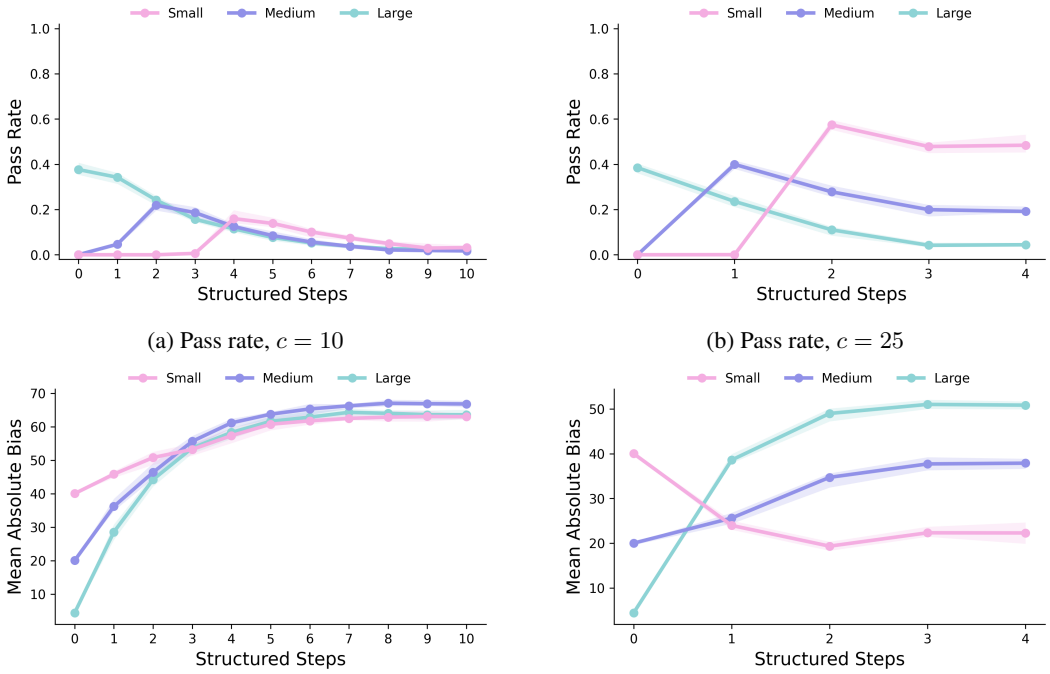
Aggressive guidance pruning. Starting from a three-distribution pool—Model 1 ($\mu = 6, \sigma = 2, [4, 8]$), Model 2 ($\mu = 8, \sigma = 3, [5, 11]$), Model 3 ($\mu = 10, \sigma = 6, [4, 14]$)—we randomly remove $m \in \{0, 1, 2\}$ distributions and sweep K as in Appendix A.2, with the agent greedily selecting the closest-mean distribution at each step. Stronger pruning lowers pass rate and amplifies final bias, with the largest gap appearing under fine decomposition where local action-scale mismatch is most exposed (Figure 7). Pruning is therefore a negative-gap intervention even when individual remaining distributions are reasonable: the retention-gap sign in Theorem 2 depends on which trajectories are removed, not how many.



(a) Pass rate vs. subgoal count under pruning

(b) Final bias vs. subgoal count under pruning

Figure 7: **Aggressive pruning amplifies granularity mismatch.** Removing more action distributions worsens both pass rate and final bias, with the largest gap appearing under fine decomposition where local mismatch is most exposed.



(a) Pass rate, $c = 10$

(b) Pass rate, $c = 25$

(c) Final bias, $c = 10$

(d) Final bias, $c = 25$

Figure 8: **Marginal stopping persists across chunk sizes.** Both pass rate and final bias remain unimodal in r at $c = 10$ and $c = 25$, with peak location shifting as predicted by Theorem 3.

Extended partial-harnessing sweeps. We complement the chunk size $c = 20$ result reported in the main text with $c = 10$ and $c = 25$ (Figure 8). The unimodal shape and the agent-dependent peak location persist across chunk sizes, with finer chunks placing the peak at larger r as predicted by the marginal rule.

B Real World Task Details

B.1 Granularity experiment on Terminal-Bench-2

This appendix specifies the full configuration of the granularity sweep referenced in Section 5.2. The experiment varies the workflow step count $k \in \{1, \ldots, 10\}$ supplied to a fixed solver and measures

Table 1: Selected Terminal-Bench 2 tasks and official resolution rates in Merrill et al. (2026).

Task	Rate	Task	Rate
bn-fit-modify	20.0%	build-cython-ext	20.0%
cancel-async-tasks	20.0%	circuit-fibsqrt	20.0%
compile-compcert	25.0%	overfull-hbox	20.0%
protein-assembly	20.0%	query-optimize	20.0%
sanitize-git-repo	20.0%	winning-avg-corewars	20.0%
adaptive-rejection-sampler	40.0%	feal-differential-cryptanalysis	40.0%
log-summary-date-ranges	40.0%	build-pov-ray	60.0%
extract-elf	60.0%	qemu-startup	60.0%
sparql-university	60.0%	tune-mjcf	60.0%
configure-git-webserver	80.0%	count-dataset-tokens	80.0%
custom-memory-heap-crash	80.0%	financial-document-processor	80.0%
fix-ocaml-gc	80.0%	headless-terminal	80.0%
kv-store-grpc	80.0%	large-scale-text-editing	80.0%
llm-inference-batching-scheduler	80.0%	model-extraction-relu-logits	80.0%
qemu-alpine-ssh	80.0%	sqlite-db-truncate	80.0%
sqlite-with-gcov	80.0%		

pass rate on real software-engineering tasks from `terminal-bench@2.0`, with all non-k factors held constant.

B.1.1 TASKS

We sample tasks from `terminal-bench@2.0` using the official Terminal-Bench leaderboard's `GLM-5` task-level correctness statistics (snapshot: May 2026). The selection rule is to focus on tasks of moderate difficulty for `GLM-5`, down-weighting both near-ceiling tasks (uninformative because already trivial) and near-floor tasks (uninformative because no workflow yields signal), so that granularity effects can be observed in the sensitive middle band. Applying this rule yields a pool of 32 tasks.

B.1.2 VARIABLES AND CONTROLS

The independent variable is the workflow step count $k \in \{1, \ldots, 10\}$. The primary dependent variable is pass rate; we additionally log `episodes_to_success` (number of solver episodes used on successful runs) as a secondary diagnostic. To isolate the effect of k, the following factors are held fixed across all conditions:

Table 2: Configuration for the Terminal-Bench-2 workflow-step experiment.

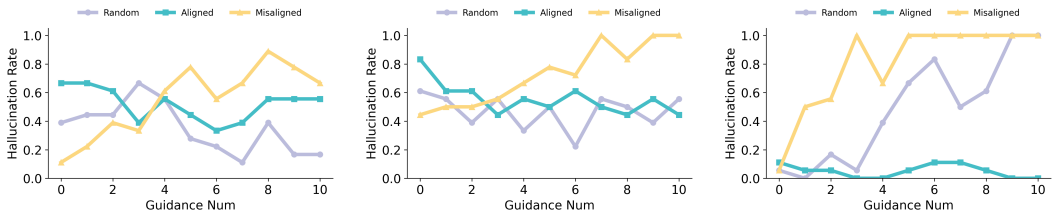
Component	**Setting**
Planner	`openrouter/qwen/qwen3.5-plus-02-15`
Solver	`openrouter/z-ai/glm-5`
Execution backend	Harbor Docker (local)
Episode budget	50 episodes per run
Attempts per cell	`N_ATTEMPTS=1` per (task, k)
Verifier	Official Terminal-Bench verifier (per task)
Prompt scaffold	Fixed; only the workflow block varies with k

All experiments are run locally on a MacBook with an Apple M4 chip.

B.1.3 PIPELINE

Each (task, k) cell follows a two-stage pipeline.

Stage 1: Workflow Generation. For each task, the planner generates one workflow per granularity $k \in \{1, \ldots, 10\}$, producing exactly k ordered steps. The planner receives the raw Terminal-Bench



(a) Qwen3.5-plus-02-15 (b) GPT-OSS-120B (c) Gemini-3-flash-preview

Figure 9: **Granularity–capability alignment in a controlled addition task.** (a) Pass rate varies with the number of subgoals and depends on the match between harness granularity and agent capability. (b) The mean absolute final bias increases as the number of subgoals grows, showing that overly fine task decomposition can accumulate larger terminal error.

task instruction together with a fixed generation prompt that requests k ordered intermediate actions or verification points and explicitly prohibits fabricated command outputs or verifier results unsupported by the instruction. Each task's workflows are saved as a single JSON object indexed by k.

Stage 2: Solver Evaluation. For each (task, k), the solver receives a prompt of the form

$$\Pi_k(x) \;=\; \texttt{Template}\big(x,\, \texttt{Format}(W_k(x))\big),$$

where x is the task instruction, $W_k(x)$ is the k-step workflow, and `Template` is a fixed scaffold containing three blocks: `## task` (verbatim x), `## guidance` (a four-bullet k-invariant rubric on inspecting the verifier, reproducing failures, and minimal local edits), and `## workflow` (the numbered k-step list). The solver is then run once under the fixed Harbor Docker environment with `MAX_EPISODES=50`, and success is determined by the verifier's reward.

B.1.4 Metric

For task c, step count k, and trial r, define the per-cell outcome as

$$s_{c,k,r} \;=\; \mathbf{1}\{\texttt{verifier_result.rewards.reward} > 0\}\,.$$

The aggregate pass rate at granularity k is

$$\text{PassRate}(k) \;=\; \frac{n_{\text{succ}}(k)}{n_{\text{valid}}(k)},$$

where $n_{\text{valid}}(k)$ is the number of evaluated cells at k (excluding missing or errored runs) and $n_{\text{succ}}(k)$ is the number of cells with $s_{c,k,r} = 1$. For successful cells, we additionally report `episodes_to_success`, taken from the agent's `n_episodes` metadata.

B.2 Plot-Reasoning Hallucination Setup

We further evaluate harness guidance in a visibility-limited chart reasoning task. This experiment complements the controlled simulations by testing whether additional guidance reduces or amplifies unsupported claims in a more realistic analysis setting. The central question is: when the model only observes a truncated Plotly representation rather than the rendered figure, how does the amount and type of guidance affect hallucination?

B.2.1 Task Construction

Each task is a chart-analysis problem. Instead of showing the model the rendered image, we provide only the raw Plotly trace representation, denoted as `plotly_repr`. This input contains partial structural information about the chart, but it does not provide full visual access to the figure. Therefore, the model must reason under limited observability.

We generate six chart types:

`box`, `line`, `scatter`, `histogram`, `violin`, `heatmap`.

For each chart type, we generate a large synthetic chart and store its Plotly representation as a task instance. The random seed is fixed to ensure reproducibility. Each task contains a chart name and a string-valued `plotly_repr`, which is later inserted into the analysis prompt.

B.2.2 Guidance Conditions

The harness augments the chart-analysis prompt with a set of textual guidance rules. We vary the number of injected rules, denoted by k. When $k = 0$, the model receives no additional guidance beyond the base prompt. As k increases, the harness imposes more requirements on how the model should analyze the chart.

We compare three guidance sources. The first source is the original guidance set, which contains general analysis requirements. The second source is an anti-hallucination guidance set, which explicitly instructs the model to respect the limited visibility of the Plotly representation and avoid unsupported visual claims. The third source is a mixed condition, where guidance rules are sampled from the two sources with equal probability. Thus, the experiment separates the effect of guidance quantity from the effect of guidance quality.

B.2.3 Generation and Judging Pipeline

Each experimental case follows the same two-stage pipeline. First, the analysis model receives the base chart-analysis prompt, the task-specific `plotly_repr`, and k sampled guidance rules. It then produces a natural-language analysis of the chart. Second, the judge model reads both the input context and the generated analysis, and assigns one of three labels:

$$\texttt{GROUNDED}, \quad \texttt{HALLUCINATION}, \quad \texttt{ERROR}.$$

A response is labeled `GROUNDED` if the analysis stays within what can be reasonably inferred from the provided representation. It is labeled `HALLUCINATION` if it makes unsupported claims, overstates visual details, or fails to acknowledge the limited observability of the chart. Cases with API failures or invalid outputs are labeled `ERROR` and excluded from the valid denominator.

B.2.4 Metrics

For each guidance count k, we aggregate results across chart types and trials. Let $n_{\text{valid}}(k)$ be the number of non-error cases, and let $n_{\text{hall}}(k)$ be the number of cases labeled as hallucination. The main metric is the hallucination rate:

$$\text{HallucinationRate}(k) = \frac{n_{\text{hall}}(k)}{n_{\text{valid}}(k)}.$$

We also report the number of error cases to ensure that a lower hallucination rate is not caused by a large number of invalid model outputs.

B.2.5 Experimental Variables

The independent variables are the guidance count k and the guidance source. By default, we evaluate

$$k \in \{0, 1, \ldots, 10\},$$

with three trials for each k and each chart type. For a fair comparison, the analysis model, judge model, task set, random seed, and tested values of k are kept fixed across guidance strategies.

B.2.6 Interpretation

This experiment is designed to test whether stronger guidance is always better. If increasing k consistently lowers hallucination rate, then additional guidance acts as useful inference-time constraint. If hallucination rate increases with k, then excessive guidance may encourage the model to satisfy more textual requirements than the limited evidence can support. Comparing the original, anti-hallucination, and mixed guidance conditions further distinguishes whether the outcome is driven by the number of guidance rules or by their alignment with the available evidence.

Table 3: Calibration of step-level confidence for predicting eventual task success.

Method	Success AUC ↑	ECE ↓	Brier ↓	Log loss ↓
Raw	0.718	0.341	0.310	0.879
Logistic	0.718	0.052	0.184	0.551
Isotonic	0.717	**0.029**	**0.181**	**0.544**

C Additional Validation and Robustness Analyses

This section provides additional analyses on the practical measurability of the theoretical quantities introduced in our framework, the robustness of our Terminal-Bench evaluation, the sensitivity to workflow and solver randomness, the generality of partial harnessing, and the reliability of the hallucination judge.

C.1 Practical Estimation of Theoretical Quantities

The theoretical quantities in our framework can be estimated from the success continuation probability

$$p_t := \Pr(\text{task succeeds} \mid K_t),$$

where K_t denotes the execution prefix up to stage t. In principle, p_t can be estimated by repeatedly continuing from the same prefix. Recoverability, retention gap, scaffold cost, and tail risk can then be computed from the corresponding continuation probabilities.

Since repeated rollouts are expensive in realistic agent settings, we use the agent's step-level confidence as a proxy for p_t and calibrate it against observed task outcomes.

Calibration substantially improves probability estimation shown in Table 3; for example, isotonic calibration reduces ECE from 0.341 to 0.029. This supports calibrated step-level confidence as a practical proxy for continuation success probability.

Using confidence for workflow refinement. The framework does not imply a universal optimal number of workflow stages, since the appropriate decomposition depends jointly on the task, model, and human-designed harness. Instead, the estimated continuation probability provides stage-wise feedback. A sharp decrease in calibrated continuation probability identifies stages at which the current harness may be counterproductive or insufficient.

The same quantity yields a practical stopping heuristic. Suppose a candidate additional scaffold transforms prefix K_t into K_{t+1}. Extending the harness is useful only when the reduction in residual execution risk exceeds the additional scaffold cost. Operationally, this can be approximated by comparing calibrated continuation probabilities before and after introducing the additional scaffold. If further scaffolding no longer improves the estimated completion probability, the harness can be stopped.

C.2 Robustness of the Terminal-Bench Evaluation

To test whether this filtering choice drives the observed trends, we report results both before and after filtering. All experiments are repeated three times, and we report the average performance together with the empirical variance.

Performance is non-monotonic in workflow depth, and adding more workflow stages does not consistently improve agent performance shown in Table 4. The same phenomenon appears for both Qwen3.5-Plus and Kimi-K2.5, suggesting that it is not specific to a single solver.

C.3 Sensitivity to Workflow and Solver Randomness

Agent performance can vary because of at least two distinct sources of randomness: the workflow itself and the stochastic execution of the solver. We study these effects separately by varying one source while controlling the other.

Table 4: Terminal-Bench 2 robustness check with GLM-5. The qualitative dependence on workflow depth K remains similar under both evaluation sets.

Model	Evaluation	$K = 2$	$K = 4$	$K = 6$	$K = 8$	$K = 10$
Qwen3.5-Plus	Mean	24.7%	33.3%	40.9%	34.4%	31.2%
	Var	0.136	0.158	0.127	0.168	0.129
Kimi-K2.5	Mean	35.5%	35.5%	36.6%	36.6%	31.2%
	Var	0.157	0.121	0.139	0.139	0.157

Table 5: Sensitivity to workflow-generation randomness on AIME with Qwen3-4B.

K	Mean	Variance
2	24.6%	0.075
4	20.8%	0.064
6	31.0%	0.087
8	34.4%	0.110
10	29.2%	0.099

Table 6: Sensitivity to solver-generation randomness on AIME using Qwen3-4B.

K	Mean	Variance
2	27.8%	0.0778
4	32.2%	0.1222
6	45.6%	0.0778
8	48.9%	0.1000
10	45.6%	0.0778

Because this analysis requires many repeated executions, we conduct the controlled experiment on AIME with Qwen3-4B.

Workflow-generation randomness. For each workflow depth K, we independently generate three workflows. We aggregate solver outcomes using majority voting in order to reduce execution-level stochasticity and isolate the variation caused by workflow generation. Table 5 reports the mean and empirical variance across these runs.

Solver-generation randomness. We next fix the workflow and execute the solver 16 times using independent generations. Table 6 reports the mean and empirical variance across these runs.

Both experiments exhibit non-negligible stochastic variation, but the main pattern remains: intermediate workflow depths can outperform both shallower and deeper workflows. This suggests that the non-monotonic relationship between harness strength and task performance cannot be explained solely by a particular workflow sample or solver generation.

C.4 Multi-Task Evaluation of Partial Harnessing

Our framework predicts that fully specifying every stage of execution need not be optimal. A harness can be useful when it improves difficult portions of the trajectory while leaving parts that the model already handles well autonomous. We therefore extend the partial-harnessing experiment beyond the case study in the main text.

For each Terminal-Bench 2 task, we vary the amount of the workflow that is explicitly scaffolded. Table 7 reports task success rates for scaffolded chunks of different sizes.

The optimal amount of scaffolding varies substantially across tasks. For example, `sanitize-git-repo` performs best with a scaffolded chunk of 8, whereas `extract-elf` performs best at 6 and `llm-inference-batching-scheduler` achieves its highest performance at both shallow and full scaffolding. Aggregated performance is also non-monotonic in scaffold size.

These results support the view that partial harnessing is not restricted to an isolated case study. Rather, different tasks benefit from different degrees and locations of intervention, motivating adaptive rather than uniformly maximal scaffolding.

Table 7: Multi-task evaluation of partial harnessing on Terminal-Bench 2. Columns indicate the scaffolded chunk size. No single scaffold depth dominates across all tasks.

Task	2	4	6	8	10
bn-fit-modify	80.0%	40.0%	60.0%	40.0%	80.0%
build-cython-ext	20.0%	20.0%	20.0%	0.0%	0.0%
extract-elf	20.0%	20.0%	40.0%	20.0%	20.0%
feal-differential-cryptanalysis	40.0%	20.0%	20.0%	20.0%	20.0%
fix-ocaml-gc	0.0%	20.0%	20.0%	40.0%	0.0%
llm-inference-batching-scheduler	100.0%	60.0%	60.0%	60.0%	100.0%
model-extraction-relu-logits	0.0%	20.0%	40.0%	20.0%	0.0%
qemu-alpine-ssh	20.0%	20.0%	20.0%	0.0%	0.0%
sanitize-git-repo	40.0%	40.0%	20.0%	60.0%	0.0%
sparql-university	60.0%	40.0%	60.0%	20.0%	80.0%
Overall	**38.0%**	**30.0%**	**36.0%**	**28.0%**	**30.0%**

Table 8: Calibration of hallucination-judge confidence against human annotations.

Method	Success AUC ↑	ECE ↓	Brier ↓	Log loss ↓
Raw	**0.948**	0.102	0.087	0.555
Isotonic	0.916	**0.058**	**0.078**	**0.246**

C.5 Reliability of the Hallucination Judge

The hallucination experiments in the main paper rely on llm-as-a-judge. We conduct two additional analyses to test whether the resulting measurements are reliable: confidence calibration against human annotations and direct agreement comparisons across multiple judges.

C.5.1 Calibration Against Human Annotations

We first calibrate the confidence produced by the Qwen3.5-Plus hallucination judge against human labels. Table 8 compares the raw confidence with isotonic calibration.

Isotonic calibration substantially improves probability calibration, reducing ECE from 0.102 to 0.058 and log loss from 0.555 to 0.246. The slight reduction in AUC is expected because calibration primarily targets probability accuracy rather than ranking performance.

C.5.2 Cross-Judge Robustness

We further compare four automatic evaluation procedures: a single Qwen3.5-Plus judge, majority voting over Qwen3.5-Plus judgments, Kimi-K2.5, and the isotonic-calibrated Qwen3.5-Plus judge.

Despite modest numerical variation, all judges yield the same qualitative pattern shown in Table 9. Good guidance produces little hallucination, Bad guidance produces substantially higher hallucination rates that increase with workflow depth, and Mixed guidance lies between these two regimes.

C.5.3 Human Agreement and Judge Error Analysis

Finally, we directly compare automatic judgments with independently collected human annotations. Table 10 reports raw agreement, Cohen's κ, and the false-positive and false-negative counts relative to the human consensus label.

Human annotators achieve 86.0% agreement with Cohen's $\kappa = 0.703$. All automatic judges exhibit high agreement with the human consensus, with Cohen's κ between 0.807 and 0.847. Moreover, the observed automatic-judge errors are false positives rather than false negatives, indicating that the judges are conservative in detecting hallucinations in this evaluation.

Together, the calibration, cross-judge comparison, and human agreement results suggest that the hallucination trends reported in the main text are not an artifact of a particular automated judge.

Table 9: Hallucination rates under different guidance conditions and judge configurations. The qualitative ordering across Good, Bad, and Mixed guidance is consistent across judges.

Guidance	Judge	$K = 2$	$K = 4$	$K = 6$	$K = 8$	$K = 10$
Good	Qwen3.5-Plus Vote	5.6%	0.0%	0.0%	5.6%	0.0%
	Qwen3.5-Plus	0.0%	0.0%	0.0%	5.6%	0.0%
	Kimi-K2.5	0.0%	0.0%	0.0%	5.6%	0.0%
	Qwen3.5-Plus Isotonic	0.0%	0.0%	0.0%	5.6%	0.0%
Bad	Qwen3.5-Plus Vote	55.6%	66.7%	100.0%	100.0%	100.0%
	Qwen3.5-Plus	55.6%	66.7%	100.0%	100.0%	100.0%
	Kimi-K2.5	61.1%	70.6%	100.0%	100.0%	100.0%
	Qwen3.5-Plus Isotonic	50.0%	66.7%	100.0%	100.0%	100.0%
Mixed	Qwen3.5-Plus Vote	16.7%	44.4%	83.3%	66.7%	100.0%
	Qwen3.5-Plus	16.7%	38.9%	82.4%	66.7%	100.0%
	Kimi-K2.5	6.2%	46.7%	83.3%	53.8%	100.0%
	Qwen3.5-Plus Isotonic	16.7%	33.3%	82.4%	61.1%	100.0%

Table 10: Human agreement and hallucination-judge error analysis.

Method	Agreement ↑	Cohen's κ ↑	FP ↓	FN ↓
Human 1 vs. Human 2	86.0%	0.703	–	–
Qwen3.5-Plus vs. Human Consensus	90.7%	0.807	8	0
Qwen3.5-Plus Vote vs. Human Consensus	91.9%	0.830	7	0
Kimi-K2.5 vs. Human Consensus	92.6%	0.847	6	0

D Omitted Proofs

D.1 Recoverability Construction and Process Loss

This appendix formalizes the chain-rule factorization $P_h(\mathsf{Succ}_x(\tau_h) \mid x) = \prod_t \bar{p}_t(h; x)$ stated without proof in Section 3, on which all subsequent stagewise bounds rely. The factorization requires a precise notion of when an intermediate prefix can still reach a successful continuation, so Appendix D.1.1 first defines the recoverable prefix sets $\mathcal{R}_t(x, h)$, identifies the stagewise event B_t with membership in $\mathcal{R}_t$, and isolates the goal-consistency condition that aligns these intermediate notions with final success. Under this condition, Appendix D.1.2 then derives the factorization by chain rule and converts it into the additive log form invoked by the bounds in Appendix D.2 and Appendix D.3.

D.1.1 Recoverable Prefixes and Goal Consistency

Recall from Section 3 the completed prefix

$$K_t := (x, g_1, \tau_1, \ldots, g_t, \tau_t), \qquad K_0 := x,$$

which records the full execution trace through stage t. For each intermediate stage $t < T$, define the recoverable set

$$\mathcal{R}_t(x, h) := \{K_t : \exists\, (\tau_{t+1}, \ldots, \tau_T) \text{ such that } \mathsf{Succ}_x(\tau_h)\}\,,$$

the set of completed prefixes from which some continuation under the remaining plan $(g_{t+1}, \ldots, g_T)$ reaches $y^\star(x)$. At the terminal stage we set $\mathcal{R}_T(x, h) := \{K_T : \mathsf{Succ}_x(\tau_h)\}$, so a terminal prefix is recoverable iff it is successful. The stagewise recoverability event of Section 3 is then

$$B_t := \{K_t \in \mathcal{R}_t(x, h)\}, \qquad t = 1, \ldots, T,$$

and the cumulative events $E_t := \bigcap_{s \le t} B_s$ form a nested sequence $E_T \subseteq \cdots \subseteq E_1$.

We call the harness plan *goal-consistent* if

$$E_T \equiv \mathsf{Succ}_x(\tau_h),$$

that is, if remaining recoverable through every stage is equivalent to final success. This rules out decompositions whose intermediate recoverability notions are misaligned with the final answer, and it is the regime under which the chain-rule identity below holds.

D.1.2 CHAIN-RULE IDENTITY

Under goal consistency, $P_h(\mathsf{Succ}_x(\tau_h) \mid x) = P_h(E_T \mid x)$. Applying the chain rule to the nested events $E_1, \ldots, E_T$ and using $E_t = E_{t-1} \cap B_t$ together with $E_{t-1} = B_{<t}$,

$$P_h(\mathsf{Succ}_x(\tau_h) \mid x) = \prod_{t=1}^{T} P_h(B_t \mid B_{<t}, x) = \prod_{t=1}^{T} \bar{p}_t(h; x),$$

which is the factorization stated in Section 3. Taking negative logarithms,

$$-\log P_h(\mathsf{Succ}_x(\tau_h) \mid x) = \sum_{t=1}^{T} -\log \bar{p}_t(h; x),$$

so the process loss is induced by the primitive final-success objective rather than added on top of it. Each summand penalizes a low conditional probability of preserving recoverability after one harness-level plan step, and this is the form used by the bounds in Appendix D.2 and Appendix D.3.

D.2 GRANULARITY–CAPABILITY ALIGNMENT

This appendix gives the formal statement and proof for the granularity–capability alignment principle of Section 4.1. The result quantifies how the harness's choice of sub-goal scale, combined with the agent's per-stage attempt budget, controls the final success probability through a per-stage mismatch penalty. Theorem 4 states the general bound under prefix-dependent stage quantities, taking infimum over all recoverable prefixes; Remark 5 then specializes this to the deterministic regime that recovers the simplified form stated in the main text. Building on this specialization, Corollary 6 characterizes when a uniform T-step decomposition incurs no mismatch penalty: the sub-goal size L_x/T must lie in the union of cumulative controllability windows reachable within the attempt budget, with deviations on either side producing the coarse-grained and fine-grained failure regimes observed in the main text.

Theorem 4 (Granularity–capability alignment bound with finite attempts)**.** *Fix a task x and a harness $h = (\kappa, \lambda, \psi)$. Let*

$$\Delta_h(x) = (g_1, \ldots, g_T), \qquad T = T_h(x),$$

be the harness-induced sub-goal sequence, and let $\tau_h(x) = (g_1, \tau_1, \ldots, g_T, \tau_T)$ be the harness-conditioned execution trajectory. Recall that the completed prefix after the first t harness-level sub-goals is

$$K_t = (x, g_1, \tau_1, \ldots, g_t, \tau_t), \qquad K_0 := x.$$

For each stage t and each completed prefix K_{t-1} satisfying $B_{<t}$, suppose there exists a latent progress coordinate ϕ_x. Define the intended latent progress required by the harness-specified sub-goal g_t as

$$\ell_t(K_{t-1}) := \phi_x(g_t) - \phi_x(K_{t-1}).$$

At stage t, suppose the agent may make at most M_t primitive attempts before the stage fails. For $m = 1, \ldots, M_t$, let $Z_{t,m}$ denote the cumulative latent progress made after m primitive attempts within stage t, measured from the prefix K_{t-1}.

Assume the following conditions hold for every stage t and almost surely over completed prefixes K_{t-1} satisfying $B_{<t}$.

(i) Finite-attempt recoverability tube. *There exists a tolerance $\varepsilon_t(K_{t-1}) \geq 0$ such that preserving recoverability after executing sub-goal g_t requires landing near the intended milestone within the finite attempt budget:*

$$B_t \subseteq \bigcup_{m=1}^{M_t} \{|Z_{t,m} - \ell_t(K_{t-1})| \leq \varepsilon_t(K_{t-1})\}.$$

(ii) Cumulative controllability windows. *For each $m = 1, \dots, M_t$, there exist quantities*

$$\mu^-_{t,m}(K_{t-1}) \le \mu^+_{t,m}(K_{t-1}), \qquad \sigma_{t,m}(K_{t-1}) > 0,$$

and a mean

$$\mu_{t,m}(K_{t-1}) \in [\mu^-_{t,m}(K_{t-1}), \mu^+_{t,m}(K_{t-1})],$$

such that, for all $u \ge 0$,

$$P_h\left(Z_{t,m} - \mu_{t,m}(K_{t-1}) \ge u \mid K_{t-1}\right) \le \exp\left(-\frac{u^2}{2\sigma^2_{t,m}(K_{t-1})}\right),$$

and

$$P_h\left(\mu_{t,m}(K_{t-1}) - Z_{t,m} \ge u \mid K_{t-1}\right) \le \exp\left(-\frac{u^2}{2\sigma^2_{t,m}(K_{t-1})}\right).$$

(iii) Boundary loss. *Crossing the t-th harness-specified sub-goal boundary incurs an irreducible coordination loss $\eta_t(K_{t-1}) \ge 0$, so that*

$$P_h(B_t \mid K_{t-1}) \le \exp\big(-\eta_t(K_{t-1})\big) P_h\left(\bigcup_{m=1}^{M_t} \{|Z_{t,m} - \ell_t(K_{t-1})| \le \varepsilon_t(K_{t-1})\} \mid K_{t-1}\right).$$

For each $m = 1, \dots, M_t$, define the prefix-wise finite-attempt mismatch

$$a_{t,m}(K_{t-1}) := \frac{\left(d\left(\ell_t(K_{t-1}), [\mu^-_{t,m}(K_{t-1}), \mu^+_{t,m}(K_{t-1})]\right) - \varepsilon_t(K_{t-1})\right)^2_+}{2\sigma^2_{t,m}(K_{t-1})}.$$

Define the best finite-attempt mismatch at stage t as

$$\rho^{(M)}_t(K_{t-1}) := \min_{1 \le m \le M_t} a_{t,m}(K_{t-1}).$$

Let

$$\mathcal{R}_{t-1}(h, x) := \mathrm{supp}_{P_h(\cdot \mid B_{<t}, x)}(K_{t-1})$$

denote the set of recoverable prefixes that can arise before stage t. Define the stage-wise certified finite-attempt granularity loss as

$$\gamma^{(M)}_t(h; x) := \inf_{K_{t-1} \in \mathcal{R}_{t-1}(h,x)} \left\{\eta_t(K_{t-1}) + \left(\rho^{(M)}_t(K_{t-1}) - \log M_t\right)_+\right\}.$$

Then the final success probability satisfies

$$-\log P_h(S(\tau_h, x) = 1 \mid x) \ge \sum_{t=1}^{T_h(x)} \gamma^{(M)}_t(h; x).$$

Equivalently,

$$P_h(S(\tau_h, x) = 1 \mid x) \le \exp\left(-\sum_{t=1}^{T_h(x)} \gamma^{(M)}_t(h; x)\right).$$

Remark 5 (Deterministic stage quantities). When the prefix-dependent quantities

$$\ell_t(K_{t-1}),\ \mu^{\pm}_{t,m}(K_{t-1}),\ \sigma_{t,m}(K_{t-1}),\ \varepsilon_t(K_{t-1}),\ \eta_t(K_{t-1})$$

are constant on $B_{<t}$—a regime we denote without the K_{t-1} argument—the infimum in $\gamma^{(M)}_t(h; x)$ collapses to its argument, and Theorem 4 reduces to

$$P_h(S(\tau_h, x) = 1 \mid x) \le \exp\left(-\sum_{t=1}^{T_h(x)} \left[\eta_t + (\rho^{(M)}_t - \log M_t)_+\right]\right),$$

which is the form stated in the main text. The two corollaries below operate in this regime.

Proof of Theorem 4. Fix a task x and a harness h, and write $T = T_h(x)$. All probabilities are taken under the harness-conditioned execution distribution induced by q_h.

Fix a stage t and a completed prefix K_{t-1} satisfying $B_{<t}$. For each $m = 1, \dots, M_t$, define the finite-attempt tube event

$$E_{t,m}(K_{t-1}) := \{|Z_{t,m} - \ell_t(K_{t-1})| \le \varepsilon_t(K_{t-1})\}.$$

The event that the agent hits the recoverability tube within the finite attempt budget is

$$E_t^{(M)}(K_{t-1}) := \bigcup_{m=1}^{M_t} E_{t,m}(K_{t-1}).$$

By the boundary-loss assumption,

$$P_h(B_t \mid K_{t-1}) \le \exp\big(-\eta_t(K_{t-1})\big) P_h(E_t^{(M)}(K_{t-1}) \mid K_{t-1}).$$

It remains to upper bound the finite-attempt tube probability $P_h(E_t^{(M)}(K_{t-1}) \mid K_{t-1})$.

Fix an attempt count $m \in \{1, \dots, M_t\}$. Let

$$I_{t,m}(K_{t-1}) := [\mu_{t,m}^-(K_{t-1}), \mu_{t,m}^+(K_{t-1})],$$

and define

$$d_{t,m}(K_{t-1}) := d\left(\ell_t(K_{t-1}), I_{t,m}(K_{t-1})\right).$$

We first show that

$$P_h(E_{t,m}(K_{t-1}) \mid K_{t-1}) \le \exp\left(-a_{t,m}(K_{t-1})\right).$$

There are three cases.

First suppose

$$\ell_t(K_{t-1}) > \mu_{t,m}^+(K_{t-1}).$$

Then

$$d_{t,m}(K_{t-1}) = \ell_t(K_{t-1}) - \mu_{t,m}^+(K_{t-1}).$$

Since

$$\mu_{t,m}(K_{t-1}) \le \mu_{t,m}^+(K_{t-1}),$$

on the event $E_{t,m}(K_{t-1})$ we have

$$Z_{t,m} \ge \ell_t(K_{t-1}) - \varepsilon_t(K_{t-1}).$$

Therefore,

$$Z_{t,m} - \mu_{t,m}(K_{t-1}) \ge \ell_t(K_{t-1}) - \varepsilon_t(K_{t-1}) - \mu_{t,m}^+(K_{t-1}) = d_{t,m}(K_{t-1}) - \varepsilon_t(K_{t-1}).$$

If

$$d_{t,m}(K_{t-1}) > \varepsilon_t(K_{t-1}),$$

the upper-tail sub-Gaussian bound gives

$$P_h(E_{t,m}(K_{t-1}) \mid K_{t-1}) \le \exp\left(-\frac{\left(d_{t,m}(K_{t-1}) - \varepsilon_t(K_{t-1})\right)^2}{2\sigma_{t,m}^2(K_{t-1})}\right).$$

If

$$d_{t,m}(K_{t-1}) \le \varepsilon_t(K_{t-1}),$$

the desired bound is trivial because the right-hand side equals 1.

Second suppose

$$\ell_t(K_{t-1}) < \mu_{t,m}^-(K_{t-1}).$$

Then

$$d_{t,m}(K_{t-1}) = \mu_{t,m}^-(K_{t-1}) - \ell_t(K_{t-1}).$$

Since

$$\mu_{t,m}(K_{t-1}) \ge \mu_{t,m}^-(K_{t-1}),$$

on the event $E_{t,m}(K_{t-1})$ we have

$$Z_{t,m} \le \ell_t(K_{t-1}) + \varepsilon_t(K_{t-1}).$$

Therefore,

$$\mu_{t,m}(K_{t-1}) - Z_{t,m} \ge \mu^-_{t,m}(K_{t-1}) - \ell_t(K_{t-1}) - \varepsilon_t(K_{t-1}) = d_{t,m}(K_{t-1}) - \varepsilon_t(K_{t-1}).$$

The lower-tail sub-Gaussian bound gives

$$P_h(E_{t,m}(K_{t-1}) \mid K_{t-1}) \le \exp\left(-\frac{(d_{t,m}(K_{t-1}) - \varepsilon_t(K_{t-1}))_+^2}{2\sigma_{t,m}^2(K_{t-1})}\right).$$

Third suppose

$$\ell_t(K_{t-1}) \in I_{t,m}(K_{t-1}).$$

Then $d_{t,m}(K_{t-1}) = 0$, and the desired bound reduces to

$$P_h(E_{t,m}(K_{t-1}) \mid K_{t-1}) \le 1,$$

which is immediate.

Combining the three cases, for every $m = 1, \dots, M_t$,

$$P_h(E_{t,m}(K_{t-1}) \mid K_{t-1}) \le \exp\left(-a_{t,m}(K_{t-1})\right).$$

Now apply the union bound over the finite attempt budget:

$$P_h(E_t^{(M)}(K_{t-1}) \mid K_{t-1}) = P_h\left(\bigcup_{m=1}^{M_t} E_{t,m}(K_{t-1}) \mid K_{t-1}\right) \le \sum_{m=1}^{M_t} P_h(E_{t,m}(K_{t-1}) \mid K_{t-1}).$$

Using the previous bound,

$$P_h(E_t^{(M)}(K_{t-1}) \mid K_{t-1}) \le \sum_{m=1}^{M_t} \exp\left(-a_{t,m}(K_{t-1})\right).$$

Since

$$\rho_t^{(M)}(K_{t-1}) = \min_{1 \le m \le M_t} a_{t,m}(K_{t-1}),$$

we have

$$\sum_{m=1}^{M_t} \exp\left(-a_{t,m}(K_{t-1})\right) \le M_t \exp\left(-\rho_t^{(M)}(K_{t-1})\right).$$

Therefore,

$$P_h(E_t^{(M)}(K_{t-1}) \mid K_{t-1}) \le \min\left\{1,\ M_t \exp\left(-\rho_t^{(M)}(K_{t-1})\right)\right\}.$$

Equivalently,

$$P_h(E_t^{(M)}(K_{t-1}) \mid K_{t-1}) \le \exp\left(-\left(\rho_t^{(M)}(K_{t-1}) - \log M_t\right)_+\right).$$

Combining this with the boundary-loss assumption gives

$$P_h(B_t \mid K_{t-1}) \le \exp\left(-\eta_t(K_{t-1}) - \left(\rho_t^{(M)}(K_{t-1}) - \log M_t\right)_+\right).$$

By the definition of $\gamma_t^{(M)}(h; x)$, for every $K_{t-1} \in \mathcal{R}_{t-1}(h, x)$,

$$\eta_t(K_{t-1}) + \left(\rho_t^{(M)}(K_{t-1}) - \log M_t\right)_+ \ge \gamma_t^{(M)}(h; x).$$

Hence, almost surely on $B_{<t}$,

$$P_h(B_t \mid K_{t-1}) \le \exp\big(-\gamma_t^{(M)}(h; x)\big).$$

Taking expectation over K_{t-1} conditional on $B_{<t}$ gives

$$\bar{p}_t(h;x) = P_h(B_t \mid B_{<t}, x) \le \exp\bigl(-\gamma_t^{(M)}(h;x)\bigr).$$

By the recoverability factorization in Section 3,

$$P_h(S(\tau_h, x) = 1 \mid x) = \prod_{t=1}^{T_h(x)} \bar{p}_t(h;x).$$

Therefore,

$$P_h(S(\tau_h, x) = 1 \mid x) \le \prod_{t=1}^{T_h(x)} \exp\bigl(-\gamma_t^{(M)}(h;x)\bigr) = \exp\left(-\sum_{t=1}^{T_h(x)} \gamma_t^{(M)}(h;x)\right).$$

Taking negative logarithms proves the claim. □

Corollary 6 (Failure of overly coarse and overly fine decompositions)**.** *Adopt the deterministic regime of Remark 5, and assume in addition that the harness uses a uniform T-step decomposition with $\ell_t = L_x/T$ for all t, that each primitive attempt has a one-step controllable window $[r^-, r^+]$ and one-step variation σ so that*

$$[\mu_{t,m}^-, \mu_{t,m}^+] = [mr^-, mr^+], \qquad \sigma_{t,m}^2 = m\sigma^2,$$

and that $\eta_t = \eta$, $\varepsilon_t = \varepsilon$, $M_t = M$. Then

$$-\log P_h(S(\tau_h, x) = 1 \mid x) \ge T\left[\eta + \left(\min_{1\le m\le M} \frac{(d(L_x/T, [mr^-, mr^+]) - \varepsilon)_+^2}{2m\sigma^2} - \log M\right)_+\right],$$

and the certified mismatch term vanishes if and only if

$$\frac{L_x}{T} \in \bigcup_{m=1}^{M} [mr^- - \varepsilon,\ mr^+ + \varepsilon].$$

Avoiding both coarse-grained (L_x/T above the union) and fine-grained (L_x/T below the union) failure regimes therefore requires the uniform sub-goal size to be reachable by some cumulative progress scale within the finite attempt budget.

Proof of Corollary 6. Substituting the uniform and additive assumptions into $\rho_t^{(M)}$ from Remark 5 gives

$$\rho_t^{(M)} = \min_{1\le m\le M} \frac{(d(L_x/T, [mr^-, mr^+]) - \varepsilon)_+^2}{2m\sigma^2},$$

which is independent of t. Plugging into the bound in Remark 5 and using $\eta_t = \eta$, $M_t = M$ for all t yields the stated lower bound on $-\log P_h$.

The mismatch term $\rho_t^{(M)}$ vanishes precisely when $d(L_x/T, [mr^-, mr^+]) \le \varepsilon$ for some $m \in \{1, \dots, M\}$, which is equivalent to $L_x/T \in [mr^- - \varepsilon, mr^+ + \varepsilon]$ for that m. The union over m characterizes the set of sub-goal sizes for which no mismatch penalty is paid; outside this union, L_x/T either exceeds every $mr^+ + \varepsilon$ (coarse-grained failure) or falls below every $mr^- - \varepsilon$ except at small m, where the small reachable scales cannot match L_x/T either (fine-grained failure). □

D.3 Guidance as Action-Space Filtering

This appendix gives the formal statement and proof for the guidance–evidence alignment principle of Section 4.2. The result casts guidance as a reweighting of the base stage-level trajectory distribution and shows that its effect on stage recoverability is governed by a single one-dimensional quantity: the log-ratio of average retention weights on recoverable versus non-recoverable trajectories. Appendix D.3.1 sets up the filtering model, defining the guidance-filtered distribution Q_{m_t} and the conditional retention weights $\bar{W}_{t,R}, \bar{W}_{t,R^c}$ that the rest of the analysis is built on. Appendix D.3.2 then uses these objects to prove the central identity, expressing the filtered recoverability probability as a sigmoid in the base log-odds plus the retention gap G_{t,m_t}. Building on this identity, Appendix D.3.3 states the alignment theorem characterizing when guidance helps, hurts, or leaves stage recoverability unchanged, together with the process-level integration that lifts per-stage retention gaps into a final-success bound. Appendix D.3.4 closes by interpreting pseudo-compliant hallucination as a concrete instance of a negative retention gap.

D.3.1 STAGE-LEVEL FILTERING MODEL

Fix a task x, a harness stage t, and a completed prefix K_{t-1} satisfying $B_{<t}$. At this stage, the harness specifies a sub-goal g_t. Let $Q_0(\tau_t \mid K_{t-1}, g_t)$ be the base stage-level trajectory distribution.

Guidance is represented by a nonnegative measurable retention weight $W_{t,m_t}(K_{t-1}, g_t, \tau_t) \geq 0$, where m_t denotes the amount of guidance imposed at stage t. The guidance-filtered distribution is

$$Q_{m_t}(\tau_t \mid K_{t-1}, g_t) = \frac{Q_0(\tau_t \mid K_{t-1}, g_t)\, W_{t,m_t}(K_{t-1}, g_t, \tau_t)}{Z_{t,m_t}(K_{t-1}, g_t)},$$
$$Z_{t,m_t}(K_{t-1}, g_t) := \mathbb{E}_{Q_0}[W_{t,m_t}].$$

and we assume throughout that $0 < Z_{t,m_t} < \infty$.

This formulation is intentionally general: it captures inference-time guidance mechanisms that retain, downweight, or upweight candidate stage trajectories. Hard action-space pruning is the special case where W_{t,m_t} is an indicator over remaining trajectories.

Let

$$\mathcal{R}_t(K_{t-1}) := \{\tau_t : K_t = (K_{t-1}, g_t, \tau_t) \text{ remains recoverable}\}$$

denote the recoverable cross-section of the stage-level trajectory space, and assume the base recoverability probability is non-degenerate, $0 < Q_0(\mathcal{R}_t \mid K_{t-1}, g_t) < 1$. Define the conditional average retention weights on the recoverable set and its complement,

$$\bar{W}_{t,R}(K_{t-1}) := \mathbb{E}_{Q_0(\cdot \mid \mathcal{R}_t, K_{t-1}, g_t)}[W_{t,m_t}], \qquad \bar{W}_{t,R^c}(K_{t-1}) := \mathbb{E}_{Q_0(\cdot \mid \mathcal{R}_t^c, K_{t-1}, g_t)}[W_{t,m_t}],$$

and assume $0 < \bar{W}_{t,R}, \bar{W}_{t,R^c} < \infty$. Boundary cases where one of these quantities is zero are handled by the same identity in the extended-real sense, with $\sigma(+\infty) = 1$ and $\sigma(-\infty) = 0$.

D.3.2 ACTION-SPACE FILTERING IDENTITY

Lemma 7 (Action-space filtering identity)**.** *Under the conditions of the previous subsection, define the base recoverability log-odds*

$$\omega^0_{t,R}(K_{t-1}) := \log \frac{Q_0(\mathcal{R}_t(K_{t-1}) \mid K_{t-1}, g_t)}{Q_0(\mathcal{R}_t(K_{t-1})^c \mid K_{t-1}, g_t)}$$

and the guidance retention gap

$$G_{t,m_t}(K_{t-1}) := \log \bar{W}_{t,R}(K_{t-1}) - \log \bar{W}_{t,R^c}(K_{t-1}).$$

Then

$$Q_{m_t}(\mathcal{R}_t(K_{t-1}) \mid K_{t-1}, g_t) = \sigma\big(\omega^0_{t,R}(K_{t-1}) + G_{t,m_t}(K_{t-1})\big),$$

where $\sigma(u) = 1/(1 + e^{-u})$.

Proof. Suppress the dependence on (K_{t-1}, g_t) and write $\mathcal{R}_t = \mathcal{R}_t(K_{t-1})$. By definition of the filtered distribution,

$$\frac{Q_{m_t}(\mathcal{R}_t)}{Q_{m_t}(\mathcal{R}_t^c)} = \frac{Q_0(\mathcal{R}_t)\, \mathbb{E}_{Q_0(\cdot \mid \mathcal{R}_t)}[W_{t,m_t}]}{Q_0(\mathcal{R}_t^c)\, \mathbb{E}_{Q_0(\cdot \mid \mathcal{R}_t^c)}[W_{t,m_t}]},$$

since the two normalizers cancel. Taking logarithms gives

$$\log \frac{Q_{m_t}(\mathcal{R}_t)}{Q_{m_t}(\mathcal{R}_t^c)} = \omega^0_{t,R} + G_{t,m_t},$$

and converting log-odds to probability via σ yields the result. □

D.3.3 GUIDANCE–ACTION-SPACE ALIGNMENT

Theorem 8 (Guidance–action-space alignment)**.** *Under the conditions of Lemma 7, guidance improves stage recoverability relative to the unguided base distribution at the same prefix if and only if* $G_{t,m_t}(K_{t-1}) > 0$*, and harms it if and only if* $G_{t,m_t}(K_{t-1}) < 0$*. More quantitatively, if* $G_{t,m_t}(K_{t-1}) \geq \gamma$ *for some* $\gamma \geq 0$*, then*

$$Q_{m_t}(\mathcal{R}_t(K_{t-1}) \mid K_{t-1}, g_t) \geq \sigma\big(\omega^0_{t,R}(K_{t-1}) + \gamma\big),$$

and if $G_{t,m_t}(K_{t-1}) \leq -\gamma$*, then*

$$Q_{m_t}(\mathcal{R}_t(K_{t-1}) \mid K_{t-1}, g_t) \leq \sigma\big(\omega^0_{t,R}(K_{t-1}) - \gamma\big).$$

Proof. By Lemma 7, $Q_{m_t}(\mathcal{R}_t) = \sigma(\omega^0_{t,R} + G_{t,m_t})$, while the unguided probability at the same prefix is $\sigma(\omega^0_{t,R})$. Both qualitative claims and quantitative bounds follow from the strict monotonicity of σ. □

Remark 9 (Process-level integration). The chain-rule identity established in Appendix D.1 integrates Theorem 8 across stages: under goal consistency,

$$-\log P_h(\mathsf{Succ}_x(\tau_h) \mid x) = \sum_{t=1}^{T_h(x)} -\log \bar{p}_t(h; x),$$
$$\bar{p}_t(h; x) = \mathbb{E}_h\big[Q_{m_t}(\mathcal{R}_t(K_{t-1}) \mid K_{t-1}, g_t) \mid B_{<t}, x\big].$$

A negative retention gap $G_{t,m_t}(K_{t-1}) \leq -\gamma_t(K_{t-1})$ on $B_{<t}$ then yields $\bar{p}_t(h; x) \leq \mathbb{E}_h[\sigma(\omega^0_{t,R}(K_{t-1}) - \gamma_t(K_{t-1})) \mid B_{<t}, x]$, so per-stage misalignment accumulates into the final-success bound.

D.3.4 Hallucination as a Special Case

The action-space filtering view also covers pseudo-compliant hallucination in content-producing tasks, where a stage trajectory τ_t may correspond to a generated answer, reasoning trace, tool-use sequence, or explanation. Let $\mathcal{M}_t(K_{t-1}) \subseteq \mathcal{R}_t(K_{t-1})^c$ denote a hallucination basin: trajectories that appear compliant with the imposed guidance but introduce unsupported content and make the resulting prefix non-recoverable. If the guidance filter assigns high retention weight to this basin, it preserves trajectories that look locally compliant yet lie outside the recoverable region—precisely a negative retention gap, $G_{t,m_t}(K_{t-1}) < 0$. Pseudo-compliant hallucination is therefore one manifestation of the general failure mode characterized by Lemma 7 and Theorem 8.

D.4 Proofs and Details for Partial Harnessing

This appendix gives the formal statement and proof for the partial-harnessing principle of Section 4.3. Without additional structure, the marginal effect of adding one scaffolded stage, $F(m+1) - F(m)$, mixes the new scaffold cost, the residual tail-risk reduction, and any shift in earlier prefix distributions induced by changing the harness; the analysis isolates the first two by working on a homogeneous progress slice. Appendix D.4.1 introduces the slice and its factorization assumption, and proves a lemma that decomposes $F(m)$ into a per-stage scaffold cost plus a residual tail cost, from which the marginal identity $F(m+1) - F(m) = c_s - \Delta(m; M)$ follows. Building on this lemma, Appendix D.4.2 adds discrete convexity of the tail to prove Theorem 3, characterizing the unimodal success curve and the smallest scaffold count that maximizes reliability or attains a target α. Appendix D.4.3 then closes the loop with the preceding two principles, expressing the scalar c_s in terms of the granularity penalty and the guidance retention gap, and Appendix D.4.4 delineates the regimes in which the slice assumptions break and the marginal rule no longer applies.

D.4.1 Homogeneous Progress Slice

Fix a task x with total latent progress demand L_x and scaffold step size $s > 0$, and let $N := \lfloor L_x/s \rfloor$, $\mathcal{J} := \{0, 1, \ldots, N\}$. For each $m \in \mathcal{J}$, the partial harness h_m executes m scaffolded stages of length s and hands the residual length $L_x - ms$ to the autonomous continuation policy. Let $A^{(m)}_t$ be the event that the t-th scaffolded stage under h_m preserves recoverability, and let $A^{(m)}_{\text{tail}}$ be the event that the autonomous tail succeeds.

Assumption 10 (Positive homogeneous slice factorization)**.** There exist scalars $q_{\text{scaf}}(s; M) \in (0, 1]$ and $q_{\text{tail}}(L_x - ms; M) \in (0, 1]$, with $q_{\text{tail}}(0; M) = 1$, such that for every $m \in \mathcal{J}$: **(i) goal consistency:** $\text{Succ}_x(\tau_h) \equiv \bigcap_{t=1}^m A^{(m)}_t \cap A^{(m)}_{\text{tail}}$; **(ii) homogeneous scaffold reliability:** $\Pr_{h_m}(A^{(m)}_t \mid A^{(m)}_{<t}, x) = q_{\text{scaf}}(s; M)$ for $t = 1, \ldots, m$; **(iii) residual-tail reduction:** $\Pr_{h_m}(A^{(m)}_{\text{tail}} \mid A^{(m)}_1, \ldots, A^{(m)}_m, x) = q_{\text{tail}}(L_x - ms; M)$.

The assumption compresses prefix-level behavior into two scalar quantities: scaffolded stages have the same effective success probability along the slice, and the tail depends only on residual length,

not on the realized prefix. Define

$$c_s := -\log q_{\mathrm{scaf}}(s; M), \qquad \kappa_{\mathrm{tail}}(d; M) := -\log q_{\mathrm{tail}}(d; M),$$

both finite on the slice grid with $\kappa_{\mathrm{tail}}(0; M) = 0$.

Lemma 11 (Slice factorization and marginal identity)**.** *Under Assumption 10, for every $m \in \mathcal{J}$,*

$$F(m) := -\log \Pr_{h_m}(\mathrm{Succ}_x(\tau_h) \mid x) = mc_s + \kappa_{\mathrm{tail}}(L_x - ms; M),$$

and for $m, m+1 \in \mathcal{J}$,

$$F(m+1) - F(m) = c_s - \Delta(m; M),$$
$$\Delta(m; M) := \kappa_{\mathrm{tail}}(L_x - ms; M) - \kappa_{\mathrm{tail}}(L_x - (m+1)s; M).$$

Adding the $(m+1)$-st scaffolded stage strictly improves reliability if and only if $\Delta(m; M) > c_s$.

Proof. By (i), $\Pr_{h_m}(\mathsf{Succ} \mid x) = \prod_{t=1}^{m} \Pr_{h_m}(A_t^{(m)} \mid A_{<t}^{(m)}, x) \cdot \Pr_{h_m}(A_{\mathrm{tail}}^{(m)} \mid A_1^{(m)}, \ldots, A_m^{(m)}, x)$ via chain rule. Substituting (ii) and (iii) gives $\Pr_{h_m}(\mathsf{Succ} \mid x) = q_{\mathrm{scaf}}(s; M)^m \, q_{\mathrm{tail}}(L_x - ms; M)$, and taking negative logarithms gives $F(m) = mc_s + \kappa_{\mathrm{tail}}(L_x - ms; M)$. The marginal identity follows by direct subtraction, and the $\Delta(m; M) > c_s$ characterization from $F(m+1) < F(m) \iff c_s < \Delta(m; M)$. □

D.4.2 Proof of Theorem 3

Assumption 12 (Convex tail risk on the slice)**.** On the grid $\{L_x - ms : m \in \mathcal{J}J\}$, the function $d \mapsto \kappa_{\mathrm{tail}}(d; M)$ is finite, nondecreasing, and convex, with $\kappa_{\mathrm{tail}}(0; M) = 0$.

Proof. By Lemma 11, $F(m+1) - F(m) = c_s - \Delta(m; M)$.

Discrete convexity. Let $d_m := L_x - ms$. By Assumption 12, $d \mapsto \kappa_{\mathrm{tail}}(d; M)$ is convex on the slice grid, so the discrete increment $d \mapsto \kappa_{\mathrm{tail}}(d; M) - \kappa_{\mathrm{tail}}(d - s; M)$ is nondecreasing in d. Since d_m decreases as m increases, $\Delta(m; M) = \kappa_{\mathrm{tail}}(d_m; M) - \kappa_{\mathrm{tail}}(d_m - s; M)$ is nonincreasing in m, hence $c_s - \Delta(m; M)$ is nondecreasing. The forward difference of F is nondecreasing, so F is discrete-convex; equivalently $\exp(-F(m))$ is log-concave and unimodal on $\mathcal{J}$.

Smallest reliability maximizer. For a discrete-convex sequence, the minimizer set is an interval whose smallest element is the first index at which the forward difference becomes nonnegative. By Lemma 11, $F(m+1) - F(m) \geq 0 \iff \Delta(m; M) \leq c_s$, giving

$$m_{\mathrm{peak}} = \min\{m \in \mathcal{J} : m + 1 \in \mathcal{J},\ \Delta(m; M) \leq c_s\},$$

with $m_{\mathrm{peak}} = \max \mathcal{J}$ if the set is empty.

Minimum scaffold for target reliability. For $\alpha \in (0, 1)$, $P_m(x) \geq \alpha \iff F(m) \leq -\log \alpha$. The feasible set $\mathcal{J}_\alpha = \{m \in \mathcal{J} : F(m) \leq -\log \alpha\}$ is nonempty iff the target is achievable, in which case $m_\alpha = \min \mathcal{J}_\alpha$. □

D.4.3 Connection to Local Alignment Results

The slice scalar $c_s = -\log \Pr(A_s)$, with A_s the event that one scaffolded stage of length s preserves recoverability, exposes how the local alignment results of Sections 4.1 and 4.2 drive the global trade-off. The granularity bound certifies $\Pr(A_s) \leq \exp(-\gamma_s)$ when the scaffolded sub-goal lies outside the agent's reachable scales, hence $c_s \geq \gamma_s$. The guidance identity gives $\Pr(A_s \mid K_{t-1}) = \sigma(\omega_t^0(K_{t-1}) + G_t(K_{t-1}))$, so a negative retention gap raises c_s while a positive gap lowers it. The slice criterion $\Delta(m; M) > c_s$ therefore summarizes the two local effects as a single global stopping condition: alignment makes additional scaffolded stages cheaper, misalignment makes them more expensive.

Remark 13 (Capability frontier in the zero-cost limit). In the idealized regime $c_s = 0$, the minimum-structure rule simplifies. Define $\mathcal{D}_\alpha := \{d \in \{L_x - ms : m \in \mathcal{J}\} : \kappa_{\mathrm{tail}}(d; M) \leq -\log \alpha\}$ and $d_\alpha := \max \mathcal{D}_\alpha$ when nonempty. Then $m_\alpha = \min\{m \in \mathcal{J} : L_x - ms \leq d_\alpha\} = \lceil (L_x - d_\alpha)/s \rceil_+$, recovering the capability-frontier reading: scaffold only until the residual lies within the agent's autonomous reliability range. For $c_s > 0$ the exact grid rule from Theorem 3 should be used.

D.4.4 When the Slice Assumptions Fail

The slice rule rests on two restrictions and can fail when they do. *Non-nested or behavior-shifting workflows:* if h_{m+1} is not a nested truncation of h_m, or if added later instructions alter the agent's earlier behavior (through prompt length, attention, or reinterpretation of the goal), then the first m stages under h_m and h_{m+1} have different distributions, violating the scalar cost model $F(m) = mc_s + \kappa_{\text{tail}}$. *Prefix-dependent tail reliability:* if tail success depends on the realized prefix rather than only on residual length, $F(m+1) - F(m)$ contains additional terms beyond c_s and $\Delta(m; M)$, and the simple rule $\Delta(m; M) > c_s$ no longer applies. *Non-convex tail risk:* reliability cliffs, irreversible overshoot, discrete action scales, or prefix-dependent repair can break the convexity of κ_{tail}; $\Delta(m; M)$ may then fail to be monotone in m and $P_m(x)$ may be multi-modal. In any of these regimes, the safe procedure is to estimate or compare $F(m)$ directly across candidate scaffold counts.

E Prompt Details

We list the two harness templates used in our experiments below. They share a common skeleton but isolate different variables: the first holds `guidance` fixed and varies `workflow` step depth (granularity experiments), while the second holds `workflow` fixed and varies the `guidance` string (guidance experiments). Placeholders in curly braces are filled in per task at evaluation time.

Harness Template on Granularity Experiments

Task: `{task}`

Guidance:

- Follow the provided workflow.
- Inspect the evaluator and target files before choosing an implementation strategy.
- Reproduce the failure or run a minimal check before editing.
- Make the smallest local change that passes the focused evaluation without touching unrelated code.

Workflow: `{workflow}`

Harness Template on Guidance Experiments

Task: Analyze the provided Plotly chart structure and produce a concise, evidence-grounded conclusion.

Guidance: `{guidance}`

Workflow:

1. Identify the chart type and the available variables.
2. State what can and cannot be inferred from the provided information.
3. Give a concise conclusion without inventing unsupported distributional details.