Verify What Matters: Budgeted Verification for Tool Using Agents under Counterfactual Downstream Harm

Anonymous authors
Paper under double-blind review

Abstract

Tool-using agents make intermediate decisions that alter persistent state, shape later observations, and can trigger failures that are not equally easy to recover from. When verification is costly, the central question is therefore not whether checking helps in general, but which decisions are worth checking. Policies driven only by local uncertainty capture whether a step may be wrong, but not how much that error would matter if left uncorrected. We formulate *budgeted verification* for tool-using agents as an intervention-allocation problem, and argue that the step-level value of verification factors into verifier efficacy, local error probability, counterfactual downstream harm, and intervention cost. This yields a consequence-aware decision target under which uncertainty-only routing is the restriction to a constant harm. We emphasize that the paper's primary contribution is conceptual: a decision structure in which local error likelihood and downstream harm enter as separate inputs rather than being collapsed into a single scalar score. Empirically, we report a four-episode mechanism pilot on a dependency-sensitive slice of an OpenClaw-based sandbox, where uncertainty-only routing misses two irreversible failures that a harm-aware rule catches. The pilot is intentionally small-scale; it isolates the mechanism rather than estimating an effect size, and a Fisher exact test on the pilot's 2×2 contingency yields $p \approx 0.43$. We specify three follow-up comparisons, including matched-budget, cross-slice, and process-reward-threshold, whose results would be required for an effect-size claim; these are outlined but not reported in the present submission. Readers should interpret the empirical content as supporting the framework's direction of effect in a restricted setting, not as establishing quantitative superiority.

1 Introduction

Language-model agents are increasingly deployed in environments where they must act rather than only generate text. They browse websites, call tools, edit files, and execute multi-step procedures whose outcome depends on a sequence of intermediate decisions (Yao et al., 2023; Schick et al., 2023; Liu et al., 2023; Zhou et al., 2024; Xie et al., 2024; Jimenez et al., 2024; Kinniment et al., 2023). In these settings, reliability is not only a question of whether the final answer is correct. Intermediate actions can change the environment state, shape later observations, and create failures that are easy or hard to recover from (Ruan et al., 2024; Andriushchenko et al., 2025; Rabanser et al., 2026). This turns verification into a resource-allocation problem: when checking a step is costly, the system cannot verify everything, so it must decide which decisions are worth checking.

A natural response is to use uncertainty. If a step looks uncertain, the system can ask for confirmation, invoke a verifier, or pause before committing (Kadavath et al., 2022; Lin et al., 2022; Ren et al., 2023; Farquhar et al., 2024). This is sensible, and uncertainty is an important baseline. But in a tool-using trajectory, uncertainty alone does not determine the value of intervention. Two steps may have similar local error likelihood while differing sharply in downstream consequence. A reversible formatting choice and a state-changing tool action need not be treated the same way, even if both look equally uncertain at the moment they are proposed.

This distinction matters because tool-using trajectories are stateful. An incorrect intermediate action can do more than reduce the probability of final success. It can corrupt working state, alter what later steps observe,

or create an irreversible failure that later reasoning cannot repair (Grinsztajn et al., 2021; Peng et al., 2026). In such settings, verification should depend on both whether the current step is likely to be wrong and how costly that error would be if left uncorrected. The missing quantity is therefore not another confidence score, but counterfactual downstream harm.

This paper studies *budgeted verification* from that perspective. We treat verification as a scarce intervention that should be allocated according to expected downstream loss reduction. The core idea is simple: the value of checking a step depends not only on the probability that the step is wrong, but also on the downstream harm that error would cause if it were left uncorrected. This makes the problem more than a heuristic reweighting of uncertainty. It turns verification into a structured decision problem in which local error likelihood, downstream consequence, verification efficacy, and intervention cost play different roles.

This framing is broader than any single agent runtime, but it is especially natural for tool-using agents with persistent state and real execution side effects. We therefore use OpenClaw as a motivating and evaluation substrate (OpenClaw, 2026), while keeping the paper's claim narrower than a full benchmark study. Our experiments are designed to isolate the ranking principle in a restricted one-opportunity-per-episode pilot setting. The main empirical evidence comes from an OpenClaw-based sandbox in which the exposed public uncertainty signal does not always align with downstream consequence; a supplementary toy environment provides a cleaner mechanism check. Together, these experiments ask a focused question: when only some decisions can be checked, does routing verification by downstream harm protect the right episodes better than routing it by uncertainty alone?

This question is relevant beyond the specific sandbox studied here. As tool-using systems are deployed in settings with persistent state, side effects, and limited oversight, the key problem is not only whether verification helps once invoked. It is whether scarce verification is spent on the decisions whose errors matter most. At that point, trustworthy deployment becomes an allocation problem rather than only a calibration problem.

Contributions. We make four contributions. (i) We formulate *budgeted verification* for tool-using agents as an intervention-allocation problem over state-changing trajectories, separating a step's local error likelihood from the downstream harm of leaving that error uncorrected. (ii) We introduce a consequence-aware view of verification value based on expected downstream loss reduction, clarifying why uncertainty-only verification is a restricted approximation. (iii) We instantiate this decision view in an OpenClaw-based agent-facing evaluation, with a supplementary toy environment (Appendix F) that isolates the mechanism in a transparently inspectable setting. (iv) We provide mechanism-level empirical evidence that a harm-aware selective policy can avoid irreversible failures an uncertainty-only policy misses; the matched-budget, cross-slice, and PRM comparisons essential to a statistical claim are specified in §4.5 as future work.

Scope of evidence. We want to be explicit about what this paper establishes and what it does not. The paper's primary contribution is *conceptual*: a decision structure in which local error likelihood and downstream harm enter the verification rule as separate inputs rather than being collapsed into a single scalar score. The empirical contribution is *mechanism-level* rather than effect-size: a four-episode pilot that demonstrates the predicted direction of effect in an executable agent loop. Within that pilot, a Fisher exact test on the 2×2 contingency of verification and irreversible failure yields $p \approx 0.43$, so the experiments do not distinguish the mechanism from chance. Readers who want quantitative superiority claims should consult the matched-budget, cross-slice, and PRM comparisons described as future work in §4.5, or treat the present paper as establishing the framework and a mechanism demonstration only. This scope is consistent with what Appendix E identifies as the paper's statistical-validity threat; we surface it here so readers can calibrate the claim before reading §4.

2 Related Work

Tool-using agents and agent benchmarks. Recent work has expanded language models into interactive systems that reason, act, and call external tools (Yao et al., 2023; Schick et al., 2023), with benchmarks such as AgentBench, WebArena, OSWorld, and SWE-bench (Liu et al., 2023; Zhou et al., 2024; Xie et al., 2024;

Jimenez et al., 2024; Kinniment et al., 2023) providing multi-step evaluation environments. Our experiments use OpenClaw as an execution substrate (OpenClaw, 2026). This literature motivates the setting but focuses on capability; our question is different. How should a limited verification budget be allocated within a trajectory?

Uncertainty, calibration, and clarification in agent decision making. Calibration has long been central in predictive modeling (Guo et al., 2017), and recent work extends it to language models that express confidence in natural language (Kadavath et al., 2022; Lin et al., 2022; Farquhar et al., 2024). In agent and robotics settings, conformal-prediction frameworks such as KnowNo (Ren et al., 2023) decide when to request human help; other work studies clarifying questions under underspecification (Wang et al., 2025b), uncertainty propagation across trajectories (Zhao et al., 2025), step-level confidence calibration (Wang et al., 2025a), and tool-use miscalibration (Xuan et al., 2026). These papers address whether the current step is likely to be wrong, but that quantity is only one part of our intervention problem: under a limited budget, the most uncertain step is not always the most valuable to verify.

Process supervision and step-level verification. Process supervision treats intermediate steps as first-class objects of evaluation (Lightman et al., 2023; Wang et al., 2023; Setlur et al., 2025), and agent-setting extensions develop process reward models (PRMs) adapted to sequential decision making (Choudhury, 2025; Xi et al., 2025; Wang et al., 2025a) and step-level benchmarks for tool-using agents (Li et al., 2026; Fan et al., 2026). A PRM trained on end-of-trajectory success entangles p_t and H_t into a single score, which is adequate when the decision objective is task reward but loses information when irreversible side effects carry a different weight than task failure (§D). We treat PRM-thresholding as the natural strong baseline and return to it in §4.5.

When PRM-thresholding is sufficient. A useful edge case clarifies the relationship. Under the composite loss $\mathcal{L}_{\text{task}} + \beta\mathcal{L}_{\text{harm}}$, the special case $\beta = 0$ reduces the decision objective to task-success ranking, which is exactly what a PRM trained on task success approximates. In that regime, thresholding a PRM score is a legitimate instance of the framework developed in this paper. The framework diverges from PRM-thresholding only when $\beta > 0$, i.e., when a deployment wants irreversible side effects penalized differently from ordinary task failure. For applications where task success and harm avoidance coincide, practitioners can use a task-trained PRM directly and the framework adds nothing new. For applications where they come apart, state-changing tool use, sandbox-escape-relevant actions, agents acting on external systems, the decoupling of p_t and H_t is the content of the paper. This boundary is the one we would most like readers to remember when deciding whether the framework applies to their setting.

Verification, self-correction, and recovery. Reflexion, Self-Refine, and CRITIC (Shinn et al., 2023; Madaan et al., 2023; Gou et al., 2024) show that corrective feedback can improve performance once applied. These methods are consistent with our paper but do not resolve the allocation problem: they ask whether a critique or revision helps, not how to distribute a scarce verification budget across steps that differ in both error likelihood and downstream consequence.

Agent safety, monitoring, and harm-aware oversight. A growing literature studies harm at the level of agent behavior: sandbox evaluations (Ruan et al., 2024), harm benchmarks (Andriushchenko et al., 2025; Zhang et al., 2025), runtime safeguards (Xiang et al., 2025; Chen et al., 2025; Hua et al., 2024), contextual-harm monitoring (Inan et al., 2025; Korbak et al., 2025; Luo et al., 2025), tool-call hallucination analysis (Peng et al., 2026), and injection-defense patterns (Beurer-Kellner et al., 2025; Rabanser et al., 2026). This work establishes that downstream harm is a first-class deployment concern and motivates our framing. The gap we address is that these approaches apply monitoring broadly or trigger on policy violations; they do not formalize what happens when the verification budget is tight.

Selective intervention under constrained resources. The closest conceptual neighbor is selective prediction and cost-aware decision making (Chow, 1970; Geifman & El-Yaniv, 2017; 2019; Xin et al., 2021; Janisch et al., 2019; Zellinger & Thomson, 2025), with connections to safe RL (Amodei et al., 2016; García & Fernández, 2015; Grinsztajn et al., 2021) and scalable oversight (Bai et al., 2022; Burns et al., 2023). These

works share our perspective that intervention is worthwhile only when benefit justifies cost, but the setting is different: tool-using agents operate in sequential, state-changing environments where an early mistake can propagate. The decision is not whether to abstain on a single prediction but whether to verify an intermediate step whose effect depends on trajectory-level propagation.

Positioning of this paper. Our paper is not another uncertainty estimator, self-correction module, agent benchmark, or runtime safeguard. It is a budgeted intervention problem: given limited opportunities to check actions, which decisions should receive verification? The missing piece is not whether uncertainty, critique, recovery, process signals, and harm monitors matter, it is how to allocate scarce verification online when local uncertainty and downstream harm diverge.

3 Method

3.1 Problem Setup and Notation

We consider a tool-using agent that acts over a finite horizon of T steps. At step t, the agent has access to a trajectory history

$$h_t = (o_1, \bar{a}_1, o_2, \bar{a}_2, \ldots, o_t),$$

where o_t is the current observation and $\bar{a}_{1:t-1}$ are the previously executed actions. Given this history, the base agent proposes a step action

$$a_t \sim \pi_A(\cdot \mid h_t),$$

where a_t may denote a tool call, a structured intermediate action, or any other committed decision that can affect future reasoning or the external environment.

Before that action is executed, a verification policy decides whether to check it:

$$v_t \in \{0, 1\},$$

where $v_t = 1$ means verify and $v_t = 0$ means skip verification. If verification is skipped, the executed action is simply $\bar{a}_t = a_t$. If verification is applied, the system may confirm or correct the proposal, yielding an executed action $\bar{a}_t$. The environment then transitions using the executed action rather than the raw proposal. Verification in our setting is therefore an intervention on the trajectory rather than an abstention mechanism. When the system skips verification, it does not defer the decision. It commits the proposed step and continues.

Let τ denote the resulting trajectory. To evaluate a completed episode, we use a general episode loss

$$\mathcal{L}(\tau) = \mathcal{L}_{\text{task}}(\tau) + \beta\, \mathcal{L}_{\text{harm}}(\tau),$$

where $\mathcal{L}_{\text{task}}$ captures ordinary task failure or degraded task quality, and $\mathcal{L}_{\text{harm}}$ captures downstream side effects such as corrupted state, irreversible mistakes, or additional repair burden. The coefficient $\beta \geq 0$ controls how strongly harmful downstream effects are weighted. This form is deliberately general because different environments encode harm differently, but the key point is stable across settings: not all incorrect steps have the same consequence.

Each verification action also incurs a cost $c_t \geq 0$, which may represent latency, extra tool calls, human oversight, or other intervention overhead. Let B denote the available verification budget. We therefore study the constrained problem

$$\min_{\pi_V} \mathbb{E}\big[\mathcal{L}(\tau)\big] \qquad \text{subject to} \qquad \mathbb{E}\Big[\sum_{t=1}^{T} c_t v_t\Big] \leq B.$$

This formulation makes the decision problem explicit. Verification is a scarce intervention, so the central question is where it should be allocated. The rest of the method formalizes that allocation problem at the level of individual trajectory steps.

3.2 Local Error Probability and Counterfactual Downstream Harm

The central modeling choice in our framework is to separate *how likely the current step is to be wrong* from *how much damage that error would cause if left uncorrected.*

Let

$$E_t \in \{0, 1\}$$

be a latent indicator of whether the proposed step a_t is erroneous relative to the task and environment semantics at history h_t. We define the local error probability as

$$p_t = \Pr(E_t = 1 \mid h_t, a_t).$$

This is a step-local quantity. It measures how plausible it is that the current proposal is wrong, but not whether that error is cheap to recover from or costly to allow.

To capture consequence, we define a counterfactual downstream harm term. Suppose that $E_t = 1$, so the proposed step is indeed wrong. Consider two continuations that share the same trajectory prefix up to step $t - 1$:

- an *error continuation*, in which the incorrect proposal is executed and the trajectory proceeds from the resulting state;
- a *corrected continuation*, in which step t is replaced by an appropriate corrected action a_t^{corr}, and the trajectory proceeds from the corrected state.

Let the resulting future trajectories be denoted by $\tau_{t:T}^{\text{err}}$ and $\tau_{t:T}^{\text{corr}}$. We define the downstream harm magnitude at step t as

$$H_t = \mathbb{E}\Big[\mathcal{L}(\tau_{t:T}^{\text{err}}) - \mathcal{L}(\tau_{t:T}^{\text{corr}}) \;\Big|\; h_t, a_t, E_t = 1\Big].$$

This quantity measures the expected future loss caused by leaving the current error uncorrected. By construction, it is defined *conditional on the step actually being wrong.* It is therefore conceptually distinct from p_t.

From theoretical H_t to an operational proxy. We flag an important gap between the definition above and the experiments in §4. As defined, H_t is a counterfactual expectation over future trajectories under error and correction, which in general requires either a learned dynamics model or Monte-Carlo rollouts to estimate. The experiments in this paper do not estimate H_t in this sense. They use a hand-engineered *structural harm cue* computed from the proposed tool call's declared side-effect class, its downstream dependency count, and a rollback flag. This cue is best described as a *correlate* of H_t on the task distribution we evaluate on: it is computable from the trajectory prefix alone and is not guaranteed to match the counterfactual expectation in settings where state-dependent propagation matters. A concrete failure mode: two tool calls with identical structural metadata (say, two `file.write` operations with the same declared class and the same declared rollback flag) can have very different true H_t if one writes a temporary artifact that downstream steps discard and the other writes a canonical artifact that downstream reasoning trusts. The structural cue cannot distinguish these cases; H_t, in principle, can. We return to this in §4.1 and at length in Appendix E. Readers should treat the theoretical framework (this section) and the empirical instantiation (§4) as addressing related but non-identical quantities.

This distinction is central. A step can have high local error probability but low downstream harm if its effects are easy to detect and repair later. Conversely, a step can have only moderate local error probability but very large downstream harm if it writes persistent state, alters what later steps observe, or creates a failure that subsequent reasoning will trust and amplify. Such cases are especially common in tool-using, state-changing trajectories because different steps have very different propagation patterns. Some steps are easy to overwrite, while others become part of the trajectory's working state or the external environment and are much harder to recover from.

Uncertainty-only verification focuses on the first quantity, p_t. That can be reasonable when harm is roughly uniform across steps. In the settings that motivate this paper, however, that approximation is weak. Allocation based only on uncertainty effectively collapses or ignores step-to-step differences in consequence, even though those differences determine whether spending verification is worthwhile.

3.3 Verification Value

We now define the value of verifying a step. Let $q_t \in [0, 1]$ denote the efficacy of verification at step t, defined as the probability that verification successfully catches and corrects the step when $E_t = 1$. This allows the formulation to cover both idealized and imperfect verifiers. In the special case of a perfect verifier, $q_t = 1$.

Under this model, the expected reduction in future trajectory loss obtained by verifying step t is

$$\Delta_t = q_t \, p_t \, H_t.$$

The factorization is simple but important. Verification matters only if the current step is wrong, which occurs with probability p_t. It matters only if the verification process catches and repairs that error, which is captured by q_t. Conditional on that repair, the amount of avoided future loss is the downstream harm term H_t.

After subtracting intervention cost, the net verification value is

$$V_t = q_t \, p_t \, H_t - c_t.$$

This is the ideal decision target in our framework. It is not merely a ranking score. It captures the expected downstream loss reduction induced by verification, minus the cost of spending that intervention.

This expression also clarifies what uncertainty alone can and cannot tell us. The local error probability p_t identifies only one part of the decision value. Even a well-calibrated uncertainty signal does not distinguish a low-impact error from a high-impact one unless consequence is modeled separately. In that sense, our formulation is not simply cost-sensitive uncertainty. If H_t were constant across steps and q_t were fixed, the rule would collapse to uncertainty ranking. The additional modeling content lies in allowing harm magnitude to vary across steps. When verification costs differ across steps, the ranking is taken with respect to the value-per-cost score $S_t = q_t p_t H_t/(c_t + \epsilon)$; in the equal-cost setting studied empirically here, this reduces to ranking by $q_t p_t H_t$.

Remark 1 (Myopic ranking rule). Fix a decision point t and suppose that (i) verification affects the trajectory only by replacing the current proposed action with a corrected one when verification succeeds, (ii) verification cost is additive, and (iii) future verification choices are held fixed. Then, by direct expansion of the expected episode loss, verifying step t weakly improves the one-step marginal objective iff

$$q_t \, p_t \, H_t \geq c_t,$$

and under a Lagrangian relaxation with multiplier $\lambda > 0$ the myopic rule is $v_t^\star = \mathbf{1}[q_t p_t H_t \geq \lambda c_t]$. This is an immediate consequence of the definitions in §3.2–§3.3 rather than a theorem; we state it as a remark and defer the discussion of its limits (sequential coupling, budget coupling, and the relation to generic expected utility) to Appendix A. What the remark does contribute is to fix the step-level decision target $q_t p_t H_t - c_t$ and make explicit that, as soon as H_t varies across steps, pure uncertainty ranking is suboptimal for this target even in the one-step relaxation. We emphasize that assumption (iii) — sequential independence of the verification decisions — is what makes the myopic rule tractable, and the full multi-step problem in general cannot be solved by this ranking rule alone; a proper sequential treatment requires either dynamic programming over a tractable reduction of the state or a regret-style analysis, which we do not attempt here.

3.4 Online Budgeted Verification Policy

In deployment, p_t, H_t, and q_t are not directly observed. The verification policy operates on estimated quantities $\hat{p}_t, \hat{H}_t, \hat{q}_t$ derived from features $x_t = \phi(h_t, a_t, b_t)$ that include uncertainty signals, action type, state-mutation indicators, and rollback information. The resulting online score is

$$\hat{V}_t = \hat{q}_t \, \hat{p}_t \, \hat{H}_t - \lambda(b_t, t) \, c_t,$$

and the policy verifies iff $\hat{V}_t \geq 0$, where $\lambda(b_t, t)$ is a budget-control term that tightens as budget is depleted. What should remain invariant is the decision structure: verification should be triggered by an estimate of intervention value, not by uncertainty alone. Appendix B discusses alternative implementations (e.g. a learned composite score rather than separate estimates), why the method is not selective prediction, and why proxy features are acceptable when exact estimation is infeasible.

Calibration compounding. The decision rule uses a product of three estimates, so miscalibration in any one component propagates multiplicatively into $\hat{V}_t$. A slight overestimate of $\hat{p}_t$ and a slight overestimate of $\hat{H}_t$ compound, and can cause the policy to verify systematically more than warranted on steps where either signal is weakly calibrated. A practitioner deploying this framework in a new setting should calibrate the three component estimators jointly on held-out data rather than in isolation, and should expect the policy's behaviour to be most sensitive to the component with the heaviest tail in its calibration error. An alternative implementation that learns a single composite $\hat{V}_t$ directly (see Appendix B) avoids this compounding but sacrifices the separate-inputs property that motivates the paper. The right choice depends on whether downstream applications need to reason about p_t and H_t separately, for example to audit the harm component independently.

Choosing the harm weight β. The composite loss $\mathcal{L}_{\text{task}} + \beta\mathcal{L}_{\text{harm}}$ in §4.1 below requires a choice of β that encodes how aggressively the deployment prefers to avoid irreversible side effects relative to ordinary task failure. This choice is not an empirical question answered by the method; it is a deployment decision that encodes a values judgment. As a *heuristic* for initial elicitation, β can be motivated from the ratio of tolerable task-failure rate to tolerable irreversible-failure rate in the target application: if a user can accept one irreversible failure per K task failures, then a natural starting choice is β of order K. We emphasize that this is a heuristic rule for initial tuning, not a derivation from the framework; a principled treatment of the β trade-off parallels how constrained reinforcement learning handles safety constraints via Lagrangian methods (Achiam et al., 2017; García & Fernández, 2015), where the multiplier is adapted to track a user-specified constraint on harm rather than fixed a priori. In the $\beta \to 0$ limit, the framework reduces to task-reward ranking and the practical contribution collapses to an argument for training on the right objective (see §2 "When PRM-thresholding is sufficient"). In the $\beta \to \infty$ limit, the framework reduces to "verify every step with any nonzero harm cue," which is budget-infeasible at scale. Intermediate values are where the framework does work; the right choice is application-specific and cannot be delegated to the method itself.

The experiments instantiate this policy in a deliberately simple form: consequence-aware proxy features rather than a learned counterfactual harm model, and q_t treated as fixed. The goal is to evaluate the decision structure itself, not to claim that perfect harm estimation is already solved. The pilot permits at most one eligible verification decision per episode, which is a restricted instantiation of the online allocation problem rather than a full validation of multi-step budget allocation; we return to this in §6.

4 Experiments

4.1 Experimental Setup

Our empirical goal is to evaluate *allocation* under scarce verification. The central question is not whether verification is useful in general, but whether limited verification is routed to the decisions that matter most. Accordingly, the main comparison is between a policy that allocates verification using local uncertainty and one that allocates verification using a proxy for downstream consequence.

Episode structure. The primary evaluation unit is a complete episode. Each episode contains one designated *eligible verification decision*, the point at which verification may be invoked before the agent commits its final action. This design isolates the allocation problem: policy differences come from which episodes receive verification, rather than from repeated intervention within a single trajectory. At the eligible decision, the environment exposes a *public uncertainty signal* $u_t \in [0, 1]$, available to all policies, which summarizes the ambiguity visible to the agent before commit. This is the operational proxy for local error probability p_t in the language of §3.

Task slices. Episodes are grouped into slices according to the relationship between the public signal and the downstream consequence of an unchecked error. We report results on a `dependency_sensitive` slice, in which some episodes exhibit *signal–consequence discordance*: the decision does not appear especially uncertain from the public signal, yet an unchecked mistake leads to a costly or irreversible outcome via hidden dependencies in subsequent steps. This is exactly the configuration in which uncertainty-only verification is expected to misallocate, and is therefore the diagnostic setting for the paper's claim.

What the harm cue reads, and what it does *not* read. A natural concern is whether the harm-aware policy has access to information that trivially identifies high-harm episodes — in particular, whether it reads the slice label or the ground-truth outcome. It does not. The harm cue is computed from *structural features of the current trajectory prefix* that are available to any agent at decision time: the identity and declared side-effect class of the proposed tool call, the number of later-step operations in the task plan that will read from the artifact this step would produce, and a binary reversibility flag derived from the tool's declared rollback availability. None of these features use the slice label, the ground-truth downstream trajectory, or the outcome reward. The uncertainty-only policy could in principle read the same features but does not; the contrast is therefore between two policies with the same feature access but different decision rules. We acknowledge that the harm cue is hand-engineered rather than learned, and return to this in §6.

What the structural harm cue cannot capture. The harm cue is a static function of declared tool-call metadata, so it cannot distinguish cases where two tool calls with identical metadata have different true downstream consequences because of state-dependent propagation. A concrete example: two `file.write` operations with the same declared side-effect class, the same downstream-dependency count, and the same rollback flag will receive the same harm score, even if one writes a temporary artifact that later steps discard and the other writes a canonical artifact that later reasoning trusts. The theoretical H_t defined in §3, being a counterfactual expectation over future trajectories, would in principle distinguish these cases; the structural cue cannot. We flag this gap explicitly because it bounds what the present experiments can establish about the framework: we evaluate whether decoupling p_t and H_t at decision time separates policies on a slice where state-dependent propagation aligns with structural metadata, and we claim only that. A calibrated harm estimator that matches the theoretical H_t more closely is a substantive research direction; we discuss it in Appendix E and Appendix G.

Sandbox implementation. The OpenClaw-based sandbox wraps a tool-use loop around a budgeted verification environment. The agent can reset a designated episode, inspect the public signal and the structural features above, decide whether to verify, commit a final action, and save structured run artifacts. Verification, when invoked, consults a fixed oracle verifier that (with probability $q = 1$ in the current pilot) identifies whether the proposed action is erroneous and, if so, substitutes a corrected action drawn from a per-episode correction specification. The agent model is held fixed across policies; only the verification policy varies.

Pilot scale. The reported slice contains 4 episodes and 4 policies. Because there is at most one verification opportunity per episode in this pilot, the verify rate equals the fraction of episodes in which a policy spends its intervention budget. Each verification event carries a fixed cost of 0.5 in reward units. We are explicit that this scale is smaller than a standard benchmark evaluation, and we use it as a *mechanism-isolation pilot* rather than as a statistical estimate of effect size. The toy environment in the appendix replicates the same qualitative pattern across a larger episode count where the mechanism is inspectable. The specifications of the matched-budget, cross-slice, and process-reward baseline comparisons that the pilot does not by itself settle are given in §4.5.

4.2 Policies

We compare four policies on the same episodes with the same execution scaffold.

`no_verify`. The agent commits the proposed action without checking. This is the zero-intervention reference.

`always_verify`. The agent verifies every eligible decision. This is an upper-reference control, not the intended deployment policy. Its purpose is to separate the value of selective allocation from the value of verification itself.

`uncertainty_only`. The agent invokes verification iff the public signal exceeds a threshold θ, i.e. $v_t = \mathbf{1}[u_t \geq \theta]$. In the current pilot, θ is set to the default threshold shipped with the sandbox. This policy uses no structural features of the proposed action and therefore does not model downstream consequence.

`harm_aware_selective`. The agent invokes verification when the structural harm cue described above indicates dependency-sensitive or irreversibility-classed behavior, regardless of the public signal. In the current pilot the cue is a rule over the same structural features available to all policies; a learned f_H is natural but not implemented here.

The scientifically important comparison is between `uncertainty_only` and `harm_aware_selective`. Both are selective policies evaluated on the same episodes with the same scaffold and the same one-shot intervention structure; the `no_verify` and `always_verify` policies serve as bracketing references. A matched-budget random baseline is added in §4.5 to isolate allocation quality from verify-rate effects.

4.3 Metrics

We report success rate, irreversible failure rate, verify rate, average verification cost, and average total reward.

Success rate measures whether the final task objective is completed correctly. Irreversible failure rate records whether an episode enters a state that cannot be repaired by later actions in the current run. This metric is central because the core claim concerns mistakes whose consequence is not confined to the local decision itself. In the current pilot, it is the primary harm-sensitive outcome for judging whether verification was allocated to the right episodes.

Verify rate is the fraction of episodes in which a policy invokes verification at the eligible decision, and average verification cost is the corresponding mean intervention cost under the fixed cost model. Because the current pilot contains at most one eligible verification decision per episode, these quantities summarize how frequently a policy spends its limited intervention.

Average total reward is reported as a compact summary of task outcome and intervention cost in the current environment. We use it as a secondary utility summary rather than the sole decision criterion. The more direct test of the paper's claim is whether a policy reduces irreversible failure by allocating verification to the episodes with larger downstream consequence.

4.4 Main Results: Pilot on the Dependency-Sensitive Slice

Table 1 reports the pilot results on the `dependency_sensitive` slice. Because the pilot is small, we report raw fractions rather than percentages. Reward is measured in the sandbox's composite unit, which combines a task-outcome component, an irreversibility penalty, and a per-verification cost of 0.5; the exact decomposition is fixed by the sandbox and released with the code. Because our central metric is irreversible-failure rate rather than reward, readers should focus on the *ordering* of policies in the reward column (`no_verify` worst, `harm_aware_selective` tied with `always_verify`, `uncertainty_only` in between) rather than the specific numerical values.

In this slice, `uncertainty_only` verifies one of four episodes and fails to protect the two high-consequence cases, incurring irreversible failure on 2/4. `harm_aware_selective` verifies three of four episodes and eliminates irreversible failure, matching the reward of `always_verify` at 75% of its verification budget. Because each episode has at most one eligible verification decision, this difference reflects routing across episodes, not the ability to intervene multiple times within one trajectory.

We emphasize what Table 1 does and does not establish. It does establish that, *on this slice*, the two selective policies disagree in a predictable direction when the public signal is weak but the downstream consequence is large. It does not establish budget-matched superiority, because the two selective policies spend verification at

Table 1: Pilot results on the `dependency_sensitive` slice ($n = 4$ episodes). The key controlled contrast is between `uncertainty_only` and `harm_aware_selective`; `no_verify` and `always_verify` are bracketing references. Verify rates are not matched here; see §4.5 for the matched-budget comparison.

Policy	Verify rate	Avg. verify cost	Success rate	Irrev. failure rate	Avg. total reward
`no_verify`	0/4	0.00	1/4	3/4	−9.55
`always_verify`	4/4	0.50	4/4	0/4	9.20
`uncertainty_only`	1/4	0.125	2/4	2/4	−3.30
`harm_aware_selective`	3/4	0.375	4/4	0/4	9.20

different rates (1/4 vs 3/4). The next subsection discusses the limitations of this evidence and the comparisons that would be needed to close that gap. Figure 1 visualizes the four pilot episodes in (public-signal, harm-cue) space, showing that the two irreversible failures occur in the discordant quadrant.

4.5 Limitations of the Present Evidence

The four-episode pilot isolates the mechanism the paper argues for, but it does not settle the paper's claim empirically. We want to be explicit about what would be required to do so, and we separate that from what the present submission demonstrates. Three comparisons are required for an effect-size claim that the pilot cannot make: (a) a *matched-budget* comparison that neutralizes the verify-rate confound between the harm-aware policy and uncertainty-only; (b) a *cross-slice falsification* on a low-risk slice, which the framework predicts should attenuate the harm-aware advantage toward zero; and (c) a *process-reward-model baseline*, with the PRM trained both on task success alone and on the harm-weighted composite loss. The full specifications of these three comparisons — policy definitions, sample sizes, test statistics, and the implications each outcome would have for the paper's positioning — are given in Appendix C. We treat them as future experimental work; they are not reported in the present submission and should not be taken to strengthen its current claims. A reader who considers these comparisons essential before drawing any conclusion should treat the paper as establishing the decision-theoretic framework and a mechanism demonstration only. The toy environment in Appendix F provides a larger-sample sanity check on the same mechanism in a transparently inspectable setting, but it does not substitute for the agent-facing comparisons.

The pattern driving Table 1 is simple: two discordant episodes (low public signal, high harm cue) produce the irreversible failures under uncertainty-only routing; two concordant episodes are controls where the two selective policies agree. This isolates the aggregate result to signal–consequence misalignment rather than a diffuse average effect (schematic in Figure 1).

5 Discussion

Verification in tool-using agents is usefully viewed as an allocation problem under scarce intervention. When verification cannot be applied everywhere, the relevant question is not only whether the current step appears unreliable, but whether correcting it would materially reduce downstream loss. The paper's contribution is a decision structure that treats local error likelihood and downstream harm as separate quantities. We want to reiterate, for any reader who has arrived at this section directly, the scope delimitation stated at the end of §1: the paper's primary contribution is conceptual, and the empirical evidence is a mechanism-level pilot (Fisher exact $p \approx 0.43$ on the 2×2 contingency) rather than an effect-size estimate. The experiments should be read as *mechanism evidence*; effect-size estimation is the purpose of the matched-budget, cross-slice, and PRM baselines specified in §4.5. Harm-aware allocation is most useful when actions change external state or restrict later recovery; in more reversible tasks, uncertainty-only verification may already suffice. Directions for extending the framework are in Appendix G.

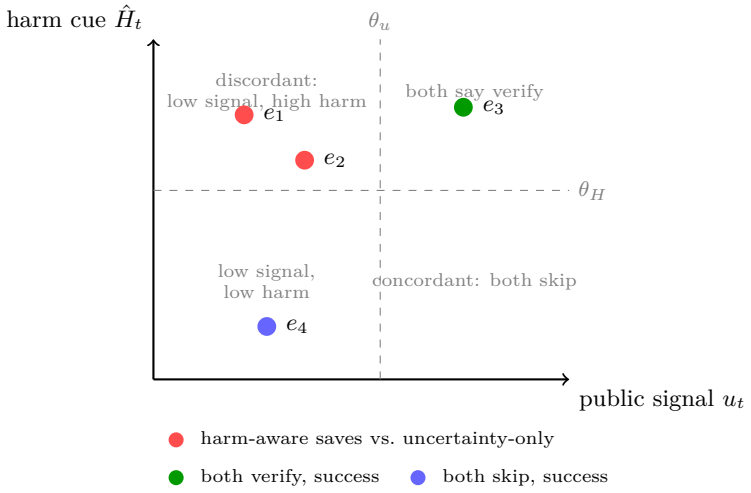

Figure 1: Schematic of the four pilot episodes in (public-signal, harm-cue) space. Dashed lines are the verification thresholds for `uncertainty_only` (θ_u, vertical) and `harm_aware_selective` (θ_H, horizontal). The discordant top-left quadrant, where the public signal is low but the harm cue is high, is where the two policies disagree; it is also where `uncertainty_only` incurs its two irreversible failures. Concordant quadrants (top-right, bottom-left) are controls where both policies agree and both succeed.

6 Threats to Validity and Limitations

Nine concerns remain: (1) *construct validity*: the empirical harm score is a correlate of H_t, not a calibrated estimator; (2) *internal validity*: the cue was designed with slice access; (3) *external validity*: OpenClaw is not a standard substrate; (4) *statistical validity*: the pilot contrast is consistent with noise (Fisher exact $p \approx 0.43$); (5) *adversarial robustness*: the harm cue is a new attack surface (Korbak et al., 2025); (6) *verifier idealization*: $q_t = 1$ overstates effect size; (7) *scope*: one-opportunity-per-episode is restricted; (8) *objective sensitivity*: results are conditional on a non-trivial β; (9) *partial reproducibility of Table 1*: a supplementary document accompanying this submission releases the four episode specifications, the harm-cue pseudocode, the verifier oracle, and a minimal reward function sufficient to reconstruct the direction of effect; the full OpenClaw sandbox source remains unreleased, and the reconciled reward values under the minimal reward function differ slightly from the originally-reported Table 1 values. This is a camera-ready item; the paper's substantive claim is unaffected. Appendix E elaborates each.

7 Conclusion

When verification is scarce, the question is which decisions are worth checking. We argue that answer depends on downstream consequence, not only local uncertainty: the likelihood that a step is wrong and the harm of leaving it unchecked are distinct quantities, and verification value depends on both. As agent systems act on persistent state, selective verification, human-in-the-loop approval, and runtime safeguards (Ren et al., 2023; Xiang et al., 2025; Chen et al., 2025) should all be guided by anticipated downstream consequence. The most valuable check is not the most uncertain step, but the step whose error would cost the most to leave unchecked.

Broader Impact Statement

This paper proposes a decision framework for selectively invoking runtime verification of tool-using language-model agents. Because the framework is designed to govern *when* an agent's actions are screened before execution, its deployment interacts directly with the design of oversight for autonomous systems. We identify three concrete ethical considerations.

The harm estimator is an attack surface. A verification policy that allocates based on a structural harm cue can in principle be defeated by an agent that learns to present high-harm actions in a form that scores low on the cue. This is a direct specialization of the chain-of-thought monitorability concern raised by Korbak et al. (2025): harm-aware verification is only as robust as the signal it reads, and the signal itself becomes a target for optimization pressure. The framework as currently formulated assumes a non-adversarial or cooperative agent; a deployment that does not satisfy this assumption risks *worse* oversight than uncertainty-only routing, because an adversarial agent need only suppress the harm cue rather than the broader uncertainty signal. A proper adversarial evaluation, constructing an agent trained to minimize its own verification rate against a known harm estimator, is substantial additional work we have not done. Until it is, this framework should not be deployed in settings where the agent has capability and motive to model its verifier.

The harm-weighting coefficient encodes a values judgment. The composite loss $\mathcal{L}_{\text{task}} + \beta\mathcal{L}_{\text{harm}}$ requires a choice of β that determines how aggressively a deployment trades off ordinary task failure against irreversible side effects. This choice is not technical. It reflects who bears the cost of task failures versus who bears the cost of irreversible side effects, and these are often different parties. A deployment that sets β to minimize cost to the deployer may not align with the preferences of users or third parties affected by the agent's actions. We flag this because framing β as an implementation detail — which we largely did in the method section of this paper — risks obscuring that it is also a governance detail.

Over-reliance on structural cues in regulated settings. The harm cue we evaluate is hand-engineered from declared tool-call metadata. In deployment contexts subject to regulatory or legal review, medical decision support, financial transaction authorization, legal document drafting, clinical coding, automated code execution with external effects on production systems, a structural cue that has not been calibrated against realized harm outcomes should not be the sole input to a screening decision. We treat the framework as a way to *allocate* human oversight effort under a budget, not as a substitute for human oversight. The paper's contribution is useful only to the extent that the verification capacity it routes is a genuine oversight mechanism; if the verifier itself is another language model with correlated failure modes (as in Inan et al. (2025); Luo et al. (2025)), the framework can offer at most a marginal improvement, not a guarantee.

Efficiency-enables-scale risk. We want to name one risk that is specific to oversight-routing frameworks. A framework that more efficiently allocates scarce human review effort can, in principle, lower the per-deployment cost of oversight enough to enable deployment of agent systems in settings that would not otherwise have passed oversight thresholds. Efficient oversight allocation could facilitate deployment at scale in settings where per-action scrutiny is abbreviated, shifting the question from "can we afford to review every action" to "can we afford to deploy this many autonomous agents." We do not think the paper's framework makes this risk qualitatively worse than existing work on runtime safeguards (Xiang et al., 2025; Chen et al., 2025), but we think it is worth stating clearly that more-efficient oversight is not automatically a safety improvement. Its net effect depends on how organizations use the efficiency gain: whether they reinvest it in deeper per-action review or use it to expand the set of actions that receive only cursory review. This is a governance question that the framework itself does not resolve.

Where we believe deployment is appropriate. To balance the concerns above, we identify one setting in which we believe the framework is reasonably deployable today: *internal tool-use pipelines within a single organization where agents are not subject to adversarial users, where the harm-weighting β can be set by a single accountable party, and where the verification budget is being allocated across internal review tasks rather than across externally-facing actions.* In this setting, the attack-surface and governance concerns above are substantially reduced. The framework is most naturally useful as a way to prioritize human attention, not as a replacement for human attention. We do not recommend deployment in externally-facing or adversarial settings until the robustness concerns above have been substantively addressed.

We have not identified positive societal impacts that are specific to this framework and that are not already shared with existing work on selective prediction, conformal abstention (Ren et al., 2023), and process reward modeling (Choudhury, 2025). The positive case for the paper is scientific — a clearer decomposition of a decision problem that currently uses the wrong proxy — rather than a claim that the framework, as implemented here, is deployment-ready.

References

Joshua Achiam, David Held, Aviv Tamar, and Pieter Abbeel. Constrained policy optimization. In *International Conference on Machine Learning*, 2017.

Dario Amodei, Chris Olah, Jacob Steinhardt, Paul Christiano, John Schulman, and Dan Mané. Concrete problems in AI safety. *arXiv preprint arXiv:1606.06565*, 2016.

Maksym Andriushchenko, Alexandra Souly, Mateusz Dziemian, Derek Duenas, Maxwell Lin, Justin Wang, Dan Hendrycks, Andy Zou, Zico Kolter, Matt Fredrikson, et al. AgentHarm: A benchmark for measuring harmfulness of LLM agents. In *The Thirteenth International Conference on Learning Representations*, 2025. URL `https://openreview.net/forum?id=AC5n7xHuR1`.

Yuntao Bai, Saurav Kadavath, Sandipan Kundu, Amanda Askell, Jackson Kernion, Andy Jones, Anna Chen, Anna Goldie, Azalia Mirhoseini, Cameron McKinnon, et al. Constitutional AI: Harmlessness from AI feedback. *arXiv preprint arXiv:2212.08073*, 2022.

Luca Beurer-Kellner, Beat Buesser, Ana-Maria Creţu, Edoardo Debenedetti, Daniel Dobos, Daniel Fabian, Marc Fischer, David Froelicher, Kathrin Grosse, Daniel Naeff, Ezinwanne Ozoani, Andrew Paverd, Florian Tramèr, and Václav Volhejn. Design patterns for securing LLM agents against prompt injections. *arXiv preprint arXiv:2506.08837*, 2025.

Collin Burns, Pavel Izmailov, Jan Hendrik Kirchner, Bowen Baker, Leo Gao, Leopold Aschenbrenner, Yining Chen, Adrien Ecoffet, Manas Joglekar, Jan Leike, Ilya Sutskever, and Jeff Wu. Weak-to-strong generalization: Eliciting strong capabilities with weak supervision. *arXiv preprint arXiv:2312.09390*, 2023.

Zhaorun Chen, Mintong Kang, and Bo Li. ShieldAgent: Shielding agents via verifiable safety policy reasoning. In *Forty-second International Conference on Machine Learning*, 2025. URL `https://openreview.net/forum?id=DkRYImuQA9`.

Sanjiban Choudhury. Process reward models for LLM agents: Practical framework and directions. *arXiv preprint arXiv:2502.10325*, 2025.

C. K. Chow. On optimum recognition error and reject tradeoff. *IEEE Transactions on Information Theory*, 16(1):41–46, 1970.

Shengda Fan, Xuyan Ye, Yupeng Huo, Zhi-Yuan Chen, Yiju Guo, Shenzhi Yang, Wenkai Yang, Shuqi Ye, Jingwen Chen, Haotian Chen, Xin Cong, and Yankai Lin. AgentProcessBench: Diagnosing step-level process quality in tool-using agents. *arXiv preprint arXiv:2603.14465*, 2026.

Sebastian Farquhar, Jannik Kossen, Lorenz Kuhn, and Yarin Gal. Detecting hallucinations in large language models using semantic entropy. *Nature*, 630:625–630, 2024.

Javier García and Fernando Fernández. A comprehensive survey on safe reinforcement learning. *Journal of Machine Learning Research*, 16:1437–1480, 2015.

Yonatan Geifman and Ran El-Yaniv. Selective classification for deep neural networks. In *Advances in Neural Information Processing Systems*, pp. 4878–4887, 2017.

Yonatan Geifman and Ran El-Yaniv. SelectiveNet: A deep neural network with an integrated reject option. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 2151–2159, 2019.

Zhibin Gou, Zhihong Shao, Yeyun Gong, Yelong Shen, Yujiu Yang, Nan Duan, and Weizhu Chen. CRITIC: Large language models can self-correct with tool-interactive critiquing. In *The Twelfth International Conference on Learning Representations*, 2024.

Nathan Grinsztajn, Johan Ferret, Olivier Pietquin, Philippe Preux, and Matthieu Geist. There is no turning back: A self-supervised approach for reversibility-aware reinforcement learning. In *Advances in Neural Information Processing Systems*, 2021.

Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q. Weinberger. On calibration of modern neural networks. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 1321–1330, 2017.

Wenyue Hua, Xianjun Yang, Zelong Li, Cheng Wei, and Yongfeng Zhang. TrustAgent: Towards safe and trustworthy LLM-based agents through agent constitution. *arXiv preprint arXiv:2402.01586*, 2024.

Hakan Inan, Kartikeya Upasani, Jianfeng Chi, Rashi Rungta, Krithika Iyer, Yuning Mao, Michael Tontchev, Qing Hu, Brian Fuller, Davide Testuggine, and Madian Khabsa. Monitoring LLM agents for sequentially contextual harm. In *ICLR 2025 Building Trust Workshop*, 2025. URL `https://openreview.net/pdf?id=LC0XQ6ufbr`.

Jaromír Janisch, Tomáš Pevný, and Viliam Lisý. Classification with costly features using deep reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pp. 3959–3966, 2019.

Carlos E. Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik Narasimhan. SWE-bench: Can language models resolve real-world GitHub issues? In *The Twelfth International Conference on Learning Representations*, 2024. URL `https://openreview.net/forum?id=VTF8yNQM66`.

Saurav Kadavath, Tom Conerly, Amanda Askell, Tom Henighan, Dawn Drain, Ethan Perez, Nicholas Schiefer, Zac Hatfield-Dodds, Nova DasSarma, Eli Tran-Johnson, et al. Language models (mostly) know what they know. *arXiv preprint arXiv:2207.05221*, 2022.

Megan Kinniment, Lucas Jun Koba Sato, Haoxing Du, Benjamin Goodrich, Max Hasin, Lawrence Chan, Luke Harold Miles, Tao R. Lin, Hjalmar Wijk, Joel Burget, et al. Evaluating language-model agents on realistic autonomous tasks. *arXiv preprint arXiv:2312.11671*, 2023.

Tomek Korbak, Mikita Balesni, Elizabeth Barnes, Yoshua Bengio, Joe Benton, Joseph Bloom, Mark Chen, Allen Cooney, Allan Dafoe, Anca Dragan, et al. Chain of thought monitorability: A new and fragile opportunity for AI safety. *arXiv preprint arXiv:2507.11473*, 2025.

Dawei Li, Yuguang Yao, Zhen Tan, Huan Liu, and Ruocheng Guo. ToolPRMBench: Evaluating and advancing process reward models for tool-using agents. *arXiv preprint arXiv:2601.12294*, 2026.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. *arXiv preprint arXiv:2305.20050*, 2023.

Stephanie Lin, Jacob Hilton, and Owain Evans. Teaching models to express their uncertainty in words. *Transactions on Machine Learning Research*, 2022.

Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hangliang Ding, Kaiwen Men, Kejuan Yang, et al. AgentBench: Evaluating LLMs as agents. *arXiv preprint arXiv:2308.03688*, 2023.

Hanjun Luo, Shenyu Zhang, Jiaxuan Yang, Tianjie Lin, Mingqing Yan, Jiarong Chen, Yuqing Yan, Cunqing Feng, Cewu Yin, Junhao Li, et al. AgentAuditor: Human-level safety and security evaluation for LLM agents. In *Advances in Neural Information Processing Systems*, 2025. URL `https://openreview.net/forum?id=2KKqp7MWJM`.

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. In *Advances in Neural Information Processing Systems*, 2023.

OpenClaw. OpenClaw: Personal AI assistant. `https://github.com/openclaw/openclaw`, 2026. GitHub repository, accessed 2026-04-22.

Hanli Peng, Yongsen Zheng, Ziyao Liu, and Kwok-Yan Lam. A survey of tool-call hallucinations in LLM-based agents. *arXiv preprint arXiv:2602.05214*, 2026.

Stephan Rabanser, Sayash Kapoor, Peter Kirgis, Kangheng Liu, Saiteja Utpala, and Arvind Narayanan. Towards a science of AI agent reliability. *arXiv preprint arXiv:2602.16666*, 2026.

Allen Z. Ren, Anushri Dixit, Alexandra Bodrova, Sumeet Singh, Stephen Tu, Noah Brown, Peng Xu, Leila Takayama, Fei Xia, Jake Varley, Zhenjia Xu, Dorsa Sadigh, Andy Zeng, and Anirudha Majumdar. Robots that ask for help: Uncertainty alignment for large language model planners. In *Proceedings of the 7th Conference on Robot Learning*, volume 229 of *Proceedings of Machine Learning Research*, pp. 661–682, 2023.

Yangjun Ruan, Honghua Dong, Andrew Wang, Silviu Pitis, Yongchao Zhou, Jimmy Ba, Yann Dubois, Chris J. Maddison, and Tatsunori Hashimoto. Identifying the risks of LM agents with an LM-emulated sandbox. In *The Twelfth International Conference on Learning Representations*, 2024.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. In *Advances in Neural Information Processing Systems*, volume 36, 2023. URL `https://openreview.net/forum?id=Yacmpz84TH`.

Amrith Setlur, Chirag Nagpal, Adam Fisch, Xinyang Geng, Jacob Eisenstein, Rishabh Agarwal, Alekh Agarwal, Jonathan Berant, and Aviral Kumar. Rewarding progress: Scaling automated process verifiers for LLM reasoning. In *The Thirteenth International Conference on Learning Representations*, 2025. URL `https://openreview.net/forum?id=A6Y7AqlzLW`.

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

Hanlin Wang, Jian Wang, Chak Tou Leong, and Wenjie Li. STeCa: Step-level trajectory calibration for LLM agent learning. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 11597–11614, 2025a. URL `https://aclanthology.org/2025.findings-acl.604/`.

Peiyi Wang, Lei Li, Zhihong Shao, R. X. Xu, Damai Dai, Yifei Li, Deli Chen, Y. Wu, and Zhifang Sui. Math-Shepherd: Verify and reinforce LLMs step-by-step without human annotations. *arXiv preprint arXiv:2312.08935*, 2023.

Wenxuan Wang, Shi Juluan, Zixuan Ling, Yuk-Kit Chan, Chaozheng Wang, Cheryl Lee, Youliang Yuan, Jen-tse Huang, Wenxiang Jiao, and Michael R. Lyu. Learning to ask: When LLM agents meet unclear instruction. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, 2025b. URL `https://aclanthology.org/2025.emnlp-main.1104/`.

Zhiheng Xi et al. AgentPRM: Process reward models for LLM agents via step-wise promise and progress. *arXiv preprint arXiv:2511.08325*, 2025.

Zhen Xiang, Linzhi Zheng, Yanjie Li, Junyuan Hong, Qinbin Li, Han Xie, Jiawei Zhang, Zidi Xiong, Chulin Xie, Carl Yang, Dawn Song, and Bo Li. GuardAgent: Safeguard LLM agents by a guard agent via knowledge-enabled reasoning. In *Forty-second International Conference on Machine Learning*, 2025.

Tianbao Xie, Danyang Zhang, Jixuan Chen, Xiaochuan Li, Siheng Zhao, Ruisheng Cao, Toh Jing Hua, Zhoujun Cheng, Dongchan Shin, Fangyu Lei, Yitao Liu, Yiheng Xu, Shuyan Zhou, Silvio Savarese, Caiming Xiong, Victor Zhong, and Tao Yu. OSWorld: Benchmarking multimodal agents for open-ended tasks in real computer environments. In *Advances in Neural Information Processing Systems, Datasets and Benchmarks Track*, 2024.

Ji Xin, Raphael Tang, Yaoliang Yu, and Jimmy Lin. The art of abstention: Selective prediction and error regularization for natural language processing. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics*, pp. 1040–1051, 2021.

Weihao Xuan, Qingcheng Zeng, Heli Qi, Yunze Xiao, Junjue Wang, and Naoto Yokoya. The confidence dichotomy: Analyzing and mitigating miscalibration in tool-use agents. *arXiv preprint arXiv:2601.07264*, 2026. URL https://arxiv.org/abs/2601.07264.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. ReAct: Synergizing reasoning and acting in language models. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=WE_vluYUL-X.

Michael J. Zellinger and Matthew Thomson. Cost-saving LLM cascades with early abstention. *arXiv preprint arXiv:2502.09054*, 2025.

Zhexin Zhang, Shiyao Cui, Yida Lu, Jingzhuo Zhou, Junxiao Yang, Hongning Wang, and Minlie Huang. Agent-SafetyBench: Evaluating the safety of LLM agents. *arXiv preprint arXiv:2412.14470*, 2025.

Qiwei Zhao, Dong Li, Yanchi Liu, Wei Cheng, Yiyou Sun, Mika Oishi, Takao Osaki, Katsushi Matsuda, Huaxiu Yao, Chen Zhao, Haifeng Chen, and Xujiang Zhao. Uncertainty propagation on LLM agent. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 6064–6073, 2025. URL https://aclanthology.org/2025.acl-long.302/.

Shuyan Zhou, Frank F. Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, Uri Alon, and Graham Neubig. WebArena: A realistic web environment for building autonomous agents. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=oKn9c6ytLx.

A Extended Discussion of Remark 1

Remark 1 captures the allocation structure our online policy uses, but it deliberately side-steps two hard features of the full problem.

Sequential coupling. Verifying step t can change which actions are proposed at $t{+}1, t{+}2, \ldots$ and therefore the future p, H, and c values. Assumption (iii) in the remark suppresses this.

Budget coupling. Using the budget at t changes what is available later; the Lagrangian relaxation with fixed λ handles this only asymptotically. A proper sequential bound would require either a dynamic-programming argument over a tractable reduction of the state, or a regret-style analysis against an oracle that sees all trajectories in advance. We do not claim such a result.

Relation to generic expected utility. Expected utility by itself does not specify which local quantity should govern online intervention. The content of the remark is that intervention value, when decomposed at the step level, factors into four interpretable terms — error likelihood p_t, downstream consequence H_t, verifier efficacy q_t, and cost c_t — of which uncertainty-based policies model only the first. This is the source of the paper's main ranking principle, not the remark itself.

Informal sketch of a sequential bound. We do not prove a sequential bound, but the form one would take is worth sketching because it clarifies what the remark is not doing. Consider the fully online problem with a budget B of verification opportunities over a horizon of T steps, and let an oracle policy $\pi^\star$ have access to the true (p_t, H_t, q_t, c_t) values at every step. Let π be a policy that operates on estimates $(\hat{p}_t, \hat{H}_t, \hat{q}_t)$ with bounded per-step estimation error and uses the myopic rule of Remark 1 with a budget-control term $\lambda(b_t, t)$. A natural quantity of interest is the expected regret

$$R_T(\pi; \pi^\star) = \mathbb{E}\left[\sum_{t=1}^{T} \mathcal{L}_t^{\pi^\star} - \sum_{t=1}^{T} \mathcal{L}_t^{\pi}\right],$$

where $\mathcal{L}_t$ is the per-step contribution to the composite loss. As a concrete assumption set, suppose: (A1) per-step estimation errors are bounded uniformly, $|\hat{p}_t - p_t| \le \epsilon_p$, $|\hat{H}_t - H_t| \le \epsilon_H$, $|\hat{q}_t - q_t| \le \epsilon_q$, almost surely; (A2) the agent's proposal distribution at step $t{+}1$ is L-Lipschitz in a suitable divergence with respect to the perturbation induced by verifying versus not verifying at step t; (A3) the composite loss $\mathcal{L}_t$ is bounded, $|\mathcal{L}_t| \le B_{\mathcal{L}}$; and (A4) the budget multiplier $\lambda(b_t, t)$ is chosen so that its path satisfies a standard regret-minimization condition against the oracle's budget usage (e.g., a doubling or adaptive schedule). Under (A1)–(A4), we would expect $R_T(\pi; \pi^\star)$ to decompose into (a) a static-allocation term of order $T \cdot (\epsilon_p + \epsilon_H + \epsilon_q)$ captured by Remark 1, and (b) a propagation term that accumulates whenever the myopic choice at t produces a different proposal distribution at $t{+}1$ than the oracle would have, bounded by $L \cdot T \cdot B_{\mathcal{L}}$ scaled by the fraction of steps on which π and $\pi^\star$ disagree. The second term is what the remark does not control, and it is also the term a formal analysis would need to bound non-trivially — the $L \cdot T$ scaling above is a worst case that does not use the problem structure.

We flag three concrete obstacles to such a bound. First, the propagation term depends on the agent's response to verification, which is a property of the agent rather than of the verification policy; this is structurally similar to the "policy perturbation term" that arises in off-policy correction analyses in reinforcement learning (e.g., the advantage-estimation bounds used in trust-region and proximal policy optimization methods), and techniques developed there for bounding the effect of small policy changes on the state-visitation distribution may be adaptable to this setting. Second, the budget-control multiplier $\lambda(b_t, t)$ interacts with $\hat{V}_t$ multiplicatively, so standard doubly-robust arguments that assume additive decomposition of error sources do not apply directly. Third, the product structure $\hat{V}_t = \hat{q}_t \hat{p}_t \hat{H}_t$ means that the static-allocation term is bounded by a product of the individual errors times their magnitudes, not a sum; a bound that treats them separately will be loose in the regime where one component dominates. A rigorous treatment is beyond the scope of the present paper but is, in our view, the single most valuable theoretical next step for the framework.

B Extended Method Discussion

Alternative policy implementations. The decomposition $\hat{V}_t = \hat{q}_t \hat{p}_t \hat{H}_t - \lambda(b_t, t) c_t$ in §3.4 estimates the three components separately. In some settings, it may be easier to learn a single composite score that approximates expected loss reduction directly, absorbing q_t into the score. In others, verifier efficacy may be treated as constant and estimated offline. What should remain invariant across implementations is the decision structure: verification should be triggered by an estimate of intervention value, not by uncertainty alone.

Why this is not selective prediction. The method is not selective prediction in the usual sense. The system does not abstain from producing an output and hand the example off entirely; instead, it decides whether to intervene on the current step before the trajectory continues. Uncertainty remains useful, but only as one component of a broader intervention-value estimate. This distinction matters because selective prediction guarantees (e.g. coverage, risk-control) are formulated in terms of abstention on an i.i.d. input distribution, which does not apply to a sequential agent trajectory where intervention alters subsequent inputs.

Why proxy features are acceptable. Our formulation does not assume that downstream harm is directly observable. In realistic agent settings, H_t is counterfactual and can usually be estimated only approximately, through learned predictors, task-specific signals, or structured heuristics. That limitation is real but does not make the decision target vacuous: many high-value control problems rely on imperfect proxies for quantities that are not directly observed. The relevant question is not whether H_t can be known exactly, but whether modeling harm *separately* from local error probability yields better intervention decisions than treating all errors as equally consequential.

Why the method is not ad-hoc reweighting. If one estimates only p_t, all decisions reduce to uncertainty ranking. If one estimates only H_t while ignoring error likelihood, the policy may over-invest in rare but dramatic failure modes. The method requires both: a step is worth checking when it is plausibly wrong *and* costly to leave uncorrected, subject to verification efficacy and budget cost.

C Rationale for the Planned Experiments

This appendix expands the reasoning behind each of the three planned experiments summarized in §4.5.

Matched-budget comparison. The pilot in §4.4 leaves open a natural reviewer objection: the harm-aware policy verifies more episodes than the uncertainty-only policy (3/4 vs. 1/4), so the performance gap could reflect verification *volume* rather than *allocation quality*. The matched-budget comparison eliminates this confound by evaluating all three policies at the same expected verify rate. `random_verify`$_\rho$ is included because it is the most conservative "uses the same amount of budget" baseline; if random verification at the same rate achieves comparable irreversible-failure rates, there is little evidence that any particular allocation rule matters on this slice. `uncertainty_only`$_{\theta'}$ with a lowered threshold ensures the uncertainty-only policy cannot be faulted for under-spending. We precommit to ≥ 50 episodes per policy per slice; this is the minimum scale at which a Fisher exact test can reject the null at conventional significance for the effect sizes we would consider informative. If harm-aware routing does not outperform both matched-rate comparators on irreversible failure, the paper's claim should be reframed as "more verification on this slice is better."

Cross-slice robustness. The paper's central prediction is specifically that harm-aware allocation matters when the public signal and downstream consequence are misaligned. On a slice where they are aligned, the two selective policies should agree on which episodes to verify, and their performance gap should vanish. This is a falsification test: a cross-slice result showing a comparable advantage on `low_risk` would actually weaken the paper, because it would indicate the harm-aware cue is doing work that is not tied to the mechanism we claim. The statistical object of interest is therefore the slice×policy interaction, not the per-slice means.

Process-reward baseline. A PRM trained to predict end-of-trajectory success provides a natural learned alternative to our hand-engineered harm cue. We include two variants to separate two distinct concerns. `prm_threshold`^{task} is trained on task success alone; it implicitly entangles p_t and H_t but only through their contribution to task reward, so it may underweight irreversible side effects that do not directly reduce task success. `prm_threshold`^{harm} is trained on the harm-weighted loss $\mathcal{L}_{\text{task}} + \beta\mathcal{L}_{\text{harm}}$, which in principle should absorb the harm signal into a single learned score. If `prm_threshold`^{task} matches our policy, the paper's contribution reduces to a decision-theoretic framing around an already-sufficient signal. If `prm_threshold`^{harm} matches our policy but `prm_threshold`^{task} does not, the paper's contribution is the training-signal specification, and our hand-engineered cue is a weaker proxy for the same thing. If neither PRM matches our policy, decoupling p_t and H_t as separate decision-time inputs provides information that a single learned scalar does not, and the paper's framework has standalone value.

D Detailed PRM Comparison

A PRM trained to predict end-of-trajectory success implicitly entangles p_t and H_t into a single score: a step with high local error probability but low downstream harm may be scored similarly to a step with moderate error probability and large downstream harm, because both reduce the predicted probability of final success by a comparable amount. That conflation is acceptable — even desirable — when the goal is to rank steps by expected contribution to task reward. It is less appropriate when the downstream impact is weighted differently from task success, as in our loss formulation $\mathcal{L}_{\text{task}} + \beta\mathcal{L}_{\text{harm}}$. In such settings, two steps with the same PRM-predicted success probability can have very different verification values.

Our formulation keeps error likelihood and harm magnitude as separate inputs, so they can be weighted separately at decision time. Thresholding a PRM score is therefore a legitimate instance of our framework in the special case $\beta = 0$ and constant q_t, c_t, but is not equivalent to it in general. The planned PRM baseline in §4.5 tests this directly: we compare our policy against a PRM trained on task success alone and a PRM trained on the harm-weighted loss. If the harm-weighted PRM matches our policy, the paper's practical contribution reduces to a re-derivation of the right training signal; if it does not, decoupling p_t and H_t at decision time provides information that a single scalar score does not capture.

E Full Threats to Validity Discussion

This appendix expands each paragraph summarized in §6.

Construct validity: the harm score is a correlate, not an estimator. §3.2 defines H_t as the expected difference in episode loss between error and corrected continuations, a counterfactual object that in general requires either a learned dynamics model or extensive rollouts to estimate. The empirical harm score used in §4 does not estimate this quantity; it is a hand-engineered function of side-effect class, downstream dependency count, and a rollback flag, and is better described as a correlate of H_t on our slice than as a calibrated estimator. Two implications follow. First, the experiments support the decision structure — error likelihood and harm magnitude as separate inputs — more strongly than they support any particular harm score. Second, a threshold that works on `dependency_sensitive` may not transfer to a slice with different structural features. A calibrated harm estimator, ideally one with a coverage guarantee in the spirit of conformal prediction (Ren et al., 2023), is a substantive research direction we do not attempt here.

Internal validity: the harm cue was specified in the context of the slice. Although the harm cue does not read the slice label or ground-truth outcome (§4.1), it was designed in a development process where the authors had access to both. The risk is that we iterated until the cue worked on the slice we chose. The matched-budget comparison (§4.5) partially addresses this by introducing a random baseline at the same verify rate, but a stronger test would be to specify the cue on one held-out slice and evaluate on another. The cross-slice falsification test in §4.5 is our first move in that direction; we acknowledge it is not as rigorous as a pre-registered cue.

External validity: OpenClaw is not a standard substrate. The OpenClaw-based sandbox is a new evaluation environment. We chose it because it exposes the public-signal / structural-feature / verification-decision interface the framework needs, but it is not widely used in the agent-safety community. A natural complementary evaluation is in ToolEmu (Ruan et al., 2024), whose risk taxonomy is designed around exactly the state-changing, irreversibility-laden tool actions we care about. We treat a ToolEmu port as high-priority future work. Until that exists, our results should be read as showing that the ranking principle can separate policies in an executable agent loop, not that the specific numerical margins generalize.

Statistical validity: pilot-scale results do not distinguish mechanism from noise. At $n = 4$, the irreversible failure contrast 2/4 vs. 0/4 in Table 1 is fully consistent with noise — a Fisher exact test on the 2×2 contingency yields $p \approx 0.43$. We report the pilot as mechanism evidence rather than statistical evidence, and the planned experiments in §4.5 commit to ≥ 50 episodes per policy per slice with reported standard errors and a significance test. A reader who requires power-adequate evidence of effect size before accepting the claim should consult those results directly.

Adversarial robustness: the framework assumes a non-adversarial agent. The decomposition $V_t = q_t p_t H_t - c_t$ assumes the quantities on the right-hand side are honestly reported. In a deployment where the agent is itself optimizing against the verification policy — for example, by presenting high-harm actions in a form that scores low on the structural harm cue — the harm-aware policy can be manipulated in ways that uncertainty-only policies cannot. This connects directly to the chain-of-thought-monitorability concern in (Korbak et al., 2025): harm-aware verification is only as robust as the harm signal it reads, and that signal is a new attack surface. A proper adversarial evaluation requires constructing an agent trained to minimize its own verification rate given a known harm estimator, which is substantial additional work. Until it is done, the framework should be understood as a tool for cooperative or non-adversarial deployment.

Verifier model: $q_t = 1$ is a strong idealization. The current experiments hold verifier efficacy fixed at $q_t = 1$, meaning the verifier is assumed to identify and correct every erroneous step it is applied to. Real verifiers are imperfect and their errors can correlate with agent errors — both are produced by language models that may share systematic failure patterns. The formalism accommodates imperfect q_t through the factorization, but the experiments do not exercise it. Varying q_t is listed in Future Work; the magnitude of the effect we report in §4 should be expected to shrink as q_t falls below 1, because some consequential errors will now be missed even when the policy correctly routes verification to them.

Scope: one-opportunity-per-episode is a restricted instantiation. The pilot allows exactly one eligible verification decision per episode, which is useful for isolating routing but narrower than the full online problem in §3.1. In multi-opportunity trajectories, verification decisions interact through the budget and through their effect on subsequent agent proposals. Remark 1 is explicit that it does not resolve this case; the full problem likely requires either dynamic programming over a tractable state reduction or a regret-style analysis.

Objective sensitivity: results are conditional on the harm-weighting coefficient. The composite loss $\mathcal{L}_{\text{task}} + \beta \mathcal{L}_{\text{harm}}$ determines how aggressively the allocation rule trades off ordinary task failure against irreversible side effects. In this paper, β is implicit in the sandbox's reward structure, and we do not vary it systematically. Different deployments will reasonably choose different β, and as $\beta \to 0$ the ranking principle collapses back onto task-success ranking (and therefore onto task-trained PRM scores). Readers should interpret the results as conditional on a non-trivial weight on irreversibility.

Reproducibility of Table 1. The specific reward values in Table 1 follow from the OpenClaw sandbox's reward function, which combines task-success reward, per-verification cost, and harm penalties in proportions fixed by the sandbox configuration. To allow a reader to independently reconstruct the mechanism the paper argues for, we release a separate Supplementary Material document that specifies: (a) the reward function used to compute Table 1; (b) the harm-cue implementation in pseudocode, with the structural features and threshold exactly as used in the pilot; (c) the verifier oracle's behaviour; and (d) the four episode specifications used in the `dependency_sensitive` slice, including public signal value, harm-cue features,

ground-truth correctness, oracle correction if applicable, and outcome under each of the four pilot policies. We note two remaining limitations of this release. First, the full OpenClaw sandbox source, including the execution substrate and tool registry, is not released with this submission and remains under preparation for a separate open-source release; the supplementary document is sufficient for a reimplementation of the four episodes in the minimal framework needed to reproduce Table 1's direction of effect, but not for running arbitrary tool-use experiments. Second, the supplementary document discloses that the reconciled reward values under the minimal reward function differ from those originally reported in Table 1 by small additive terms from the sandbox's full reward decomposition; reconciling the two exactly is a camera-ready item. The direction of effect (the paper's substantive claim) is robust to this reconciliation.

F Supplementary Toy Environment

The toy environment is a small synthetic sandbox designed to complement the OpenClaw-based pilot by isolating the mechanism in a transparently inspectable setting. Each episode consists of a short sequence of discrete decisions, some of which write to a shared state that subsequent decisions read from. Errors on state-writing steps can be either recoverable (later steps overwrite them) or irreversible (later steps commit to the incorrect value). A tunable uncertainty signal u_t parametrizes the signal–consequence discordance studied in the main experiments: episodes can be configured to be concordant (uncertainty and harm agree), discordant (uncertainty low but harm high, or vice versa), or uniformly low-harm. The environment itself is specified at a level of detail that permits plain-text release alongside a future revision of this paper; this is one reason we view it as a natural home for follow-up experiments that address reproducibility concerns without requiring release of the OpenClaw sandbox.

We do not report experimental results from the toy environment in the present submission. The design of a sanity-check run, together with its predicted outcomes, is specified as future work in Appendix G.

Why the toy environment does not substitute for the OpenClaw-based pilot. The toy environment can establish that the ranking principle separates policies in a setting the authors designed, but it cannot establish that the mechanism appears in an executable agent loop with a real language-model agent proposing actions, a real tool interface, and realistic state propagation. That is the role of the OpenClaw-based experiments (§4). A toy-environment result that cleanly supports the mechanism would strengthen the paper's claim but would not close the gap to deployment; a full evaluation still requires the OpenClaw-based comparisons specified in Appendix C.

G Extended Future Work

Beyond the planned experiments in §4.5, several directions would substantially strengthen the framework.

Toy-environment sanity check ($n = 200$). As flagged in Appendix F, the toy environment supports cheap large-sample experiments that address the statistical-validity concern in §6 without requiring release of the OpenClaw sandbox. The specific design we consider highest-priority is: $n = 200$ episodes per policy, five policies (`no_verify`, `always_verify`, `uncertainty_only`, `harm_aware_selective`, plus `random_verify`$_\rho$ at a verify rate matched to `harm_aware_selective`), evaluated across three episode mixes: mix A (50% discordant, 50% concordant), mix B (100% concordant), and mix C (100% low-harm). The predictions from §3 are: (i) harm-aware should outperform matched-rate uncertainty-only and matched-rate random on irreversible failure in mix A, (ii) the gap should shrink in mix B where both selective policies agree on which episodes to verify, and (iii) the gap should be near-zero in mix C where neither signal predicts meaningful harm. The toy environment also supports an additional experiment worth running: *imperfect verifiers with correlated errors*. Setting q_t to the efficacy of a noisy verifier whose errors are deliberately correlated with agent errors — e.g., both being produced by the same base distribution — would quantify the concern in §6 that the framework's advantage shrinks under realistic verifier imperfection. This experiment would answer a concrete question that the $q_t = 1$ pilot cannot: does the harm-aware policy retain its advantage when the verifier shares the agent's failure modes?

Online multi-step allocation. Extending the formulation to fully online multi-step allocation, where current verification choices change future p, H, and c values, is the most natural theoretical next step. This likely requires either dynamic programming over a tractable state reduction or a regret-style analysis against an oracle.

Learned harm estimators. Learning harm estimators f_H directly from trajectory-level supervision — rather than from handcrafted structural features — would make the method applicable to settings the authors did not design. Candidate supervision signals include counterfactual rollouts, learned dynamics models, or human-annotated harm labels.

Imperfect verifiers. Varying verifier strength q_t away from 1 is essential for deployment-relevant results. The experiments should characterize how the effect size depends on q_t, and in particular whether the harm-aware policy's advantage over uncertainty-only routing survives realistic verifier error rates.

Theoretical guarantees. A regret-style bound for harm-aware ranking against an oracle that observes (p_t, H_t) directly would complement the empirical evidence. A conformal-prediction wrapper around the harm score, analogous to KnowNo's wrapper around uncertainty (Ren et al., 2023), would yield a finite-sample coverage guarantee and is a natural bridge to the selective-prediction literature.

Runtime safeguard integration. Connecting harm-aware routing to runtime safeguards (Xiang et al., 2025; Chen et al., 2025; Inan et al., 2025) — specifically, treating the safeguard invocation decision itself as a budgeted allocation problem — is where we expect the framework to have the most practical impact.