LEARNING TO VERIFY: ALLOCATING REVISION EFFORT IN EMBODIED AGENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Improving an embodied agent requires recognizing when stable execution fails to satisfy an instruction. A robot may remain upright while moving in the wrong direction, traveling too far, or performing actions out of order. We propose Learn to Verify, then Revise (LVR), which turns feedback into executable task checks and uses them to guide subsequent revisions. LVR selects clarification questions by their expected effect on verification decisions, then reuses the learned checks without further feedback. Our analysis characterizes when a shared check is reliable enough for additional candidate search not to reduce expected task success. On recorded revision chains for 47 simulated robot tasks, LVR achieves 87.2% success, compared with 76.6% when every task is revised, while using 58% fewer revisions. With separately generated revision chains on 39 tasks, updated checks achieve 89.7% success versus 74.4% before feedback, selecting 50 versus 40 revisions under offline stopping. Classification and transfer tests show no corresponding established gain in held-out accuracy. These results motivate evaluating learned verification through the revision decisions it enables, alongside its predictive accuracy.

1 INTRODUCTION

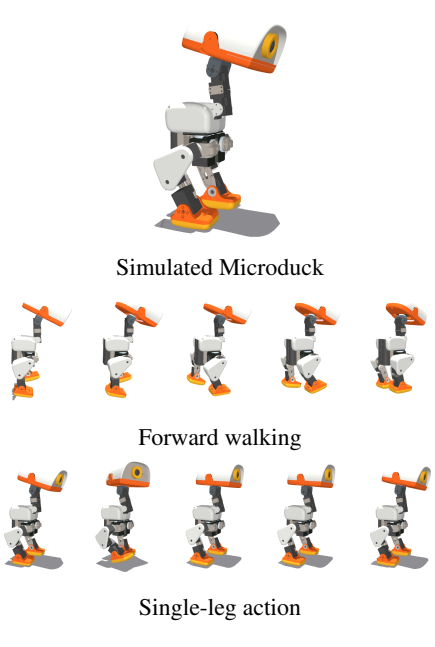
Improving an AI agent requires a way to determine whether its outputs satisfy the task. Learned verifiers provide such a signal by assessing candidate solutions and guiding which ones to retain or revise (Cobbe et al., 2021; Lightman et al., 2024). In embodied tasks, however, successful execution is only part of this assessment. A robot may remain upright and track a motion accurately while walking too short a distance, turning in the wrong direction, or performing actions in the wrong order. The distinction matters whenever an agent uses its own assessment to decide what to do next.

Feedback can improve this assessment by clarifying what should be checked. For example, feedback about a walking attempt may specify that distance should be measured relative to the robot's initial heading. Recording that interpretation as an executable check allows the agent to assess subsequent attempts without requesting the same clarification again. The resulting learning problem concerns both the content of a check and the tasks to which it applies. A check applied too narrowly cannot be reused; one applied too broadly may reject valid behavior.

We study a verification-guided revision framework in which feedback updates task checks before those checks allocate later revisions. Each check specifies an observable trajectory feature, a condition on that feature, and an applicability scope. The agent selects questions by their expected effect on decisions across development tasks. Once feedback ends, the checks are frozen: a failed check identifies an attempt for repair and supplies a diagnostic to the planner (Figure 1). The low-level controller remains fixed throughout.

A useful verifier must support good revision decisions as well as accurate judgments. Consider a robot asked to walk forward 40 centimeters. Detecting that it stopped short can prompt a useful revision; incorrectly rejecting a successful motion can trigger a revision that makes it worse. We therefore evaluate checks both by whether they correctly judge a motion and by whether using them to guide revisions improves task success. A language model supplies feedback during check learning; independent specifications score task outcomes.

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.



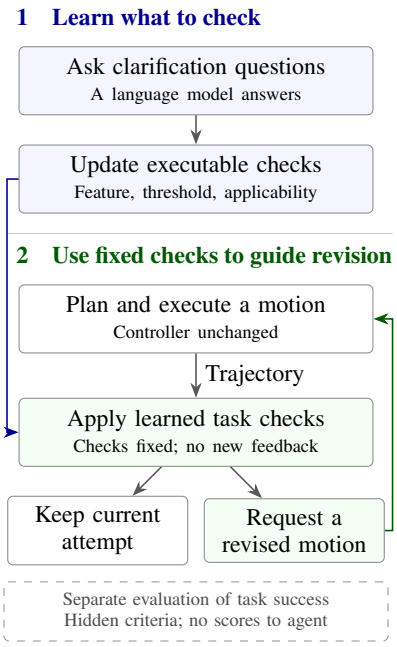


Figure 1: Verification for embodied repair. **Left:** simulated Microduck and base-library motions, illustrating the platform rather than repair outcomes. **Right:** model responses update executable task checks. Fixed checks then select attempts and diagnose revisions. The diagram shows the policy; stopping is evaluated offline on generated chains. A separate evaluator scores task success using hidden criteria unavailable to the agent.

1.1 Related work

Learning to verify connects learned evaluation, active feedback acquisition, and robot adaptation. We distinguish prior work by what feedback changes and how the resulting evaluator guides behavior.

Learned verifiers use supervision about solution quality to guide the selection of generated answers. Cobbe et al. (2021) train a verifier for mathematical solutions; Uesato et al. (2022) and Lightman et al. (2024) examine supervision on outcomes and intermediate steps. Math-Shepherd constructs step-level supervision automatically (Wang et al., 2024). Our verifier evaluates executed trajectories using explicit conditions on observable features. This representation lets a failed condition identify a requirement for the repairer, in addition to producing a task verdict. We measure both the verdict's accuracy and the outcome of acting on it.

A verifier's errors can change whether generating more candidates helps. Reward-model overoptimization can improve a proxy score while reducing performance under the intended objective (Gao et al., 2023). Work on resampling and test-time verification examines the resulting limits on additional computation (Stroebl et al., 2024; Dorner et al., 2026); Dorner et al. (2026) relate these effects to the verifier's receiver operating characteristic curve. Our analysis models uncertainty in a check learned once and shared across subsequent candidates. Under its false-acceptance assumptions, we characterize how reliable that check must be for success not to decrease over a specified search budget.

Active feedback acquisition asks which answer would help the learner make a better decision. Preference-learning methods select trajectory comparisons and account for factors such as question difficulty and the availability of demonstrations (Sadigh et al., 2017; Bıyık & Sadigh, 2018; Bıyık et al., 2019; 2022). Myers et al. (2023) use expected value of information to choose online preference queries. We formulate the queried uncertainty as alternative executable checks, including their thresholds and applicability. The query score measures their anticipated effect on accept, reject, and abstain decisions across a development pool, relative to response cost. Whether this score improves on simpler querying rules is an empirical question tested separately from the revision policy.

Feedback cannot resolve a requirement that the learner has no way to represent. Feature Expansive Reward Learning and its extension acquire missing features for reward learning (Bobu et al., 2021; 2022b). Other work examines misspecified hypothesis spaces and acquires perceptual concepts through queries (Bobu et al., 2020; 2022a). Merker et al. (2026) request demonstrations targeted at uncertain features of an inferred reward. Our experiments retain a fixed trajectory-feature vocabulary and update the conditions imposed on those features. This permits a direct audit of thresholds and applicability, while leaving feature acquisition as a separate problem.

Language feedback can change a robot's plan, its training objective, or its assessment of an executed motion. KnowNo calibrates when a planner should request help (Ren et al., 2023), while Cui et al. (2023) use language corrections during manipulation. Language-based reward design converts task descriptions into objectives (Kwon et al., 2023; Yu et al., 2023). Eureka refines generated reward code using training feedback, and Text2Reward supports refinement through human feedback (Ma et al., 2024; Xie et al., 2024). In our setting, feedback updates trajectory checks; their diagnostics subsequently guide revisions to reference motions under a fixed controller. The adaptation therefore occurs in verification and reference generation.

Motion-tracking methods make it possible to study task interpretation while keeping the controller fixed. Motion imitation and learned skill representations provide reusable behaviors (Peng et al., 2018; 2021; 2022), and humanoid tracking and whole-body control extend the range of reference motions that can be executed (Luo et al., 2023; 2024; Cheng et al., 2024). Our planner composes a restricted motion library. We then evaluate whether the executed result satisfies the instruction, separately from whether the controller tracks the reference stably.

Updating the evaluator itself is also a subject of research on self-improving language models. Meta-Rewarding uses a meta-judge to assess judgments and improve the model's evaluation capability (Wu et al., 2025). Our update acts on explicit trajectory checks rather than a language model's judging parameters. We freeze those checks before repair and test their subsequent use. This protocol separates the effect of learning the evaluator from the decisions made with a particular learned version.

1.2 Main Contributions

First, we formulate verification learning through its effect on revision decisions. LVR learns executable checks of trajectory features, thresholds, and applicability, then freezes them to select attempts and diagnose revisions under a fixed controller.

Second, we characterize how shared-check errors affect further search. Under explicit false-acceptance assumptions, we derive a reliability condition for additional candidates not to reduce expected success and relate clarification value to uncertainty reduction and reuse across tasks.

Third, we evaluate learned verification through classification, transfer, and revision allocation. On shared chains, LVR achieves 41/47 successes with 59 selected revisions versus 36/47 with 141 for repair-all. On separately generated chains, updated checks achieve 35/39 versus 29/39 before feedback, selecting 50 versus 40 revisions under offline stopping. Repair gains coexist with no established held-out classification gain, motivating evaluation beyond predictive accuracy.

2 Learning Checks and Allocating Revisions

LVR learns an interpretation of task success, then uses it to guide behavior repair. We first define the task and evaluation setting, then describe check representation, feedback updates, query selection, and repair. The controller remains fixed throughout; only the checks are updated during learning and the reference motions during repair. Figure 2 shows the information flow; Appendix C lists the implementation stages.

2.1 Problem Formulation

A task consists of an observed instruction x and an evaluation specification s hidden from the agent. A generator proposes a reference motion $r \in \mathcal{R}$, which a fixed controller π executes with simulator seed ξ to produce $\tau = \mathrm{Rollout}(\pi, r, \xi)$. Execution diagnostics $E(\tau)$ describe properties such as

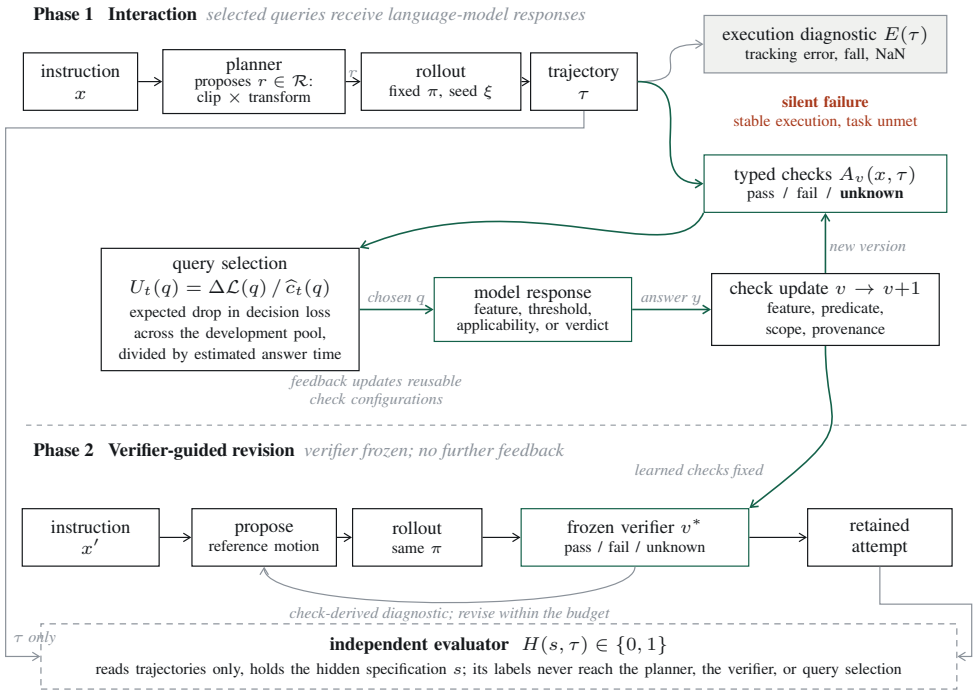


Figure 2: Learning and reusing task checks. Selected model responses update check configurations. The frozen verifier then supplies decisions and diagnostics for reference-motion repair under the same controller. Hidden specifications are used only by the independent evaluator. Reported revision chains are generated before offline stopping is applied; preference ranking is not evaluated.

tracking and stability. The independent evaluator $H(s, \tau) \in \{0, 1\}$ instead measures satisfaction of the task specification. Passing the execution diagnostics need not imply task success.

The verifier predicts task acceptance from the instruction and the observed trajectory. We write its output as $A_v(x, \tau) \in \{\text{pass}, \text{fail}, \text{unknown}\}$, allowing it to abstain when the represented requirements are unresolved. Feedback updates parameters v within a finite space of check configurations. The agent has access to its instruction, trajectories, and query responses, but not to hidden specifications or test labels. Development tasks support configuration and query-value estimation; interaction tasks supply feedback; locked test tasks measure transfer. Splits are assigned by semantic template.

2.2 Representing task requirements

A task check specifies when a requirement applies and how to test it on an executed trajectory. Its applicability condition uses the public task family, template, and instruction keywords. Its criteria specify a feature, unit, predicate, and threshold or interval; its metadata records the feedback source and version. Features include signed displacement, heading change, trunk orientation, and leg events. Missing observations produce *unknown*, while an inapplicable check contributes no evidence. These distinctions prevent missing information from being treated as either failure or success.

Learning selects among explicit interpretations within a fixed check vocabulary. The implementation extracts instruction slots using fixed rules and enumerates configurations that vary the primary feature, threshold tier, applicability scope, and inclusion of secondary constraints. For example, a forward-walking request can be checked using net forward displacement or total path length; these disagree when the robot moves sideways or returns toward its starting position. Template-level and family-level scopes determine where the resulting criterion can be reused. Enumeration retains at most sixteen configurations per task, together with a catch-all hypothesis that predicts unknown. Thus, an update changes the interpretation of trajectory evidence without retraining the controller or synthesizing unrestricted verification code.

2.3 UPDATING CHECKS FROM FEEDBACK

Feedback changes the probability assigned to each candidate check configuration. Let $p_t(\theta)$ denote the belief over configurations after t responses. A question q asks about a threshold, applicability, feature choice, or the verdict on an observed trajectory. Given answer y, the update is

$$p_{t+1}(\theta) = \frac{p_t(\theta)P(y \mid q, \theta)}{\sum_{\theta'} p_t(\theta')P(y \mid q, \theta')}. \tag{1}$$

The response likelihood expresses how likely each answer would be under a candidate interpretation. A feature-choice answer favoring net displacement, for example, increases the weight on configurations that use that feature; it does not directly repair the current motion. The implementation normalizes beliefs separately for each task. Applying the same observation to configurations associated with other tasks uses a pseudo-likelihood approximation, rather than a joint Bayesian model of all tasks. The response model and its calibration are specified in Appendix F.

The posterior determines whether the trajectory should be accepted, rejected, or left unresolved. Applicable checks produce a predictive distribution over pass, fail, and unknown. The suite aggregation used for transfer permits an applicable failed check to reject a trajectory; own-task aggregation restricts evidence to checks associated with that task. Given the predictive distribution, the implemented decision rule minimizes expected loss: an incorrect acceptance or rejection costs 1, abstention costs 0.3, and accepting or rejecting an unknown hypothesis outcome costs 0.5. The aggregation setting therefore affects both the verdict and whether a scope question can change it. Experiments B and C state which setting is used.

2.4 CHOOSING QUESTIONS BY THEIR EFFECT ON DECISIONS

The query rule favors answers expected to improve decisions across the development trajectory pool. Let $\mathcal{P}_{\mathrm{dev}}$ contain task–trajectory pairs, and let $\mathcal{D} = \{\mathrm{accept}, \mathrm{reject}, \mathrm{abstain}\}$. The loss $\ell_\theta(d, x, \tau)$ compares decision d with the verdict implied by configuration θ, using the costs above. The expected decision risk is

$$\mathcal{L}(p) = \frac{1}{|\mathcal{P}_{\mathrm{dev}}|} \sum_{(x,\tau)\in\mathcal{P}_{\mathrm{dev}}} \min_{d\in\mathcal{D}} \mathbb{E}_{\theta\sim p}[\ell_\theta(d, x, \tau)]. \tag{2}$$

For each eligible question, the algorithm enumerates its possible answers, forms hypothetical posterior updates, and recomputes this risk. It selects questions using the expected reduction per unit of modeled response cost:

$$U_t(q) = \frac{\mathcal{L}(p_t) - \mathbb{E}_{y\sim p_t(y|q)}[\mathcal{L}(p_t(\theta \mid q, y))]}{\widehat{c}_t(q) + \epsilon}, \qquad \epsilon > 0. \tag{3}$$

Here the answer distribution is computed from the belief of the task being queried. Response costs are assigned by question type, and the experiments share cached language-model answers across query strategies. After a response, the belief and eligible questions are updated before the next selection. Hidden evaluation labels enter neither the response source nor the query-risk calculation.

Decision relevance distinguishes this objective from resolving uncertainty alone. If two possible answers leave every accept, reject, and abstain decision unchanged, the question has zero expected decision value even when its answer is uncertain. Conversely, clarifying a feature or applicability condition can affect several candidate judgments. This is an anticipated benefit under the check and response models; the experiments separately measure whether it is realized. Appendix C specifies the coverage-audit provision and the optional preference interface; pairwise preference learning is not used in the reported experiments.

2.5 USING LEARNED CHECKS TO GUIDE REPAIR

The revision stage uses the frozen verifier to identify both an attempt to revise and the requirement its revision should address. The planner receives a diagnostic derived from violated checks and proposes a new reference motion, which the fixed controller executes before the next verification decision. In the walking example, insufficient net displacement supplies a specific reason to revise a motion that may otherwise be stable. Subsequent rounds can therefore change the reference while

preserving the learned interpretation of success. No further feedback updates the verifier during this stage, and cross-task motion caches are disabled.

The evaluation separates stopping decisions from the generation of revision candidates. The original comparison and a three-round learning replay apply alternative stopping rules to shared recorded chains. A further comparison generates a separate chain for each verifier snapshot, allowing its diagnostics to change the proposed motions. In both designs, the driver generates revisions beyond the states a scored policy would retain; stopping is applied afterward. Selected revision counts therefore describe the evaluated policy rather than the total work performed to construct the experimental chains.

The theoretical analysis asks how reliable a learned check must be before additional search helps. A flawed verifier can prefer an unsuccessful candidate to a successful one, so drawing more candidates creates more opportunities to select a misleading result. Because the check is learned once and reused, one learning error affects the whole search. Theorem 1 gives the maximum probability of that error for expected success not to decrease at any step up to a specified candidate budget.

The value of a clarification depends on the decisions its answer can change. Theorem 2 expresses this value as the reduction in posterior decoding error multiplied by the consequence of the shared condition across future tasks. A question can therefore resolve uncertainty yet have no decision value. These results assume fixed proposals, no false rejection, and a Bayesian decoder for the clarification analysis. They provide conditional insights into search and query selection, rather than an explanation of the observed false rejections. Full statements and proofs appear in Appendices A and B.

3 Experiments

We evaluate execution–task disagreement (A), feedback-driven classification (B), transfer (C), and revision allocation (D). Classification and transfer use fixed trajectory pools; repair experiments compare stopping policies on shared chains and verifier snapshots with separately generated chains. All outcomes are scored by an independent evaluator.

3.1 Platform, tasks, and evaluation

We use simulated Microduck motions to separate stable execution from task satisfaction. A fixed controller executes references composed from seventeen clips derived from nine released policies; the planner scales, mirrors, repeats, or concatenates them (Appendix D). The pool contains 49 tasks from 25 instruction templates: 10 development, 16 interaction, and 23 test tasks, covering direction and displacement, head motion, and posture or event sequences. Each task has a successful recorded candidate and three simulator seeds. Eleven additional requests never satisfied by a tested candidate assess abstention.

Feedback and final scoring have separate sources. A fixed language model supplies responses; one researcher authored the hidden task specifications. No human judgments or pairwise preferences enter the results. Confidence intervals resample whole instruction templates, keeping their tasks and rollouts together. A template defines a shared task pattern, such as turning in place with different angles. Seeds within a task are not independent learning runs.

3.2 Experiment A: Execution and task satisfaction

Stable execution frequently fails to establish task satisfaction, motivating checks beyond controller diagnostics. We compare a permissive planning prompt, which allows approximation, with a strict prompt requiring the requested motion or abstention. Both use the same 39 interaction and test tasks and three seeds per plan. Later experiments use the strict prompt.

Approximately half of the stably executed tasks fail their specifications under either prompt (Table 1). Execution is stable in 110/117 permissive and 98/99 strict rollouts, yet 18/36 and 15/32 stable tasks, respectively, remain unmet. We call this disagreement *silent failure*. It persists under both prompts; the paired task test detects no difference ($p = 1.00$), without establishing equiva-

Table 1: A1: execution versus task satisfaction on 39 locked tasks, three seeds each. The permissive prompt allows substituting a related motion when a request is out of reach; the strict prompt requires the requested motion or an explicit abstention. The two differ in no other respect.

	permissive	strict
rollouts executed stably	110/117 (94.0%)	98/99 (99.0%)
task success (all tasks)	18/39 (46.2%) [0.32, 0.61]	17/39 (43.6%) [0.29, 0.59]
task success (excluding abstentions)	18/39 (46.2%)	17/33 (51.5%) [0.35, 0.68]
abstentions	0	6 (15.4%)
silent failure (stable but unmet)	**18/36 (50.0%)**	**15/32 (46.9%)**

Table 2: B1: five conditions on 16 interaction tasks (9 templates) and the same 92 scored rollouts, matched at 1440 s of modeled response cost. Loss charges 1 for a wrong decision and $\lambda = 0.3$ for an abstention. The paired difference is against the no-questions verifier on the full panel, with a 95% confidence interval (CI) from a bootstrap over instruction templates. The post-hoc oracle reference is the best any hypothesis in the frozen candidate space achieves on each task's own rollouts; it never abstains and is not a generalizable ceiling.

	questions	coverage	loss (λ=0.3)	paired diff. vs. fixed [95% CI]	false rejections	selections /16
fixed verifier	0	0.66	0.177	—	7/44	5
direct labels	16	0.84	**0.125**	**−0.052 [−0.094, −0.017]**	7/44	**8**
random queries	21	0.84	0.147	−0.030 [−0.107, +0.038]	9/44	7
uncertainty queries	37	0.66	0.177	0.000 [0.000, 0.000]	7/44	5
proposed	16	**0.91**	0.157	−0.021 [−0.147, +0.102]	12/44	7
post-hoc oracle reference	—	1.00	0.076	—	—	11

lence. The strict planner also abstains on six reachable tasks and on none of the eleven requests never satisfied by a tested candidate.

A displacement example illustrates why stable tracking is insufficient: a recorded motion travels only 6–13 cm for a 30 cm request despite trunk uprightness above 0.995 (Figure 5, appendix).

3.3 Experiment B: Feedback and classification

Feedback increases the verifier's coverage but does not establish an advantage for the proposed query rule in classification loss. We compare fixed verification, direct labels, random queries, uncertainty queries, and value-based queries on the same recorded trajectories (Table 2). This holds candidate generation fixed while testing how feedback changes judgments. Direct labels memorize queried outcomes and retain the fixed verifier's decisions elsewhere.

The comparison uses a shared response source and a matched modeled budget. The panel contains 93 rollouts from 16 interaction tasks in nine templates: 44 pass, 48 fail, and one has no evaluator outcome. Each querying condition receives 1440 seconds of assigned response cost; responses are cached from a fixed language model and prompt. Following the diagnosis in Experiment C, checks apply only to their own tasks, a deviation from the registered aggregation rule. Coverage is the fraction receiving a pass/fail decision, and loss charges 1 for an incorrect decision and $\lambda = 0.3$ for abstention.

Only direct labeling has a loss improvement whose template-bootstrap interval excludes zero. The proposed rule changes loss by -0.021 relative to asking nothing, with a 95% interval from -0.147 to $+0.102$. It increases coverage from 0.66 to 0.91 while increasing false rejections from 7/44 to 12/44. No false acceptance is observed among 48 failing rollouts. The ordering depends on the abstention cost: the proposed rule is worst when abstaining is free and best when it costs as much as an error (Appendix F).

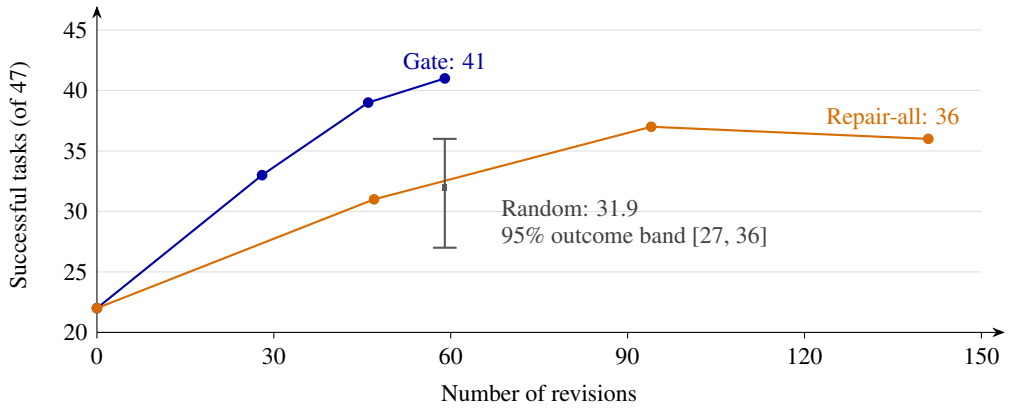


Figure 3: Revision allocation on 47 tasks. Points show the initial state and each of three measured rounds; connecting lines do not represent additional budget experiments. At 59 revisions, the random-policy marker shows the mean and central 95% outcome band over 10 000 draws on the same recorded chains, not a confidence interval for generalization.

The uncertainty baseline exposes a mismatch between its questions and the implemented decision rule. Its 37 questions leave every verdict unchanged. Under own-task aggregation, applicability questions cannot alter decisions by construction. This illustrates the need to consider decision relevance, but does not establish a general disadvantage of uncertainty-based active learning. The results cover one seed and one budget point; a decision-relevant uncertainty baseline remains untested.

3.4 Experiment C: Verifier transfer on held-out tasks

The learned checks do not improve classification on tasks excluded from their learning phase. We freeze checks learned on the ten development tasks and evaluate them on 431 recorded trajectories from the 39 interaction and test tasks (197 pass, 234 fail). Every query strategy produces the same verdicts as its no-learning baseline. This tests transfer within a fixed candidate pool; it does not bound repair with newly generated candidates.

Exploratory diagnostics associate broad check applicability with false rejection, but do not establish a transfer mechanism. Restricting scope reduces false rejection while increasing abstention. An oracle-answer replay is inconclusive because the likelihood effectively discards applicability answers; registered corrections also fail to recover transfer. The aggregation comparisons, tied oracle configurations, and correction outcomes are reported in Appendix G.

3.5 Experiment D: Verification-guided revision

Repair experiments test whether checks improve revision allocation and whether updating the checks improves outcomes. The original comparison applies stopping policies to shared recorded chains on 47 tasks, with at most three revisions per task. Two initially accepted interaction tasks lack complete chains and are excluded. Repair-all revises every task; no repair retains the initial attempt. A random policy spends the gate's 59 revisions uniformly among tasks with remaining capacity over 10 000 draws. Appendix G gives the full tables and exclusion audit.

The deployed verifier yields more successes with fewer selected revisions than repair-all (Figure 3). It reaches 41/47 successes with 59 revisions versus 36/47 with 141, a 58% reduction. Random allocation averages 31.9 successes, with a central 95% outcome band of $[27, 36]$; no draw reaches 41. This comparison is conditional on these chains and this random rule. The gate-minus-repair-all success difference is $+0.106$ (95% template-bootstrap CI $[+0.024, +0.196]$), while exact McNemar $p = 0.0625$. On a 37-task subset, a stability gate yields 19 successes versus 17 without repair and 34 with the learned gate.

Table 3: Learning comparison with separately generated revision chains. Entries show successful tasks and policy-selected revisions after three rounds. Both arms generated revisions for every task; the reported revision counts follow offline stopping decisions and are not total experimental expenditure.

		Before feedback		After feedback	
Split	Tasks	Successes	Revisions	Successes	Revisions
Interaction	16	8	24	13	28
Test	23	21	16	22	22
Pooled	39	29	40	35	50

Learning with separately generated chains. Updated checks improve success when each verifier supplies diagnostics for its own revisions (Table 3). The two snapshots share initial records, controller, planner configuration, three simulator seeds, and a three-round cap on 39 tasks from 20 templates. Each generates new motions from its own diagnostics. All task–round revisions are executed; stopping is evaluated afterward. Thus, selected revision counts measure the scored policy rather than total experimental execution.

The updated verifier achieves 35/39 successes versus 29/39 before feedback, with seven successes unique to the updated arm and one unique to the prior arm. The difference is $+0.154$ (95% template-bootstrap CI $[+0.024, +0.308]$); exact McNemar $p = 0.0703$. Five additional successes occur on interaction tasks and one on test tasks. The update selects 50 rather than 40 revisions: $+0.26$ per task, CI $[-0.03, +0.61]$. This comparison uses a common round cap, not equal expenditure. Two invalid planner responses in the prior arm required retries under a rule implemented after the failures; one retry affects the scored outcome (Appendix G).

A shared-chain replay also favors updated selection while holding revisions fixed. On 37 tasks, the update yields 34 successes with 49 revisions versus 30 with 41 before feedback. The success difference is $+0.108$ (95% template-bootstrap CI $[0.000, +0.250]$), with exact McNemar $p = 0.125$ (Table 11, appendix). Together, the comparisons distinguish allocation by the deployed verifier, changes in diagnostics and generated repairs, and selection changes alone. Revision counts exclude feedback, verification, and learning costs.

4 Conclusion

We proposed LVR, which learns executable task checks from feedback and uses them to guide subsequent repair. The analysis relates check reliability to the value of additional candidate search. In the recorded-chain comparison, the deployed verifier improves task success while selecting fewer revisions than repair-all. With separately generated chains, feedback-updated checks improve success over the pre-feedback verifier while selecting more revisions; a shared-chain replay also favors updated selection. The gains are concentrated on interaction tasks, while held-out classification gains remain unestablished. These results support evaluating verification learning through both predictive accuracy and the decisions made with the learned checks.

Evidence is limited to one simulated robot and one pair of verifier snapshots, with model feedback and offline stopping. Generalization to new templates, human feedback, and hardware remains untested; Appendix L details these limits.

AI Use Statement

An AI assistant supported literature retrieval, proof drafting, and manuscript editing.

REFERENCES

Erdem Bıyık and Dorsa Sadigh. Batch active preference-based learning of reward functions. In *Conference on Robot Learning (CoRL)*, 2018.

Erdem Bıyık, Malayandi Palan, Nicholas C. Landolfi, Dylan P. Losey, and Dorsa Sadigh. Asking easy questions: A user-friendly approach to active reward learning. In *Conference on Robot Learning (CoRL)*, 2019.

Erdem Bıyık, Dylan P. Losey, Malayandi Palan, Nicholas C. Landolfi, Gleb Shevchuk, and Dorsa Sadigh. Learning reward functions from diverse sources of human feedback: Optimally integrating demonstrations and preferences. *The International Journal of Robotics Research*, 41(1): 45–67, 2022. doi: 10.1177/02783649211041652.

Andreea Bobu, Andrea Bajcsy, Jaime F. Fisac, Sampada Deglurkar, and Anca D. Dragan. Quantifying hypothesis space misspecification in learning from human–robot demonstrations and physical corrections. *IEEE Transactions on Robotics*, 36(3):835–854, 2020. doi: 10.1109/TRO.2020.2971415.

Andreea Bobu, Marius Wiggert, Claire Tomlin, and Anca D. Dragan. Feature expansive reward learning: Rethinking human input. In *ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pp. 216–224, 2021. doi: 10.1145/3434073.3444667.

Andreea Bobu, Chris Paxton, Wei Yang, Balakumar Sundaralingam, Yu-Wei Chao, Maya Cakmak, and Dieter Fox. Learning perceptual concepts by bootstrapping from human queries. *IEEE Robotics and Automation Letters*, 7(4):11260–11267, 2022a. doi: 10.1109/LRA.2022.3196164.

Andreea Bobu, Marius Wiggert, Claire Tomlin, and Anca D. Dragan. Inducing structure in reward learning by learning features. *The International Journal of Robotics Research*, 41(5):497–518, 2022b. doi: 10.1177/02783649221078031.

Maya Cakmak and Andrea L. Thomaz. Designing robot learners that ask good questions. In *ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pp. 17–24, 2012. doi: 10.1145/2157689.2157693.

Xuxin Cheng, Yandong Ji, Junming Chen, Ruihan Yang, Ge Yang, and Xiaolong Wang. Expressive whole-body control for humanoid robots. In *Robotics: Science and Systems (RSS)*, 2024. doi: 10.15607/RSS.2024.XX.107.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

Yuchen Cui, Siddharth Karamcheti, Raj Palleti, Nidhya Shivakumar, Percy Liang, and Dorsa Sadigh. "no, to the right": Online language corrections for robotic manipulation via shared autonomy. In *ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pp. 93–101, 2023. doi: 10.1145/3568162.3578623.

Florian E. Dorner, Yatong Chen, André F. Cruz, and Fanny Yang. ROC-n-reroll: How verifier imperfection affects test-time scaling. In *International Conference on Learning Representations (ICLR)*, 2026. URL https://openreview.net/forum?id=3Gy5mmyuxn.

Leo Gao, John Schulman, and Jacob Hilton. Scaling laws for reward model overoptimization. In *International Conference on Machine Learning (ICML)*, volume 202 of *Proceedings of Machine Learning Research*, pp. 10835–10866. PMLR, 2023.

Minae Kwon, Sang Michael Xie, Kalesha Bullard, and Dorsa Sadigh. Reward design with language models. In *International Conference on Learning Representations (ICLR)*, 2023.

Lauro Langosco, Jack Koch, Lee Sharkey, Jacob Pfau, Laurent Orseau, and David Krueger. Goal misgeneralization in deep reinforcement learning. In *International Conference on Machine Learning (ICML)*, 2022.

Manling Li, Shiyu Zhao, Qineng Wang, Kangrui Wang, Yu Zhou, Sanjana Srivastava, Cem Gokmen, Tony Lee, Li Erran Li, Ruohan Zhang, Weiyu Liu, Percy Liang, Li Fei-Fei, Jiayuan Mao, and Jiajun Wu. Embodied agent interface: Benchmarking LLMs for embodied decision making. In *Advances in Neural Information Processing Systems (NeurIPS) Datasets and Benchmarks Track*, 2024a.

Xuanlin Li, Kyle Hsu, Jiayuan Gu, Oier Mees, Karl Pertsch, Homer Rich Walke, Chuyuan Fu, Ishikaa Lunawat, Isabel Sieh, Sean Kirmani, Sergey Levine, Jiajun Wu, Chelsea Finn, Hao Su, Quan Vuong, and Ted Xiao. Evaluating real-world robot manipulation policies in simulation. In *Conference on Robot Learning (CoRL)*, volume 270 of *Proceedings of Machine Learning Research*, pp. 3705–3728. PMLR, 2024b.

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. In *International Conference on Learning Representations (ICLR)*, 2024.

Zhengyi Luo, Jinkun Cao, Alexander Winkler, Kris Kitani, and Weipeng Xu. Perpetual humanoid control for real-time simulated avatars. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023.

Zhengyi Luo, Jinkun Cao, Josh Merel, Alexander Winkler, Jing Huang, Kris Kitani, and Weipeng Xu. Universal humanoid motion representations for physics-based control. In *International Conference on Learning Representations (ICLR)*, 2024.

Yecheng Jason Ma, William Liang, Guanzhi Wang, De-An Huang, Osbert Bastani, Dinesh Jayaraman, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Eureka: Human-level reward design via coding large language models. In *International Conference on Learning Representations (ICLR)*, 2024.

Helena Merker, Nick Walker, and Andreea Bobu. Robots that know what to ask: Recovering misaligned rewards through targeted explanations. *arXiv preprint arXiv:2605.22986*, 2026. URL `https://arxiv.org/abs/2605.22986`.

Vivek Myers, Erdem Bıyık, and Dorsa Sadigh. Active reward learning from online preferences. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2023.

Malayandi Palan, Gleb Shevchuk, Nicholas Charles Landolfi, and Dorsa Sadigh. Learning reward functions by integrating human demonstrations and preferences. In *Robotics: Science and Systems (RSS)*, 2019. doi: 10.15607/RSS.2019.XV.023.

Alexander Pan, Kush Bhatia, and Jacob Steinhardt. The effects of reward misspecification: Mapping and mitigating misaligned models. In *International Conference on Learning Representations (ICLR)*, 2022.

Xue Bin Peng, Pieter Abbeel, Sergey Levine, and Michiel van de Panne. Deepmimic: Example-guided deep reinforcement learning of physics-based character skills. *ACM Transactions on Graphics*, 37(4):1–14, 2018. doi: 10.1145/3197517.3201311.

Xue Bin Peng, Ze Ma, Pieter Abbeel, Sergey Levine, and Angjoo Kanazawa. AMP: Adversarial motion priors for stylized physics-based character control. *ACM Transactions on Graphics*, 40 (4):1–20, 2021. doi: 10.1145/3450626.3459670.

Xue Bin Peng, Yunrong Guo, Lina Halper, Sergey Levine, and Sanja Fidler. ASE: Large-scale reusable adversarial skill embeddings for physically simulated characters. *ACM Transactions on Graphics*, 41(4):1–17, 2022. doi: 10.1145/3528223.3530110.

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 *Conference on Robot Learning (CoRL)*, 2023.

Dorsa Sadigh, Anca D. Dragan, Shankar Sastry, and Sanjit A. Seshia. Active preference-based learning of reward functions. In *Robotics: Science and Systems (RSS)*, 2017. doi: 10.15607/RSS.2017.XIII.053.

Joar Skalse, Nikolaus H. R. Howe, Dmitrii Krasheninnikov, and David Krueger. Defining and characterizing reward gaming. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

Benedikt Stroebl, Sayash Kapoor, and Arvind Narayanan. The limits of inference scaling through resampling. *arXiv preprint arXiv:2411.17501*, 2024.

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

Akifumi Wachi, Takumi Tanabe, and Youhei Akimoto. Safety hacking in constrained best-of-N inference-time scaling. *arXiv preprint arXiv:2608.22915*, 2026. URL https://arxiv.org/abs/2608.22915. Preprint, no peer-reviewed venue.

Peiyi Wang, Lei Li, Zhihong Shao, Runxin Xu, Damai Dai, Yifei Li, Deli Chen, Yu Wu, and Zhifang Sui. Math-shepherd: Verify and reinforce LLMs step-by-step without human annotations. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 9426–9439, 2024. doi: 10.18653/v1/2024.acl-long.510.

Tianhao Wu, Weizhe Yuan, Olga Golovneva, Jing Xu, Yuandong Tian, Jiantao Jiao, Jason Weston, and Sainbayar Sukhbaatar. Meta-rewarding language models: Self-improving alignment with LLM-as-a-meta-judge. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 11537–11554, 2025. doi: 10.18653/v1/2025.emnlp-main.583.

Tianbao Xie, Siheng Zhao, Chen Henry Wu, Yitao Liu, Qian Luo, Victor Zhong, Yanchao Yang, and Tao Yu. Text2reward: Reward shaping with language models for reinforcement learning. In *International Conference on Learning Representations (ICLR)*, 2024.

Wenhao Yu, Nimrod Gileadi, Chuyuan Fu, Sean Kirmani, Kuang-Huei Lee, Montserrat Gonzalez Arenas, Hao-Tien Lewis Chiang, Tom Erez, Leonard Hasenclever, Jan Humplik, Brian Ichter, Ted Xiao, Peng Xu, Andy Zeng, Tingnan Zhang, Nicolas Heess, Dorsa Sadigh, Jie Tan, Yuval Tassa, and Fei Xia. Language to rewards for robotic skill synthesis. In *Conference on Robot Learning (CoRL)*, volume 229 of *Proceedings of Machine Learning Research*, pp. 374–404. PMLR, 2023.

A Verification Quality and Subsequent Search

We analyze two questions under a simplified model: how reliable a learned check must be before more search helps, and what a question about a shared condition is worth. Prior theory characterizes error amplification under a *fixed* verifier (Wachi et al., 2026; Dorner et al., 2026); the difference here is that the check is learned once, before search, and its correctness is shared across the search horizon. Proofs and the supporting constructions are in Appendix B.

Model. Candidates are drawn independently. A candidate succeeds with probability $g > 0$; with probability $b > 0$ it is a false acceptance, which the flawed verifier ranks above every success; the rest fail, and $g + b < 1$. A correct check removes false acceptances without rejecting successes or changing proposals, so after n draws the success probabilities are

$$F_n = (1-b)^n - (1-b-g)^n, \qquad C_n = 1-(1-g)^n. \tag{4}$$

If feedback learns the correct check with probability $1-\eta$, expected success is $S_n(\eta) = (1-\eta)C_n + \eta F_n$. Here η is uncertainty about one shared rule, not an independent error per candidate.

Theorem 1 (Confidence required for a search horizon)**.** *Let $N \geq 2$, $a_n = g(1-g)^n$, and $d_n = (b+g)(1-b-g)^n - b(1-b)^n$. Then $S_1(\eta), \ldots, S_N(\eta)$ is nondecreasing if and only if*

$$\eta \leq \eta_N^* := \min_{1 \leq n < N} \frac{a_n}{a_n + [-d_n]_+}, \qquad [z]_+ = \max\{z, 0\}. \tag{5}$$

When $b < g$, $\eta_N^* \sim (g/b)[(1-g)/(1-b)]^{N-1}$ as $N \to \infty$.

The threshold tightens with the horizon because rare false acceptances eventually dominate selection, so demanding that *every* search increment help is a strong requirement: for fixed $0 < b < g$, $g + b < 1$, and independent answers correct with fixed probability $1/2 + \gamma$, majority decoding needs $\Theta(N)$ answers (Corollary 1). Accepting a fixed additive tolerance ε instead requires only $\eta \leq \varepsilon$, independently of N. These are different guarantees for the specified response model.

Reuse across tasks. Suppose future tasks carry weights μ_x and each depends on one shared binary condition $j(x)$, decoded by a maximum a posteriori rule with error $\eta_j = \min\{p_j, 1-p_j\}$, where p_j is the Bayesian posterior probability of condition j. Decoding uses the available feedback before the future candidate pools are seen. Writing $W_j(N) = \sum_{x:j(x)=j} \mu_x [C_x(N) - F_x(N)]$ for the weight of condition j at budget N:

Theorem 2 (Value of a reusable clarification)**.** *If query q concerns condition j, changes no other condition posterior, and leaves proposals and conditional success functions fixed, its expected gain is*

$$V_N(q) = W_j(N)\,[\eta_j - \mathbb{E}_Y \eta_j(Y)] \geq 0, \tag{6}$$

where Y is its response and $\eta_j(Y)$ the updated decoding error.

Equation (6) is a value-of-information score (Myers et al., 2023) in which uncertainty reduction is weighted by $W_j(N)$, the effect that condition has on later decisions. A question has zero expected gain when $W_j(N) = 0$, even if the condition is maximally uncertain. Appendix B.5 constructs such a case, and Section 3.3 contains a measured one, where an uncertainty-driven baseline spent its entire budget on questions of exactly this kind. The theorem does not establish a greedy guarantee. The model assumes fixed proposals, no false rejection, and non-interacting conditions. The analysis concerns amplification of false *acceptance* during search. The classification experiments observed no false acceptance; their errors were false rejections. The analysis therefore describes a regime our experiments did not enter, and we do not offer it as an explanation of the empirical results.

B Theoretical Analysis and Proofs

B.1 Candidate model and shared learning uncertainty

For a fixed task, independent candidate draws have three mutually exclusive types: genuine successes G of mass $g > 0$, unsuccessful accepted candidates B of mass $b > 0$, and other failures of

mass $1 - g - b > 0$. Every B outranks every G under the flawed verifier; if neither appears, selection or abstention is a failure. A correct check rejects all B and preserves all G, with unchanged proposals. The flawed selector succeeds exactly when no B and at least one G appear. Therefore

$$F_n = \Pr(\text{no } B) - \Pr(\text{no } B \text{ and no } G) = (1-b)^n - (1-b-g)^n, \tag{7}$$

$$C_n = \Pr(\text{at least one } G) = 1 - (1-g)^n. \tag{8}$$

This recovers Equation (4); the event calculation is an attributed special case of Wachi et al. (2026), equation (80). The correction opportunity is

$$D_n = C_n - F_n = 1 - (1-g)^n - (1-b)^n + (1-b-g)^n. \tag{9}$$

It equals the probability of observing both G and B. Coupling pools as prefixes of one candidate stream shows that D_n is nonnegative and nondecreasing. Meanwhile, $F_n \to 0$ and $C_n \to 1$.

The learned check is correct with probability $1 - \eta$. Conditional on correct learning, success is C_n; conditional on incorrect learning, it is F_n. These effects are model assumptions. Future candidate draws are independent of learning error conditional on the task. Hence $S_n(\eta) = (1-\eta)C_n + \eta F_n$ averages over both candidate draws and the learned check. A check is learned once per run. Substituting ηb for b in F_n would incorrectly replace shared learning uncertainty with independent candidate-level errors.

B.2 Proof of Theorem 1

Proof. Direct subtraction gives $C_{n+1} - C_n = a_n$ and $F_{n+1} - F_n = d_n$. Thus

$$S_{n+1}(\eta) - S_n(\eta) = (1-\eta)a_n + \eta d_n. \tag{10}$$

Because $a_n > 0$, this is nonnegative for every $\eta \in [0,1]$ if $d_n \geq 0$. If $d_n < 0$, it is nonnegative exactly when $\eta \leq a_n/(a_n - d_n)$. Taking the intersection over $1 \leq n < N$ proves Equation (5).

For $b < g$, define $r = (1-b)/(1-g) > 1$ and $s = (1-b-g)/(1-g) \in (0,1)$. Then

$$\frac{-d_n}{a_n} = \frac{b}{g} r^n - \frac{b+g}{g} s^n. \tag{11}$$

The first term increases strictly and the subtracted positive term decreases strictly. Its positive part is therefore nondecreasing in n. The most restrictive increment is $n = N - 1$, giving

$$\eta_N^* = \left(1 + \left[\frac{b}{g} r^{N-1} - \frac{b+g}{g} s^{N-1}\right]_+\right)^{-1}. \tag{12}$$

Since $r > 1$ and $s < 1$, the first term dominates asymptotically. This yields $\eta_N^* \sim (g/b) r^{-(N-1)}$. □

The endpoint simplification requires $b < g$. For $g = 0.2, b = 0.4$, the values of $[-d_n]_+/a_n$ at $n = 1, 2, 3, 4$ are 0, 0.375, 0.46875, and 0.4453125. Checking only the last increment would miss a stricter earlier constraint. Equation (5) applies in either regime.

B.3 Feedback bounds for a specified response channel

Suppose a binary parameter θ determines the check. Independently, each response reports θ correctly with probability $1/2 + \gamma$ conditional on either value of θ, where $0 < \gamma < 1/2$. An odd number m of answers is decoded by majority vote before search. Correct decoding gives C_n, and incorrect decoding gives F_n, with the same g, b under either parameter value. Candidate observations do not modify this estimator.

Corollary 1 (Majority feedback and search effort)**.** *Writing η_m for majority error, we have*

$$(1/2 - \gamma)^m \leq \eta_m \leq e^{-2\gamma^2 m}. \tag{13}$$

For $\eta_N^* < 1$, a sufficient condition for $S_1(\eta_m), \ldots, S_N(\eta_m)$ to be nondecreasing is

$$m \geq \frac{\log(1/\eta_N^*)}{2\gamma^2}, \tag{14}$$

with odd-integer rounding. A necessary condition for this majority procedure is

$$m \geq \frac{\log(1/\eta_N^*)}{\log(1/(1/2-\gamma))}. \tag{15}$$

For fixed $0 < b < g$, $g + b < 1$, and $0 < \gamma < 1/2$, the smallest qualifying odd m is $\Theta(N)$ as $N \to \infty$.

Proof. All m answers being wrong forces a wrong majority and occurs with probability $(1/2-\gamma)^m$. For the upper bound, let K count correct answers and $p = 1/2 + \gamma$. For $t > 0$, Markov's inequality gives

$$\Pr(K \leq m/2) \leq e^{tm/2}(1 - p + pe^{-t})^m.$$

Taking $t = \log(p/(1-p))$ makes this $[2\sqrt{p(1-p)}]^m = (1-4\gamma^2)^{m/2} \leq e^{-2\gamma^2 m}$. The majority-error event is contained in $\{K \leq m/2\}$. Combining the lower and upper bounds with Theorem 1 proves Equations (14) and (15). Equation (12) implies $\log(1/\eta_N^*) = (N-1)\log r + \log(b/g) + o(1)$ for $b < g$. Both bounds are proportional to N for fixed parameters, and rounding to an odd integer leaves this order unchanged. □

This is a bound for one learning procedure and a particular criterion, not a lower bound for all feedback strategies. Monotonicity concerns expectation over learned-check realizations and includes increments with very small absolute size. In contrast, ensuring a correct learned check with probability $1 - \delta$ requires only $m \geq \log(1/\delta)/(2\gamma^2)$. Also, $C_n - S_n(\eta) = \eta D_n \leq \eta$, so a fixed additive tolerance ε requires only $\eta \leq \varepsilon$. Neither condition depends on the search horizon. Independent binary responses are an analytical channel; repeated judgments by one provider or natural-language clarifications need not satisfy its assumptions.

B.4 Proof of Theorem 2

Let $\mathcal{H}$ be the interaction history, $p_j = \Pr(\theta_j = 1 \mid \mathcal{H})$, and $\eta_j = \min(p_j, 1 - p_j)$. A decoder fixed from this history maximizes success by choosing a maximum a posteriori value because $C_x(N) \geq F_x(N)$, with these values depending on decoding correctness alone. This optimality is restricted to such decoders; it does not include inference from future candidate pools.

Proof. Before querying, the expected success averaged over future tasks is

$$\sum_x \mu_x\{F_x(N) + [C_x(N) - F_x(N)](1 - \eta_{j(x)})\} = \sum_x \mu_x F_x(N) + \sum_k W_k(N)(1 - \eta_k). \tag{16}$$

After observing Y, only the jth term changes in expectation, so the improvement is $W_j(N)[\eta_j - \mathbb{E}_Y \eta_j(Y)]$. Posterior probabilities satisfy $\mathbb{E}_Y p_j(Y) = p_j$, and $u \mapsto \min(u, 1-u)$ is concave. Jensen's inequality therefore gives $\mathbb{E}_Y \eta_j(Y) \leq \eta_j$. Since $W_j(N) \geq 0$, the gain is nonnegative. □

For one equal-cost query, maximizing its exact value is optimal within this model. A gain-per-cost objective can divide by a positive cost; this does not prove greedy optimality for a budgeted sequence. If a query informs several correlated conditions, the value is the sum of their weighted error reductions, provided the separable task model still holds. Tasks involving interacting unresolved conditions can violate that model.

For a symmetric binary channel with correctness probability $a > 1/2$ conditional on either parameter value, the expected posterior error is

$$\mathbb{E}_Y \eta_j(Y) = \min\{p_j a, (1-p_j)(1-a)\} + \min\{p_j(1-a), (1-p_j)a\}. \tag{17}$$

Each term is the smaller of the two joint state-response probabilities for a possible answer. A query can carry information but have zero one-step decision value when neither answer changes the preferred decoder. Thus a sequence of questions can have value not captured by the first question's immediate gain.

B.5 CLASSIFICATION ACCURACY AND SELECTION VALUE

Proposition 1 (Bayesian query separation)**.** *There is an equal-cost two-query problem in which the unique query maximizing expected classification-error reduction has zero expected selection benefit, while the other eliminates the selection error.*

Proof. Let θ, ϕ be independent fair bits. Candidate types A_0, A_1, C, G have acceptance labels $1-\theta, \theta, \phi, 1$. A classification distribution puts total mass r on A_0, A_1, mass c on C, and the remainder on G, where $0 < r < c$ and $r + c \leq 1$. Equal-cost noiseless queries reveal either θ or ϕ. Before feedback, Bayes classification error is $(r + c)/2$. Querying θ reduces it by $r/2$, while querying ϕ reduces it by $c/2$; hence accuracy strictly favors the latter.

For selection, the candidate pool is $\{A_0, A_1\}$ with probability $h \in (0, 1]$ and $\{G, C\}$ otherwise, independently of the bits. The agent maximizes posterior acceptance probability. Without feedback, it succeeds with probability $1/2$ in the first pool and one in the second, giving $1 - h/2$. Revealing θ identifies the valid candidate in the critical pool and raises success to one. Revealing ϕ leaves success at $1 - h/2$, since G is already valid in the other pool. The same θ may govern multiple future tasks, making its answer reusable. □

This construction uses structured pools, distinct from the independent-draw model. It illustrates the decision-value principle underlying active reward learning (Myers et al., 2023), rather than asserting a new general principle.

B.6 A REPAIR COMPARISON UNDER ONE PROPOSAL DISTRIBUTION

Add a successful class C of mass $c > b$ that the verifier falsely rejects; other failures have mass $1 - g - b - c \geq 0$. Initially only B and G are accepted, with B ranked higher. Two equal-cost perfect interventions either reject B or accept C below G. Classification accuracy improves by b or c, respectively, so it favors correcting the false rejection. The task-success gains after n independent draws are

$$\Delta_B(n) = 1 - (1-g)^n - (1-b)^n + (1-b-g)^n, \tag{18}$$

$$\Delta_C(n) = (1-b-g)^n - (1-b-g-c)^n. \tag{19}$$

The first event is the co-occurrence of B and G; the second is the occurrence of C with neither B nor G. These event descriptions prove the formulas. At $n = 1$, the gains are zero and c. As $n \to \infty$, they converge to one and zero. The preferred repair therefore changes with search effort despite the unchanged classification-accuracy ordering. This compares specified repairs, not optimal Bayesian querying.

B.7 RESIDUAL ERRORS AND ADAPTIVE PROPOSALS

For a fixed realized check, let residual harmful mass be $e > 0$ and retained successful mass be $g > 0$, with $e + g < 1$. If residual harmful candidates outrank every success, then $\widetilde{S}_n = (1-e)^n - (1-e-g)^n$. Rearranging its marginal improvement gives

$$\widetilde{S}_{n+1} \geq \widetilde{S}_n \quad \Longleftrightarrow \quad n \leq \frac{\log(e/(e+g))}{\log((1-e-g)/(1-e))}. \tag{20}$$

The logarithm in the denominator is negative, which reverses the inequality when dividing. Every fixed $e > 0$ eventually gives $\widetilde{S}_n \to 0$. False rejection must be reflected in the retained good mass rather than assumed absent.

A weaker bound allows adaptive proposals. For every reachable history before draw t, suppose the conditional probability of a retained success is at least a deterministic $\alpha_t \in [0, 1]$, and that of an accepted failure is at most $\beta_t \in [0, 1]$. Assume the selector succeeds if some retained success and no accepted failure occur. Repeated conditioning gives $\Pr(\text{no accepted failure}) \geq \prod_t (1 - \beta_t)$ and $\Pr(\text{no retained success}) \leq \prod_t (1 - \alpha_t)$. Subtraction yields

$$\Pr(\text{success after } N) \geq \max\left\{0, \prod_{t=1}^{N}(1-\beta_t) - \prod_{t=1}^{N}(1-\alpha_t)\right\}. \tag{21}$$

Table 4: LVR stages and their implementation. Evaluation labels are used only for offline scoring. The low-level controller is fixed throughout.

Stage	Input and operation	Output retained
Construct checks	Parse instruction slots; enumerate feature, threshold, scope, and secondary constraints.	Typed configurations and initial beliefs.
Acquire feedback	Score questions by expected decision-risk reduction per modeled cost.	Question, response, and updated beliefs.
Verify execution	Evaluate trajectory features and aggregate applicable checks.	Decision and violated criteria.
Allocate repair	Use the frozen verifier to select attempts and provide diagnostics to the planner.	Revised references and stopping decisions.

This elementary bound need not increase with N. Marginal validation errors do not establish its history-uniform assumptions. The exact curves above therefore describe fixed-proposal analysis, while the adaptive search experiments measure effects without assuming those curves hold.

C Verifier and Query Specification

This appendix specifies the check representation and query objective used by the framework, followed by an optional preference component and the intended search interface. Experimental departures from that interface are stated explicitly: the reported feedback is automated, and the preference component was not exercised.

C.1 Typed checks and evidence

A check stores an applicability condition, a trajectory feature, a predicate or threshold, its feedback source, and a version identifier. Candidate features include signed displacement, heading change, head orientation, and ordered events, subject to validation of available simulator observations. For example, a directional clarification may determine whether a displacement threshold applies in the initial robot frame or a fixed world frame. Reusing that interpretation across parameterized tasks is distinct from replaying a corrected motion.

The initial method uses a predefined vocabulary of feature and predicate templates. It learns which checks apply and estimates their parameters within this vocabulary. It does not claim to discover arbitrary features or synthesize unrestricted verification programs. An instruction parser proposes applicable checks using only the observed instruction; its model, prompt, and access budget are held constant across comparisons.

Acceptance is conservative with respect to the checks represented by the verifier: a supported violation yields fail; satisfied represented requirements yield pass only if the coverage mechanism reports no unresolved requirement; otherwise the output is unknown. This rule does not guarantee complete semantic coverage. A requirement omitted by both the parser and the hypothesis space can still cause false acceptance, which the independent evaluator must measure.

C.2 Clarification and preference feedback

We consider two query types. A targeted clarification asks for a task condition, such as a coordinate frame, an ordering constraint, or an acceptable duration. A pairwise query presents two trajectories and permits A, B, same, or neither. Clarifications can update acceptance checks when their scope is explicit. Pairwise responses update ranking preferences; neither triggers a diagnostic follow-up rather than automatically labeling both trajectories as failures.

For strict pairwise preferences, a simple initial model is

$$P(a \succ b \mid x, w) = \sigma\big(w^\top [\phi(x, \tau_a) - \phi(x, \tau_b)]\big)\,, \tag{22}$$

where ϕ contains validated trajectory features and σ is the logistic function. Preference weights are separate from acceptance thresholds. Same and neither are recorded as diagnostic responses and do not enter this likelihood. No pairwise responses were collected in the experiments reported here, so this component is specified but untested. Feedback provenance is retained so that an update can be audited or withdrawn without silently rewriting earlier versions.

C.3 Query value across development tasks

Section 2 gives the implemented classification objective, averaged over task–trajectory pairs with accept, reject, and abstain decisions. The formulation below describes the broader candidate-selection interface, including an optional preference term that was not used in the reported experiments. These objectives should not be conflated when reproducing the query scores. Let θ index a finite hypothesis set of check configurations and preference parameters, with belief $p_t(\theta)$ after t responses. For each development task x, let $\mathcal{D}_x$ contain candidate-selection and abstention decisions. A model-defined loss $\ell_\theta(d, x)$ penalizes acceptance of a trajectory that fails the hypothesis's checks, permits abstention at a specified cost, and optionally incorporates preference among accepted candidates. This loss is a query-selection surrogate, not the independent task evaluator. Define

$$\mathcal{L}(p) = \frac{1}{|\mathcal{X}_{\mathrm{dev}}|} \sum_{x \in \mathcal{X}_{\mathrm{dev}}} \min_{d \in \mathcal{D}_x} \mathbb{E}_{\theta \sim p}[\ell_\theta(d, x)]. \tag{23}$$

For a candidate query q and response y, the proposed score is

$$U_t(q) = \frac{\mathcal{L}(p_t) - \mathbb{E}_{y \sim p_t(y|q)}[\mathcal{L}(p_t(\theta \mid q, y))]}{\widehat{c}_t(q) + \epsilon}, \qquad \epsilon > 0, \tag{24}$$

where $\widehat{c}_t(q)$ is the assigned response cost for the question type. Query responses are enumerated within the allowed response space, and hypothetical posterior updates estimate their consequences across the development pool. In contrast to maximizing response entropy alone, this score values uncertainty resolution according to its expected effect on downstream decisions. Its usefulness depends on the hypothesis space and the response model; Section 3.3 measures it and does not find an advantage over asking nothing.

Agreement among hypotheses cannot detect a condition omitted by every hypothesis. We therefore reserve a prespecified fraction of the interaction budget for coverage-audit queries and report their cost separately. The uncertainty baseline receives the same audit mechanism. Reported interaction costs are assigned response costs, not measured human response times; an audit-removal ablation is not reported.

C.4 Search with a frozen verifier

At each search round, the generator proposes a bounded number of references using the instruction and permitted prior search state. All candidates are executed before trajectory-based verification. The optimizer first considers accepted candidates and then ranks them by preference. When none passes, it records an abstention and may revise candidates within the remaining compute budget using verifier diagnostics. Unknown candidates are not counted as successful merely because they receive a high preference score.

After interaction ends, the verifier version is fixed and no further human answers are supplied. The search state can evolve within a task, but cross-task motion caches are disabled in the principal comparison. This protocol tests whether previously acquired checks support new search rather than merely retrieving an earlier correction.

D The Robot, the Motion Library, and the Method Conditions

The platform combines a fixed tracking controller with a restricted reference-motion library. The figures below illustrate the available behaviors, and Table 5 summarizes the feedback conditions used to update task checks.

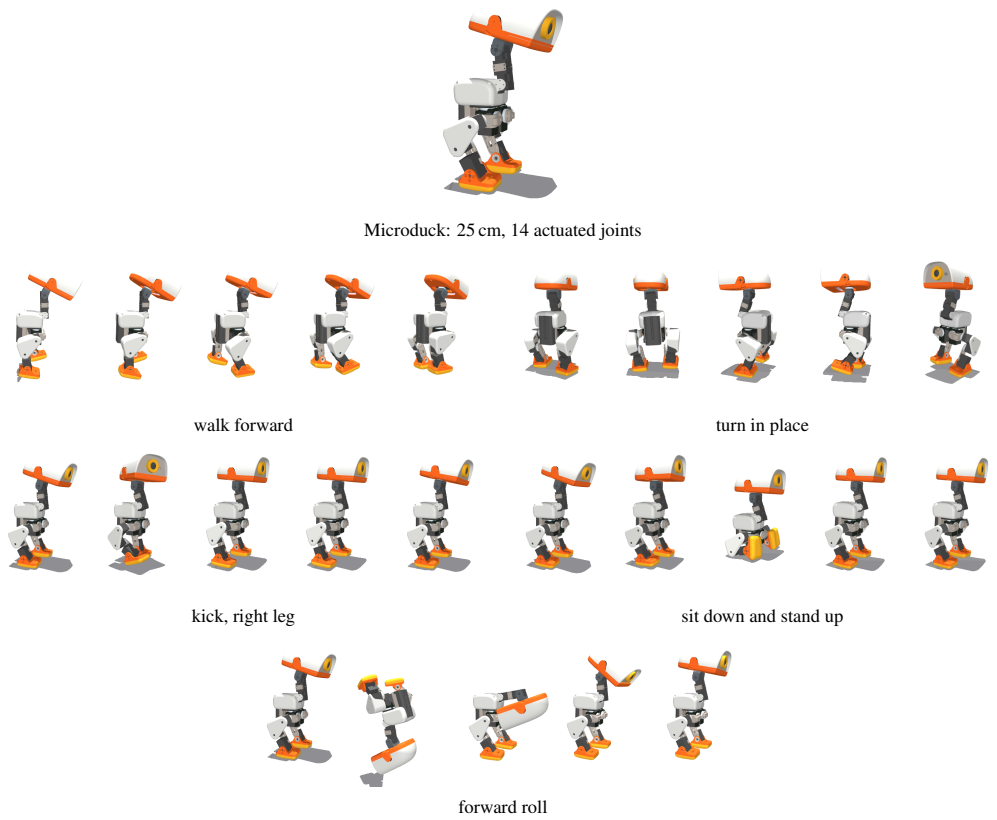


Figure 4: The robot and five of the seventeen base clips the planner composes from. Frames are chosen at the moment that defines each skill rather than at even time intervals, because the event that distinguishes a kick from standing occupies a small fraction of the clip. Camera and scale are identical across rows. The five differ in kind: a gait, a rotation, a single-leg event, a posture change, and a whole-body roll. The task families are built along these differences, and an instruction outside their span cannot be satisfied by composition alone.

Method	Retained feedback	Query rule
Fixed verifier	None	No queries
Direct labels	Memorized verdicts	Query the most uncertain rollout
Random verifier updates	Structured checks	Random eligible query
Uncertainty updates	Structured checks	Response uncertainty
Proposed	Structured checks	Development decision value per modeled cost

Table 5: Feedback conditions compared in Experiment B on a shared trajectory panel. Querying conditions have matched modeled response budgets; the fixed verifier uses no feedback.

E Additional Related Work

Representation misspecification and evaluation design provide further context for the execution–satisfaction distinction. Studies of reward misspecification, goal misgeneralization, and reward gaming examine how optimized behavior can diverge from an intended objective (Pan et al., 2022; Langosco et al., 2022; Skalse et al., 2022). Our experiments examine a specific instance: stable execution of a reference that does not meet the task specification.

Embodied benchmarks also motivate separating the components of task performance. Embodied Agent Interface evaluates stages of embodied decision making (Li et al., 2024a), while simulation-based evaluation studies how measured robot performance relates to deployment (Li et al., 2024b).

Our independent evaluator checks task conditions after execution, separately from both the planner and the learned verifier.

Question design complements the choice of which uncertainty to resolve. Work on robot questions examines the information people can supply (Cakmak & Thomaz, 2012), and demonstrations can be combined with preference comparisons (Palan et al., 2019). These findings motivate multiple feedback formats, but the present experiments use automated clarification responses. Their results therefore do not establish human response quality, effort, or preference-learning performance.

F Interaction Cost, Uncertainty, and Ablations

This appendix reports the diagnostic comparisons, uncertainty estimates, and protocol deviations underlying Experiments A–D. The analyses separate observed changes from post-hoc interpretations and from predictions registered before a follow-up test.

Execution accounting for separately generated chains. After retries, the archive contains 234 valid task–round revisions: two arms, 39 tasks, and three rounds, each evaluated with three simulator seeds. Two initial invalid-JSON responses and two omitted downstream revisions left 230 valid units before retry; four additional valid revisions completed the archive. The evaluated stopping policies select 40 and 50 of their respective arm's revisions. These counts exclude all generated revisions after a policy would stop and exclude invalid planner calls. The two arms were run sequentially, so the comparison is not replicated over independently randomized execution orders. Original indexes are retained for the two retried tasks. No pre-retry full-pool result is claimed because the original chains were incomplete.

Pre-registered gate correction (D1). A post-hoc analysis found the gate's mistakes to be almost entirely over-rejection, and that dropping a secondary criterion and relaxing thresholds by 5% would have cut false rejections from 12 to 5 on the data that suggested it. We froze that correction, pre-registered the prediction, and ran it on the development split, which had never been used for a repair experiment. The corrected and original gates agreed on all 120 decision units, because the dominant cause was bound to templates absent from that split. We report it as a failure to replicate rather than as a tuning step, and it is the reason the gate used throughout the paper is the original one.

Effect of diagnostic content (D1). Gating decides which attempts get revised; this ablation asks whether it matters what the reviser is told. On the 39 tasks present in both conditions, a diagnostic rendered from the failing check ended with 33 passing and a message saying only that the attempt had failed ended with 25, a difference of $+0.205$ in pass rate whose template-clustered interval is $[0.000, +0.417]$, including zero, with 12 versus 4 discordant task pairs in the exact McNemar test, $p = 0.077$. The paired test does not establish a difference at the 0.05 level. In addition, under the generic message every selector reached the same score, which is consistent with diagnostic content affecting the outcome; this comparison does not isolate that mechanism. Per-task outcomes are in `evidence/D1/compare_conditions.json`.

The C1 diagnosis separates configuration choice from check aggregation (Table 6). We use a *post-hoc oracle reference*: for each task, the best score any configuration in the frozen candidate space achieves on that task's own rollouts. It is selected after seeing the labels and on the same data it is scored on, so it is a reference value for this candidate space and this data, not an attainable or generalizable ceiling. The reference reaches 95.1%, and taking the prior's most likely configuration without asking anything reaches 84.2%, a gap of 10.9 percentage points. Threshold calibration accounts for at most 5.6 points with the structure held fixed.

Ties prevent the oracle reference from identifying a unique check configuration. On average 10.3 of 14.7 candidate configurations tie for a task's best score. Broken down by axis, the score determines the secondary condition in 19 of 39 tasks, the feature in 12, the threshold tier in 3, and the applicability scope in *none*: in all 39 tasks the tied set contains both the template-level and the family-level scope. Any single configuration reported as "the oracle's" is therefore one arbitrary representative of a large equivalence class, and its applicability axis is decided by the tie-break, not by the data.

The error is one-sided in every configuration we ran: no false acceptance was observed among the 234 rollouts whose label is fail, and the entire observed error is true passes rejected, 161 of 197 when cross-task rulings are allowed and 21 of 197 when they are not. Low false acceptance must

Table 6: C1: where the transfer loss is, on 39 unseen tasks and 431 scored rollouts. Accuracy is computed over decided rollouts, so coverage is reported beside it: abstaining more is not an improvement. The first block varies which single check is used; the second varies whether checks learned on one task may rule on another.

	accuracy	balanced accuracy	coverage	false accepts	true passes rejected
single check, chosen differently					
oracle over the frozen space	0.951	—	431/431	—	—
prior mode, nothing learned	0.842	—	431/431	—	—
after structural transfer	0.789	—	431/431	—	—
random configuration (20 draws)	0.771	—	431/431	—	—
full check suite, same prior					
cross-task rulings allowed	0.617	0.576	420/431	0	161/197
gated by scope belief ($\tau = 0.5$)	0.719	0.685	398/431	0	112/197
own-task checks only	**0.932**	**0.923**	311/431	0	21/197

be interpreted together with coverage and false rejection. The informative quantity is how much false rejection each aggregation rule adds. Together with A1, this identifies two distinct errors: stable execution can leave task requirements unmet, while uncertain check applicability can reject successful motions.

The diagnostic comparisons identify limitations of the implemented transfer mechanisms without establishing their cause. The mechanisms we implemented for carrying checks across tasks did not improve this evaluation: the structural prior transferred negatively (0.576 to 0.534 ungated), and no query rule changed a decision. We cannot conclude that cross-task transfer is unavailable in principle here, because establishing that would require knowing the correct applicability scope, and the data do not determine it in a single task. Under the tested disjunctive aggregation, uncertain applicability is associated with false rejection. Whether applicability gating improves selection at matched coverage requires evaluation on tasks excluded from this diagnosis.

Abstention, obtained by instruction alone, does not track reachability. The strict planner abstained on six tasks, four of which the permissive planner completed successfully; on the eleven requests never satisfied by a tested candidate it abstained zero times and attempted all of them. A negative control supports the evaluator rather than the planner: across those eleven requests and 33 rollouts per arm, no rollout was scored as a task success (permissive: 10 fail, 1 unknown; strict: 9 fail, 2 unknown), so stable execution alone does not induce a passing label.

Direct labeling has the lowest observed loss at the registered abstention cost, but retains only queried verdicts. It stores queried verdicts; the other 76 rollouts retain the prior verifier's decisions. These labels do not transfer to unqueried tasks. It does not repair behavior and it is not a feedback-memory baseline for a planner; a comparison against that stronger alternative is not made here. Three limits apply to the whole table: one seed and one budget point, so variance across seeds and budgets is unestimated; the responder is a language model, not a person; and the 1440 s budget is a modeled cost assigned per question type, not measured response time.

The proposed queries increase coverage from 0.66 to 0.91 relative to the fixed verifier. The proposed rule also raises false rejections from 7 of 44 truly successful rollouts to 12 of 44. The loss ranking depends on the assigned abstention cost. At $\lambda = 0$, where abstaining is free, the proposed rule is the worst arm in the table (0.130 against 0.076 for asking nothing); at $\lambda = 1$, where an abstention costs as much as an error, it is the best (0.217 against 0.239 for direct labels). The crossover occurs near $\lambda \approx 0.4$. We report $\lambda = 0.3$ because it was fixed in advance, but no ranking among these arms survives the sweep, and we do not claim one.

We pre-registered two repairs before running them. Scope gating was predicted to reach balanced accuracy 0.85 and to produce different decisions after learning; it reached 0.685 and left the learning-versus-no-learning comparison at 0 discordant pairs. Putting a floor on the reliability of constant-but-correct answers, which the response model had been treating as carrying no information, raised

the posterior mass on the correct scope from 0.5 to 0.643 without changing subsequent decisions. Five of the seven registered predictions failed. The reported outcomes are evaluated against the registered predictions.

The oracle-answer replay cannot separate response quality from the update mechanism. The simulated responder agrees with an oracle responder on 85–100% of the questions actually asked, and replaying the same query sequence with every answer replaced by the oracle's leaves the downstream numbers unchanged (61.7% accuracy and 57.6% balanced before and after). That comparison is weaker than it looks, because the response likelihood had assigned the applicability panel a reliability of about 2.5×10^{-79}, so those answers were being discarded whatever their content. An unchanged result is therefore compatible with either informative or uninformative answers. The pre-registered reliability floor below addresses part of this and still produced no downstream change, which limits the conclusion to this implementation.

For binary independent labels H and verifier decision A, false acceptance is $P(A = \text{pass} \mid H = 0)$, accepted-set error is $P(H = 0 \mid A = \text{pass})$, and decision coverage is $P(A \neq \text{unknown})$. Undefined denominators are reported explicitly. Coverage and abstention rates accompany error rates so that rejecting or deferring every candidate cannot appear to solve verification.

Response cost is modeled, not measured. No human answered any question in these experiments. Each question type is charged a fixed cost: 30 s for an applicability question, 40 s contrastive, 45 s for a threshold, 60 s for a feature choice, 90 s for a verdict label. The 1440 s budget is that model's currency. Measured human response times would change the arms' question counts and are not available.

Clustering. The independent unit is the instruction template, because tasks generated from one template share wording, parameters, and the plan the planner tends to produce. B1's 92 rollouts come from 16 tasks over 9 templates; C1's 431 rollouts come from 39 tasks over 25 templates. Table 7 gives the B1 paired loss differences under bootstraps at both levels. Template-level intervals are wider, as expected, and are the ones quoted in the body.

Table 7: B1 paired loss difference against the no-questions verifier, full panel, $\lambda = 0.3$, 10 000 cluster bootstrap resamples. Negative favours the arm.

arm	difference	95% CI by task	95% CI by template
direct labels	-0.052	$[-0.088, -0.022]$	$[-0.094, -0.017]$
random queries	-0.030	$[-0.097, +0.030]$	$[-0.107, +0.038]$
uncertainty queries	0.000	[0.000, 0.000]	[0.000, 0.000]
proposed	-0.021	$[-0.119, +0.090]$	$[-0.147, +0.102]$

Sensitivity to abstention cost. Table 8 sweeps λ, the loss charged for returning unknown, holding everything else fixed. At $\lambda = 0$ the proposed rule is the worst arm, since it abstains least and abstention is free; at $\lambda = 1$ it is the best, for the same reason. The crossover is near $\lambda \approx 0.4$. We fixed $\lambda = 0.3$ before running, and we do not claim a ranking that a different defensible choice would reverse.

Table 8: Full-panel loss as a function of the abstention charge λ. The post-hoc oracle reference never abstains and has loss 0.076 throughout.

λ	0.0	0.1	0.3	0.5	0.7	1.0
fixed verifier	0.076	0.110	0.177	0.245	0.312	0.413
direct labels	0.076	0.092	0.125	0.158	0.190	0.239
random queries	0.098	0.114	0.147	0.179	0.212	0.261
uncertainty queries	0.076	0.110	0.177	0.245	0.312	0.413
proposed	0.130	0.139	0.157	0.174	0.191	0.217

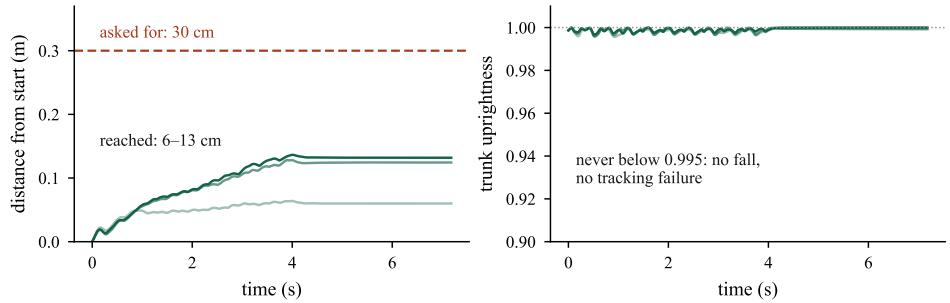


Figure 5: A silent failure, from the recorded trajectories of task F1-001 under the permissive prompt (three seeds). The request was 30 cm of forward walking; the robot traveled 6 to 13 cm. Left: distance from the start. Right: trunk uprightness, a diagnostic the agent can compute without knowing the task. The motion passes this stability diagnostic. The diagnostics available to the agent do not separate it from the rollouts that did satisfy the request, which is the difficulty this paper addresses.

Table 9: C1 aggregation diagnosis on 431 rollouts from 39 tasks. Accuracy is conditional on a decision; restrictions increase abstention. These are diagnostic comparisons on the same pool, not held-out confirmation of a mechanism.

Check aggregation	Accuracy	Decided	False rejections
Cross-task rulings	0.617	420/431	161/197
Scope belief > 0.5	0.719	398/431	112/197
Own-task checks	0.932	311/431	21/197

F.1 Additional experimental details

The A1 paired task outcomes are 13 successes under both prompts, 17 failures under both, and 5 versus 4 successes unique to either prompt (exact McNemar $p = 1.00$, $n = 39$). The 39 tasks span 25 templates. Four of the six reachable tasks on which the strict planner abstains succeed under the permissive prompt.

C1 also evaluates fixed changes to check scope. Forcing applicability to the task template corrects 12 recorded decisions with no regressions. Threshold calibration accounts for at most 5.6 percentage points of the post-hoc oracle gap. The structural prior reduces the ungated score from 0.576 to 0.534. These diagnostics share the original trajectory pool and do not constitute new-template confirmation.

D1's revision reduction remains below -1.68 per task when any one template is omitted. In the first-round learning replay, five of the twenty tasks skipped by the pre-feedback verifier are initially failing; five random additions yield a 95% outcome band of $[-1, +2]$. The excluded interaction tasks are F2-027 (failing) and F2-028 (passing), both accepted initially. Including them would give gated repair 42/49 successes.

G Supplementary Evaluation Results

The following tables and diagnostics supplement the main comparisons, preserving protocol details, uncertainty estimates, and exploratory analyses.

G.1 Platform and sampling details

We use a simulated biped and a restricted motion library to separate motion execution from task interpretation. Microduck has 14 actuated joints and is 25 cm tall (Figure 1). A fixed controller trained with proximal policy optimization (PPO) maps 85-dimensional observations to 14 joint targets in MuJoCo. The planner scales, mirrors, repeats, or concatenates seventeen clips recorded from nine publicly released policies. No experiment retrains the controller. Appendix D describes the observations and motion library.

Table 10: D1: the verifier as a repair gate, three rounds, pooled over three splits. Intervals are cluster bootstraps over instruction templates, 10 000 resamples; the random row is the mean and 95% band over 10 000 draws with fixed per-split seeds. The pre-registered McNemar test gives $p = 0.0625$ (five corrections, no regressions); clustered intervals are descriptive.

Policy	Tasks	Successes	Revisions
no repair	47	22	0
random allocation, gate's budget	47	31.9 $[27, 36]$	59
gated by the verifier	47	**41**	**59**
repair everything	47	36	141
Gate minus repair-all: difference [95% interval]			
Success rate		$+0.106$ $[+0.024, +0.196]$	
Revisions per task		-1.74 $[-2.19, -1.27]$	

Table 11: Three-round selection replay on 37 tasks. Both snapshots select from identical recorded revisions. The first-round results reproduce the earlier learning comparison.

	Round 1		Round 3	
Verifier	Successes	Revisions	Successes	Revisions
Before feedback	22	17	30	41
After feedback	25	22	34	49

The task pool covers direction and displacement, head motion, and posture or event sequences. Its 49 tasks span 25 templates: 10 development, 16 interaction, and 23 test tasks, with 16, 18, and 15 tasks in the three families. Each task has at least one independently successful candidate and is executed with three simulator seeds. Eleven additional requests over nine templates were never satisfied by a tested candidate; they are evaluated separately for abstention. Chinese instructions and prompts are provided with translations in Appendix H.

Evaluation specifications and feedback have distinct sources. One researcher authored the task specifications, while all interactive responses come from a fixed language model. No human judgments or pairwise preference responses enter the reported experiments. For uncertainty estimates, we resample instruction templates, keeping tasks and rollouts from each template together. A template defines a task pattern, such as turning in place, whose instances vary parameters such as angle or direction. This template-clustered bootstrap accounts for dependence among related tasks; individual rollouts are not independent observations.

The failures reflect specific task requirements that stability cannot certify. Discrete leg events are particularly affected: 9/11 stable tasks fail, compared with 3/13 for posture sequences under the permissive prompt. Figure 5 gives a displacement example: scaling a walking clip to amplitude 0.7 produces only 6–13 cm of travel for a 30 cm request, despite trunk uprightness above 0.995 across all three seeds. Thus, accurate execution of a reference motion does not establish that the reference implements the instruction.

G.2 Transfer diagnostics

Uncertain applicability is associated with rejecting successful trajectories. Under disjunctive aggregation, any failed check can reject an attempt, including checks with uncertain scope. Restricting cross-task rulings reduces false rejection but also reduces coverage (Table 9). Scope gating corrects 37 recorded decisions with no regressions. Own-task checks reject 21/197 successful trajectories, compared with 161/197 under cross-task rulings. No false acceptance is observed among the 234 failing trajectories. These diagnostics do not establish a gain in selection at matched coverage.

The candidate space contains better-scoring configurations but does not identify a unique correct scope. A post-hoc oracle reference reaches 95.1% accuracy versus 84.2% for the prior mode, selecting and scoring configurations on the same labels. On average 10.3 of 14.7 configurations tie

for the maximum, and both scopes occur among the maximizers for every task. Consequently, this comparison cannot show that a particular scope is correct or that transfer is structurally impossible.

The response model and aggregation rule limit what can be inferred from the null transfer result. An oracle-answer replay also leaves decisions unchanged, but the likelihood assigns applicability answers reliability near 2.5×10^{-79} and effectively ignores them. A registered reliability floor and structural-prior correction do not recover transfer; five of their seven predictions fail (Appendix F). Applicability-gated aggregation therefore remains a mechanism to test on new templates, rather than a confirmed explanation.

G.3 Repair protocols and audits

Experiment D measures the allocation value of the frozen verifier over at most three repair rounds. Policies select stopping points on the same recorded revision chains, following the protocol in Section 2.

The baselines separate selective repair from additional revision expenditure. Repair-all revises every task in every round; no repair retains the initial attempt. A budget-matched random policy repeatedly samples uniformly among tasks with remaining revision capacity, spending the gate's 59 revisions over 10 000 draws with distinct fixed seeds by split. A stability gate stops once execution is stable. The pooled comparison contains 47 tasks; two initially accepted interaction tasks, one passing and one failing, lack third-round records and are excluded.

Gated repair achieves more successful tasks with fewer revisions than repair-all (Table 10). It reaches 41/47 successes with 59 revisions, compared with 36/47 with 141 revisions, a 58% reduction in revisions. At the same budget, random allocation achieves 31.9 successes on average; no draw reaches 41. This Monte Carlo comparison is conditional on the recorded chains and the tested random rule. Repair-all peaks at 37 successes after two rounds before declining to 36 (Figure 3).

Execution stability alone provides little allocation benefit on the subset where it can be evaluated. On 37 tasks, the stability gate stops immediately on 35, uses 3–4 revisions, and yields 19 successes, compared with 17 without repair, 34 with the learned gate, and 30 with repair-all. Aggregate later-round stability records leave one task's revision count uncertain.

Learning with separately generated revision chains. Feedback-updated checks improve final success when each verifier supplies the diagnostics for its own revisions (Table 3). We compare pre-feedback and updated snapshots on 39 tasks from 20 templates, holding the initial records, controller checkpoint, planner configuration, three simulator seeds, and three-round cap fixed. Each arm generates new reference motions from its own check-derived feedback. The driver executes all tasks in each round; each verifier's stopping rule is then applied to its recorded chain. Thus, the comparison includes changes in diagnostics and generated motions, but does not measure a live early-stopping implementation.

The updated verifier reaches 35/39 successes (89.7%), compared with 29/39 (74.4%) before feedback. Seven tasks succeed only after the update and one only before it. The paired success-rate difference is $+0.154$, with a 95% template-clustered bootstrap interval of $[+0.024, +0.308]$; the exact task-level McNemar test gives $p = 0.0703$. The gain is concentrated on interaction tasks (13 versus 8 successes), with one additional success on the test split (22 versus 21). These results support a benefit on the evaluated pool, with limited evidence for transfer beyond it.

The learning benefit comes with additional selected revisions. The updated verifier selects 50 revisions rather than 40, a difference of $+0.26$ per task with a template-clustered interval of $[-0.03, +0.61]$. The comparison therefore tests outcomes under a common round cap, not at equal revision expenditure. The updated arm loses previously achieved success on one task, compared with three before feedback; this counts a passing state at any earlier round followed by failure at termination. These results are separate from the savings against repair-all in Table 10.

Two invalid planner outputs required retries in the pre-feedback arm. Round-two responses for F3-045 and F2-034 contained no valid JSON; those tasks were regenerated through rounds two and three, with the first-pass indexes retained. F3-045 is selected through round three and ends in success after retry. F2-034 stops at its initial state under the scored policy, so its later retry does not affect that policy's final outcome. Retry eligibility was implemented after these failures; we do not

attribute them to a transient API fault or claim a prespecified retry protocol. Appendix F records the execution accounting.

Selection on shared revision chains. A complementary replay isolates the effect of selection while holding revisions fixed (Table 11). Applying both snapshots through all three rounds of the same recorded chains yields 34 successes after feedback and 30 before it on 37 tasks from 19 templates. The updated policy selects 49 revisions, compared with 41 before feedback. Four tasks succeed only after the update and none only before it; the success-rate difference is $+0.108$ with a template-clustered interval of $[0.000, +0.250]$ and exact McNemar $p = 0.125$. This agrees in direction with the separately generated chains, while evaluating a different intervention.

The comparisons separate revision allocation from the effect of updating the verifier. The 47-task repair-all comparison measures allocation by the deployed verifier; the 39-task comparison changes the snapshot and regenerates revisions; the 37-task replay changes selection on fixed revisions. Recomputing deployed selections from raw trajectory features matches the recorded lists across all three rounds on the interaction and test splits. This checks consistency of those decisions, rather than constituting a complete information-flow audit. Revision counts exclude feedback, verification, learning, and experimental execution overhead.

H Task Specifications and Independent Evaluation

Each task record carries an instruction, family, template, split, reachability evidence, and an evaluator specification held separately from everything the agent reads. Directional tasks fix a coordinate frame, an observation window, and a displacement tolerance; event tasks fix onset, completion, ordering, and duration tolerances. The evaluator is unit-tested on synthetic trajectories with known answers, including boundary cases. Instructions are in Chinese, as authored; translations accompany every instruction quoted in this paper. A task is listed as reachable only when at least one candidate motion was independently scored as passing it, and the 11 requests in the separate pool are described as *never satisfied by any candidate we tested*, which is what was observed; it is not a proof of unreachability.

I Verifier Versions and Reproducibility

Every run records source revision, checkpoint hash, simulator configuration, model identifiers, prompts, seeds, and immutable input and output paths in a manifest. Experiments A–C use one tracking checkpoint, taken at 1500 iterations, because B1 and C1 score rollouts that A1 recorded; a better 6000-iteration checkpoint exists (body-position error 0.0176 against 0.0114) and was not used, so the execution-stability figures come from the weaker controller. Each verifier update records its parent version, the query, the answer, the tasks it applies to, and the checks it changed.

Run isolation is enforced rather than assumed, because an earlier version of this pipeline reported twenty of twenty tasks passing when a failed evaluator subprocess allowed the caller to read a result file left by a previous successful run, so the numbers described a different motion library. Every run now writes to its own directory, a non-zero exit produces no result, and a result file is rejected unless it carries the current run identifier and was written after the call began. The regression test for exactly this failure runs without a GPU.

J Human Judgment Kit

The annotation kit supports a future comparison between instruction-based judgments and the evaluator's explicit criteria. It has been piloted internally, but no formal human evaluation contributes to this paper's results. The English-language interface collects instruction-only judgments for all clips before beginning a separate criterion-by-criterion audit (Figure 6).

Rendering must preserve the information needed to judge the instruction. A tracking camera obscures displacement relative to the world, while sparse frames can miss discrete leg events. The renderer therefore uses a fixed world camera and full-rate video with replay and slow motion. The criterion view includes a 10 cm grid, start line, and target band; the left and right feet are consistently

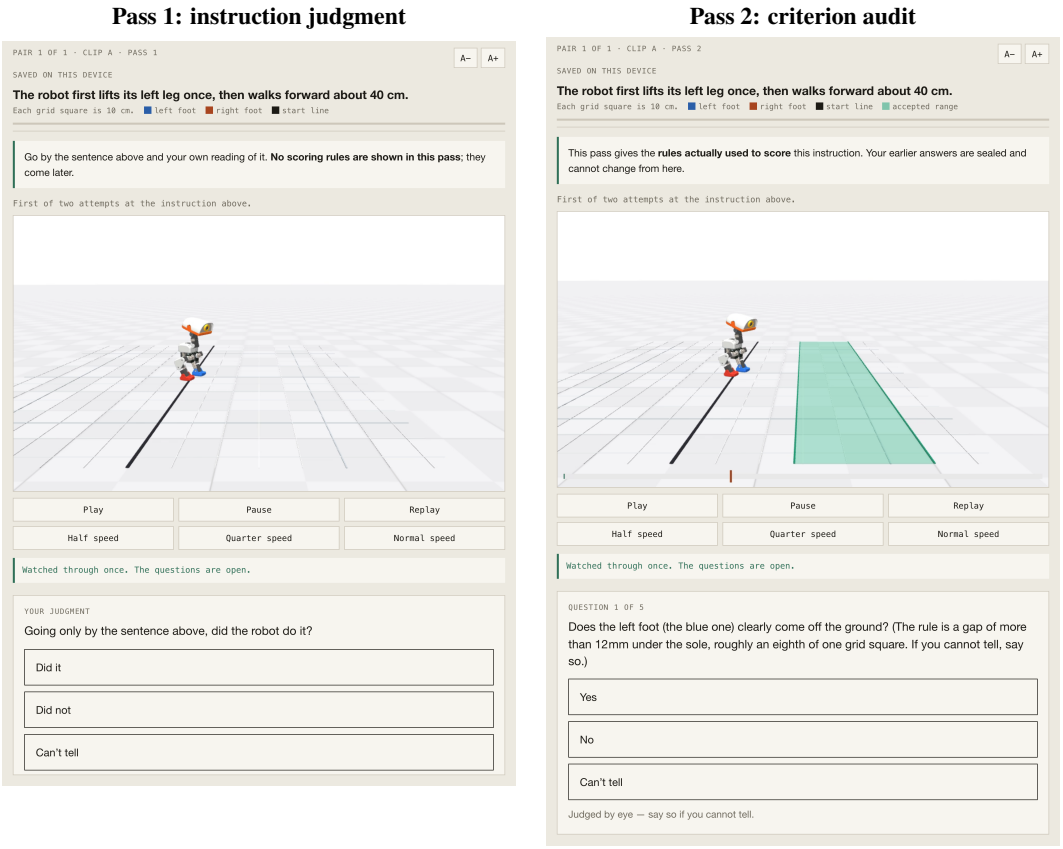


Figure 6: English-language annotation interface, illustrated with demo clips. **Left:** annotators first judge every clip against the instruction alone. **Right:** after all first-pass answers are sealed, a second pass presents explicit scoring criteria and marked videos. The two passes are not interleaved within a clip. Neutral grid lines and foot colors remain visible in both views; acceptance bands and timing markers appear only in the criterion view. Replay, slow motion, and a *Cannot tell* response support uncertain judgments. Visually imprecise criteria are labeled as qualitative audits. No method names or automated scores are shown. These interface examples are not human-evaluation results.

colored blue and red. When joint states require re-executing a saved plan, per-seed uprightness is checked against the scored record to detect mismatched trajectories.

Upon acceptance, we will publicly release the source code under an open-source license, together with experimental protocols, task specifications, prompts, analysis scripts, and the associated experimental data needed to reproduce the reported results.

K Prompts and Result Provenance

The evidence archive contains the raw per-rollout inputs used to recompute Experiments A–D. The reproduction script checks the recorded-chain D1 table against the manuscript, reproduces the B1 loss and template-bootstrap outputs, and recomputes the C1 diagnosis. Separate scripts reproduce the three-round learning replay and the comparison with separately generated revision chains. This reproduces analyses from archived records, not new planner responses or simulator executions.

Two protocol deviations affect the interpretation of the feedback experiments. B1 uses own-task aggregation after the C1 diagnosis showed that cross-task aggregation suppressed differences between query rules. The two C1 corrections were registered before their follow-up runs, but chosen after examining the target pool, so those comparisons remain exploratory. The manuscript does not reproduce the full prompt inventory; model and prompt records must be distributed with the corresponding experimental evidence.

L SCOPE AND LIMITATIONS

The findings establish a bounded result for one simulated embodiment and one revision procedure. The controller and feature vocabulary are fixed, and specifications reflect one researcher's interpretation of the instructions. Conditions missing from the vocabulary can remain undetected. Feedback is automated, the preference component is untested, and neither human interaction nor hardware transfer is evaluated.

The learning comparisons cover one pair of verifier snapshots on an existing task pool. Separately generated chains include each snapshot's diagnostics and revisions, but stopping is evaluated offline, and most of the gain occurs on interaction tasks. Additional runs are needed to assess sensitivity to learning and planner randomness; new templates are needed to test generalization. Simulator seeds within a task do not provide independent learning runs. Equal-expenditure comparisons, a live stopping implementation, and a new-template mechanism test remain open.