YOUR POLICY IS SECRETLY A HARNESS: THE DECISION GEOMETRY OF AGENT INTERFACES

Anonymous authors
Paper under double-blind review

ABSTRACT

Language model agents are trained through a harness of prompts, tools, and control logic. Under KL-regularized control, an optimal policy's log odds between executed actions recover the relative action scores that its harness induces, which determine every optimal decision. In web tasks whose control problems we solve exactly, post-training moves a model's action scores toward its training harness's optimum and makes them specific to that harness. Even reversing the order of the observation fields, which keeps every optimal decision, moves these scores and lowers a trained model's success from 85.0% to 19.2%. Action scores fix decisions but leave out the best return a harness offers, which we call its ceiling. An agent's return equals the ceiling minus its mismatch, the return lost to suboptimal decisions along its own trajectories. Training changes only the mismatch. Adding batched clicks raises the ceiling but lowers a trained model's success, and after equal further training under each harness, the batched-click harness gives the best agent. Among swaps between several harnesses, a frozen comparison rejects seven that raise the ceiling, and after adaptation all seven exceed the original agent's return. Ceiling and mismatch measured before adaptation predict the adaptation gains far better than the frozen return change. With the optimum known, mismatch estimated from sampled actions predicts how harness changes and adaptation move return. For coding agents, teacher-based estimates of ceiling minus mismatch order several harnesses by accuracy. A frozen comparison of harnesses therefore mixes what each harness offers with how well the checkpoint uses it.

1 INTRODUCTION

Language model agents are post-trained and deployed through a harness, the prompts, tools, and control logic that present a task to the model and execute its outputs. Interface design and prompt formatting change agent performance (Yang et al., 2024; Sclar et al., 2024). Training can move behavior that a prompt or a harness supplies into the weights (Snell et al., 2022; Ye et al., 2026), and recent methods adapt harnesses and weights together (Hebbar et al., 2026; Chen et al., 2026; Kim et al., 2026). These practices raise the question we study:

What does a policy learn from its harness, and what survives when the harness changes?

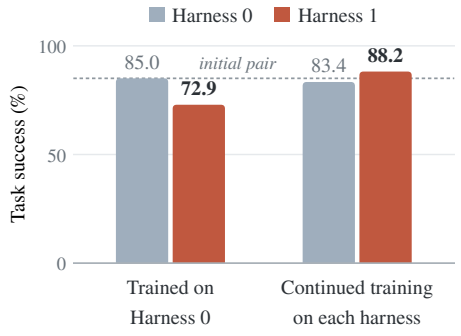

Figure 1: Adaptation reverses a frozen comparison. Colors mark the evaluation harness.

Two experiments make the question concrete. In the first, we take a web agent post-trained under one harness and reverse the order of its observation fields. The actions, their execution, the rewards, and the reference policy stay the same, so every optimal decision stays the same. The model still changes its most probable action at about half of the states, and its task success falls from 85.0% to 19.2%. Keeping the decisions a model should make does not keep the decisions it makes.

In the second, a new harness adds batched clicks, which execute two clicks in one decision. The new actions raise the best achievable return, yet the trained model's success drops, and a frozen

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.

comparison would reject the new harness. After the same further training under each harness, the model trained under the new harness succeeds more often than both the initial model and the model trained further under the original harness. The harness that first hurt the model gives the best agent. Figure 1 shows the task success at each step, with the new harness as Harness 1.

We explain both results in the control problem that a harness induces over semantic actions, the operations an agent executes together with their arguments. Under KL-regularized control, the optimal policy is a softmax over action scores, each adding an action's long-term value to its reference log probability, so an optimal policy's log odds between two actions are proportional to the difference of their scores. This is the sense of our title. A policy trained to optimality under a harness carries the relative action scores that the harness induces, and behavior reveals nothing more, since different rewards, reference policies, and transition laws can induce the same scores. In our controlled study, real post-training moves a model's action scores toward the optimum of its training harness, and each harness leaves its own preferences in the model.

The same coordinates show why transfer fails. A new harness can move the optimal scores, and it can move the scores that the model expresses through the interface, which we call its readout. Reversing the observation fields moves only the readout.

To convert these changes into return, we apply the regularized performance-gap identity (Geist et al., 2019; O'Donoghue, 2022) to the harness-induced problem. Deployed return equals the harness's ceiling C, its optimal return, minus the policy's mismatch M, the KL divergence from the optimal policy accumulated along the policy's trajectories. Training under a fixed harness changes only M, while a harness change can move both terms. A frozen comparison of harnesses therefore mixes what each harness offers with how well the checkpoint uses it. Batched clicks raise the ceiling slightly and the frozen model's mismatch ten times as much, and adaptation then lowers it below a threshold we derive. Figure 2 illustrates both steps. Our contributions are as follows.

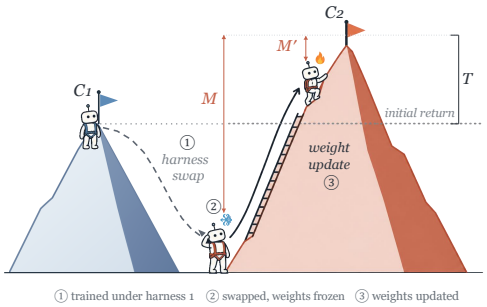

Figure 2: A harness revision and a weight update act on different terms. Summit heights are the ceilings, the robot's altitude is its return, and the gap is the mismatch. The swap raises the ceiling but lowers the return. The weight update lowers only the mismatch, and the adapted pair outperforms the initial pair once its mismatch M' falls below T.

1. What a policy learns under a harness. Optimal action log odds identify the relative action scores that a harness induces, and different mechanisms can share them. On MiniWoB++, post-training moves action preferences toward the training harness's optimum and makes them harness-specific.

2. Why transfer fails. We split a transferred policy's score error into a target shift, a training residual, and a readout drift. Presentation changes that keep every optimal decision isolate the readout drift and can make a trained model fail on most tasks.

3. What transfer costs and how to measure it. Ceiling minus mismatch gives deployed return exactly and the mismatch threshold at which adaptation makes a harmful revision beneficial. Batched-click adaptation crosses this threshold and outperforms continued training under the original harness. Across six harnesses, ceiling and mismatch predict which rejected swaps adaptation makes beneficial. Mismatch estimated from sampled actions predicts return changes when the optimum is known, and teacher-based estimates order five coding-agent harnesses by Terminal-Bench Lite accuracy.

2 Harnesses Induce Semantic Control Problems

A harness turns model outputs into executed operations. We call an executed operation together with its arguments a semantic action $a \in \mathcal{A}$, such as clicking a particular element or running a query. Different outputs can execute the same action. For example, a JSON object and a function-call string may issue the same parameterized query. Let $s \in \mathcal{S}$ denote the semantic decision state. An interface $\mathcal{I} = (c_{\mathcal{I}}, d_{\mathcal{I}}, e_{\mathcal{I}})$ formalizes a harness. It encodes the state into a context, decodes a model response into an action, and executes that action. For a token policy π_θ, the induced semantic policy sums the

probabilities of the outputs that decode to the same action:

$$\Pi_{\theta,\mathcal{I}}(a \mid s) = \sum_{y:\, d_{\mathcal{I}}(y;s)=a} \pi_\theta(y \mid c_{\mathcal{I}}(s)). \tag{1}$$

Both the context $c_{\mathcal{I}}$ and the decoder $d_{\mathcal{I}}$ enter Eq. (1), so fixed weights do not fix the semantic policy.

Together with a fixed task environment, $\mathcal{I}$ induces a transition law $P_{\mathcal{I}}$, a utility $r_{\mathcal{I}}$ that includes execution costs, and a semantic reference policy $\pi^{\mathrm{ref}}_{\mathcal{I}}$. Every call to the evaluated model is a decision, including calls for verification or retry, and the state includes the harness's controller memory, such as tool results and remaining budget. Appendix B specifies this construction and the conditions under which training over tokens yields semantic optimality.

Across comparisons, we fix the initial distribution ρ_0, discount $0 \le \gamma < 1$, and regularization coefficient $\tau > 0$, and measure rewards on a common scale. Each interface's reference is part of its control problem and does not change during training. For finite $\mathcal{S}, \mathcal{A}$, bounded rewards, and $\pi^{\mathrm{ref}}_{\mathcal{I}}(a \mid s) > 0$, the regularized return of a semantic policy Π is

$$J_{\mathcal{I}}(\Pi) = \mathbb{E}_{\rho_0,\Pi,P_{\mathcal{I}}}\left[\sum_{t\ge 0}\gamma^t\left(r_{\mathcal{I}}(s_t,a_t) - \tau\log\frac{\Pi(a_t \mid s_t)}{\pi^{\mathrm{ref}}_{\mathcal{I}}(a_t \mid s_t)}\right)\right]. \tag{2}$$

Write $\Pi^*_{\mathcal{I}}$ for the optimal policy and $d^{\Pi}_{\mathcal{I}}(s) = (1-\gamma)\sum_{t\ge 0}\gamma^t \Pr^{\Pi}_{\mathcal{I}}(s_t = s)$ for the discounted occupancy of Π.

Controlled study. In our mechanism experiments, every quantity above can be computed exactly. Six harnesses act on three MiniWoB++ task families (Liu et al., 2018), and we compile each induced control problem into a finite graph. H0 executes one native action per decision, and H1–H5 add batched clicks, automatic checks, verification tools, or extra instructions. Each prompt lists the complete action menu with one token per action, so the model's next-token distribution gives $\Pi_{\theta,\mathcal{I}}$ exactly. The reward is binary task success. We post-train Qwen3.5-2B with exact policy gradients, and the frozen base model serves as the reference. A soft Bellman solve gives each harness's optimum. Evaluations use tasks held out from training, and Appendix F.1 gives the full protocol.

3 What a Policy Learns under a Harness

We now ask what a model's action probabilities reveal about the control problem it was trained on.

3.1 Optimal Policies Encode Relative Action Scores

The optimal action value $Q^*_{\mathcal{I}}$ depends on the harness's utility and transition law through the soft Bellman equations, and the optimal policy takes the Gibbs form $\Pi^*_{\mathcal{I}}(a \mid s) \propto \pi^{\mathrm{ref}}_{\mathcal{I}}(a \mid s)\exp(Q^*_{\mathcal{I}}(s,a)/\tau)$ (Haarnoja et al., 2017; Geist et al., 2019). An action's optimal probability therefore depends on its long-term value and its reference probability. We combine the two into the optimal action score

$$F^*_{\mathcal{I}}(s,a) = Q^*_{\mathcal{I}}(s,a) + \tau\log\pi^{\mathrm{ref}}_{\mathcal{I}}(a \mid s), \qquad \Pi^*_{\mathcal{I}}(\cdot \mid s) = \mathrm{softmax}\big(F^*_{\mathcal{I}}(s,\cdot)/\tau\big). \tag{3}$$

For instance, an optimal policy may favor a batch of clicks because the batch leads to better outcomes or because the reference places more weight on it. The score records the combined effect.

Adding the same constant to every action's score at a state leaves the policy unchanged, so only score differences within a state affect decisions. We write $[F]$ for the class of score fields related by such statewise constants. Two interfaces have *equivalent optimal decisions* when $\Pi^*_{\mathcal{I}_1}(\cdot \mid s) = \Pi^*_{\mathcal{I}_2}(\cdot \mid s)$ at every aligned state.

Proposition 3.1 (Behavioral identification)**.** *Two interfaces have equivalent optimal decisions if and only if* $[F^*_{\mathcal{I}_1}] = [F^*_{\mathcal{I}_2}]$. *If a checkpoint is optimal for its training interface at every semantic state, then*

$$[\tau\log\Pi_{\theta,\mathcal{I}_{\mathrm{tr}}}] = [F^*_{\mathcal{I}_{\mathrm{tr}}}], \qquad \tau\log\frac{\Pi_{\theta,\mathcal{I}_{\mathrm{tr}}}(a \mid s)}{\Pi_{\theta,\mathcal{I}_{\mathrm{tr}}}(b \mid s)} = F^*_{\mathcal{I}_{\mathrm{tr}}}(s,a) - F^*_{\mathcal{I}_{\mathrm{tr}}}(s,b). \tag{4}$$

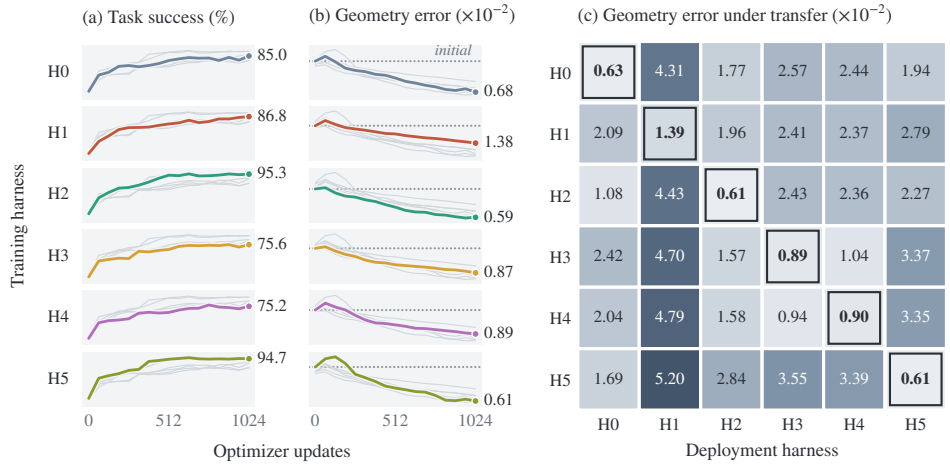

Figure 3: Post-training moves action preferences toward the optimum of the training harness and makes them specific to it. Rows denote training harnesses. (a) Task success and (b) geometry error relative to the training harness's optimum during post-training. Colored curves show the harness of each row, gray curves the other harnesses, and dotted lines the error at initialization. (c) Geometry error of each final checkpoint under each deployment harness. Outlined entries mark the lowest error in each column, attained by the checkpoint trained under that harness.

Similar relations between policies, rewards, and values underlie DPO and its token-level extension (Rafailov et al., 2023; 2024). Here the probabilities belong to executed semantic actions under a specified harness, so the identified object is the decision structure that the harness induces.

Behavior determines these relative scores and nothing more about the harness. Different reference policies, rewards, and even transition laws can share one optimal score field and therefore one optimal policy, in line with partial identifiability results for reward learning (Skalse et al., 2023). Behavior also leaves absolute value undetermined, since adding a constant to every reward raises the optimal return and changes no decision. A policy can thus carry the decisions a harness induces without its code, tools, or execution machinery. The value a harness offers is a separate quantity, which Section 5 adds back. Appendix C proves the proposition from the Gibbs form and constructs such mechanisms.

A real checkpoint departs from the optimum. To compare the two, we represent each class by its centered member, subtracting the mean score across actions at each state: $\bar{F} = P_\perp F$ with $P_\perp = I - \mathbf{1}\mathbf{1}^\top/n$ and $n = |\mathcal{A}|$. For positive semantic probabilities, the checkpoint's centered scores are $\widehat{F}_{\theta,\mathcal{I}} = \tau P_\perp \log \Pi_{\theta,\mathcal{I}}$. We call them the model's readout through $\mathcal{I}$, the action preferences it expresses through the interface. Its score error under the training interface, $e_{\mathrm{tr}} = \widehat{F}_{\theta,\mathcal{I}_{\mathrm{tr}}} - \bar{F}^*_{\mathcal{I}_{\mathrm{tr}}}$, vanishes exactly when the checkpoint is optimal.

3.2 Post-Training Makes Preferences Harness-Specific

Proposition 3.1 concerns optimal policies. We ask whether real post-training moves a model toward the optimum of its training harness, and whether what it learns depends on that harness. We post-train Qwen3.5-2B under each of H0–H5. The geometry error of a checkpoint is the mean squared difference between its centered scores and the optimal ones, averaged over a fixed bank of states.

Under every harness, post-training raises task success, and the geometry error first rises and then falls well below its initial level, on held-out calibration tasks as much as on training tasks. The model learns effective task behavior while its action scores approach the optimum of its training harness.

To test whether the learned preferences depend on the training harness, we deploy each final checkpoint under all six harnesses on test tasks. A fixed deployment harness fixes the dynamics, reference, and optimal scores, so the six checkpoints are compared against the same target. Under every deployment harness, the checkpoint trained under that harness has the lowest error. Post-training thus makes the model's preferences specific to its training harness. Figure 3 shows the learning curves in its first two panels and this comparison in the third.

Takeaways: What a policy learns

A policy trained to optimality under a harness carries that harness's relative action scores. Real post-training moves toward them and leaves harness-specific preferences.

4 Why Policy Transfer Fails

The same weights need not express the same preferences under another harness.

4.1 A Harness Change Moves the Target and the Readout

Let interfaces 1 and 2 be the training and deployment interfaces. A change in execution can change which actions are optimal, and a change in presentation can change which actions the frozen model selects. On aligned states and actions, the deployed score error splits into three terms:

$$\underbrace{\widehat{F}_{\theta,2} - \bar{F}_2^*}_{\text{deployed error}} = \underbrace{\bar{F}_1^* - \bar{F}_2^*}_{g_{12}\text{: target shift}} + \underbrace{\widehat{F}_{\theta,1} - \bar{F}_1^*}_{e_1\text{: training residual}} + \underbrace{\widehat{F}_{\theta,2} - \widehat{F}_{\theta,1}}_{\eta_{12}\text{: readout drift}}. \tag{5}$$

The training residual exists before the change, and the target shift and the readout drift are its two effects. They are vectors, so they can add up or cancel. When the reference comes from a reference model, a new context can also shift the reference probabilities and thereby move both the target and the readout. Isolating the readout therefore requires holding the semantic reference fixed.

4.2 Equivalent Optimal Decisions, Different Behavior

When two interfaces have equivalent optimal decisions, $g_{12} = 0$, and any change in the model's error comes from readout drift. We test this with the H0-trained checkpoint. We reverse the order of the observation fields, render the action menu as a list, or reassign the action codes, and keep the transitions, rewards, and semantic reference fixed. The optimal policy and the optimal return thus stay the same, as numerical checks confirm.

Table 1: Equivalent optimal decisions, different choices. Choice change is the share of states whose most probable action changes from the original. M and J are the mismatch and regularized return of Section 5.2.

Presentation	Choice change	M	J	Task success
Original	0.0%	0.015	0.423	85.0%
Reversed fields	52.0%	0.484	−0.046	19.2%
Menu as list	9.2%	0.023	0.415	86.0%
Shifted codes	27.7%	0.148	0.289	57.1%
Reversed codes	30.0%	0.130	0.308	68.2%
Permuted codes	25.3%	0.110	0.328	67.6%

Equivalent optimal decisions do not produce equivalent behavior. Reversing the field order has the largest effect. It changes the model's most probable action at about half of the states, and most tasks then fail. Reassigning the action codes has a moderate effect, and the list format has a small one. Prompt-format sensitivity is known (Sclar et al., 2024). Here the control problem certifies that the optimal policy is unchanged, so the entire effect comes from the model's readout. Table 1 lists the results, Appendix F.2 gives the protocol, and Appendix D.4 bounds readout drift by the model's sensitivity to corresponding outputs.

Takeaways: Why transfer fails

A harness change can move the optimal decisions and the model's choices separately. A new presentation alone can make a trained model fail on most tasks.

5 What Transfer Costs and When Adaptation Recovers It

Separating the two changes shows what moved. What they cost in return depends on which decisions they affect and how often the policy reaches them.

5.1 From Score Errors to Lost Return

Consider a single decision at state s in a fixed interface $\mathcal{I}$. If the policy Π makes this decision and the optimal policy makes every later one, the return lost relative to acting optimally throughout is $\tau \mathrm{KL}(\Pi(\cdot \mid s) \| \Pi^*_{\mathcal{I}}(\cdot \mid s))$. If Π also makes the later decisions, each of them adds its own loss, and summing along the trajectories of Π gives the total gap.

Proposition 5.1 (Regularized performance gap). *For any semantic policy* Π *in a fixed interface,*

$$J_{\mathcal{I}}(\Pi_{\mathcal{I}}^*) - J_{\mathcal{I}}(\Pi) = \frac{\tau}{1-\gamma}\mathbb{E}_{s\sim d_{\mathcal{I}}^{\Pi}}\mathrm{KL}\big(\Pi(\cdot \mid s)\|\Pi_{\mathcal{I}}^*(\cdot \mid s)\big). \tag{6}$$

This is the regularized-control gap identity (O'Donoghue, 2022) applied to the harness-induced problem. The occupancy weights each local loss by how often the evaluated policy reaches the state, so errors on the policy's common paths cost the most. In score coordinates, the local loss is the Bregman divergence $D_\Phi(F, G)$ of $\Phi(F) = \tau \log \sum_a e^{F_a/\tau}$, the gap between $\Phi(F)$ and the linear approximation of Φ at G. For positive semantic action probabilities,

$$\tau\mathrm{KL}(\Pi_{\theta,\mathcal{I}}\|\Pi_{\mathcal{I}}^*) = D_\Phi(F_{\mathcal{I}}^*, \widehat{F}_{\theta,\mathcal{I}}). \tag{7}$$

Action scores supply the coordinates, and D_Φ measures the cost of a decision error. Together they form the decision geometry of a harness, and averaging D_Φ under deployment occupancy turns the score errors of Eq. (5) into lost return. Appendix D proves both identities.

5.2 Ceiling and Mismatch

Relative scores fix the decisions and leave the value of a harness undetermined. We add that value back. Define the ceiling $C(\mathcal{I}) = J_{\mathcal{I}}(\Pi_{\mathcal{I}}^*)$, the optimal regularized return of the harness, and the mismatch $M(\theta, \mathcal{I})$, the return the checkpoint leaves unrealized. Proposition 5.1 gives

$$J_{\mathcal{I}}(\Pi_{\theta,\mathcal{I}}) = C(\mathcal{I}) - M(\theta,\mathcal{I}), \qquad M(\theta,\mathcal{I}) = \frac{\tau}{1-\gamma}\mathbb{E}_{d_{\mathcal{I}}^{\Pi_{\theta,\mathcal{I}}}}\mathrm{KL}(\Pi_{\theta,\mathcal{I}}\|\Pi_{\mathcal{I}}^*). \tag{8}$$

The ceiling depends on the harness and the objective, and the mismatch depends on the harness and the checkpoint, including errors inherited from training. With the harness fixed, a weight update changes return only through M, and a harness change can move both terms. In the summit illustration of Figure 2, the step from ① to ② is a harness revision, and the step from ② to ③ is a weight update. For a frozen checkpoint, write J_j, C_j, and M_j for the quantities under interface j. Then

$$J_2 - J_1 = (C_2 - C_1) - (M_2 - M_1), \tag{9}$$

so a revision helps a frozen model only when its ceiling gain exceeds the added mismatch. In the presentation study, the ceiling is fixed, so each increase in mismatch appears as an equal loss in return. Table 1 reports both for each presentation.

5.3 When Adaptation Makes a Revision Beneficial

When a revision raises the ceiling but lowers the frozen model's return, adapting to it can realize the gain, since every unit of mismatch removed adds a unit of return. Write $J(\theta, \mathcal{I}) = J_{\mathcal{I}}(\Pi_{\theta,\mathcal{I}})$.

Corollary 5.2 (Adaptation after a harmful revision). *Let* θ_0 *and* θ_1 *be the initial and adapted checkpoints, and let* $\Delta C = C_2 - C_1$. *Suppose the revision is initially harmful,* $J(\theta_0, \mathcal{I}_2) < J(\theta_0, \mathcal{I}_1)$. *The adapted pair outperforms the initial pair,* $J(\theta_1, \mathcal{I}_2) > J(\theta_0, \mathcal{I}_1)$, *if and only if*

$$M(\theta_1, \mathcal{I}_2) < C_2 - J(\theta_0, \mathcal{I}_1) = \Delta C + M(\theta_0, \mathcal{I}_1). \tag{10}$$

The threshold $T = \Delta C + M(\theta_0, \mathcal{I}_1)$ is the new ceiling minus the initial pair's return. Three comparisons describe the outcome. *Recovery* compares the adapted pair with the initial pair, *harness ranking* compares the two harnesses at the adapted checkpoint, and *gain over continuation* compares adaptation with continued training under the source harness at the same budget, which tests whether the benefit comes from the revision or from further training. Appendix D.6 proves the corollary, and Appendix D.5 illustrates reversal and adaptation in a two-action example.

Batched clicks. From the H0-trained checkpoint, we create two branches with identical weights, update budgets, and task draws. One continues under H0, and the other adapts to H1, which adds batched clicks. Their returns stay comparable because both harnesses share the reward scale, discount, and regularization. H1 raises the ceiling slightly, but the frozen checkpoint carries its H0 preferences into H1 and adds far more mismatch, so return and task success fall. Adaptation under H1 removes most of this mismatch, and H0 continuation leaves it in place. As return under H1 rises and return under H0 falls, H1 overtakes H0 before the adapted pair passes the initial pair. The adapted pair finally beats both the initial pair and H0 continuation in task success and in return, where paired intervals exclude zero. Figure 4 shows both branches in its first panel, and Appendix F.3 gives the endpoint measurements.

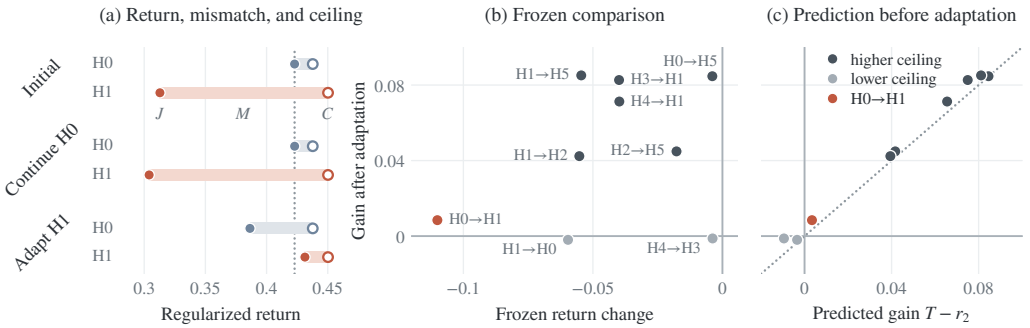

Figure 4: Harmful revisions become beneficial after adaptation when they raise the ceiling. (a) Return J, ceiling C, and mismatch M of the H0-trained checkpoint, its H0 continuation, and its H1 adaptation under each harness. The dotted line marks the initial pair's return. Gains of the adapted pair over the initial pair in the nine swaps that a frozen comparison rejects and that have $T > 0$, plotted against (b) the frozen return change and (c) the predicted gain $T - r_2$. In (b) and (c), dark points raise the ceiling, light points lower it, and the red point is H0→H1.

Across harness swaps. Batched clicks are one of many swaps among H0–H5, each starting from the checkpoint post-trained under the source harness. We adapt every swap that a frozen comparison rejects and whose T is positive, since no policy under the new harness can beat the initial pair when $T \leq 0$. All nine such swaps lose return when frozen. After adaptation, the seven that raise the ceiling all beat the initial pair, and the two that lower it gain nothing. Three of the seven switch to H5, which adds only instructions to H0. The frozen reference model also reads them and gives more weight to successful actions, which raises the ceiling.

The frozen loss says little about this final gain. H0→H5 and H4→H3 lose the same small amount when frozen, yet H0→H5 ties for the largest gain and H4→H3 gains nothing. H0→H1 has the largest frozen loss and still gains. The frozen return change, $T - M(\theta_0, \mathcal{I}_2)$, and the gain, $T - M(\theta_1, \mathcal{I}_2)$, differ only in the target mismatch. Before adaptation, this mismatch comes from the source harness and varies widely. Adaptation removes most of it, so the gain follows T, which differs across swaps mainly through the ceiling gain.

Ceiling and mismatch can therefore predict each gain before adaptation. If adaptation lowers the target mismatch to r_2, the mismatch that post-training leaves under the target harness, the gain is $T - r_2$. This prediction needs only the oracle ceilings and the six post-training runs of Section 3.2. Recorded before adaptation for every swap except H0→H1, it gets the sign of every gain right and nearly matches their order, while the frozen return change barely orders them. Figure 4 plots the gains against the frozen return change in its second panel and against $T - r_2$ in its third, and Appendix F.4 lists all nine swaps with their rank correlations.

Takeaways: What transfer costs

Deployed return is what the harness offers (the ceiling) minus what the policy leaves unrealized (the mismatch). A harness change moves both terms, and training moves only the mismatch. A frozen comparison therefore penalizes a new harness for mismatch that adaptation later removes. All seven rejected swaps that raise the ceiling outperform the initial pair after adaptation.

6 Measuring Mismatch in Deployment

The results so far use full action distributions and a solved optimum. A deployed agent exposes only sampled actions and may have no known optimum, so we estimate mismatch from action samples, first with an oracle target and then with a teacher.

6.1 Estimating Mismatch from Action Samples

Measuring mismatch requires a target policy and action samples that the deployed model draws through its unchanged interface at states on its own trajectories, since mismatch weights each decision by how often the model reaches it.

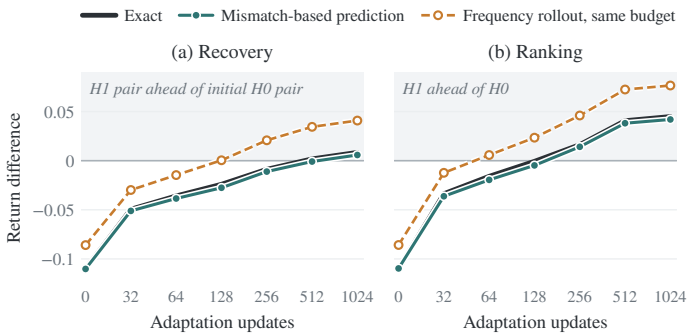

Figure 5: Finite measurements predict recovery and harness ranking separately. (a) Return of the H1-adapted checkpoint under H1 minus the initial H0 pair's return. (b) Return under H1 minus return under H0 for the same checkpoint.

Pilot rollouts supply these states. Stopping each rollout at a random time t drawn with probability $(1-\gamma)\gamma^t$ gives an exact draw from discounted occupancy. At each of m such states, we sample N actions through the deployed interface and smooth the counts into an action distribution $\widetilde{p}(\cdot \mid s)$. The mismatch estimate averages the KL divergence over the sampled states:

$$\widehat{M} = \frac{\tau}{(1-\gamma)m} \sum_{i=1}^{m} \mathrm{KL}(\widetilde{p}(\cdot \mid s_i) \| \Pi^*_{\mathcal{I}}(\cdot \mid s_i)) \,. \tag{11}$$

Log odds between two actions, $\tau \log[\Pi_{\theta,\mathcal{I}}(a \mid s)/\Pi_{\theta,\mathcal{I}}(b \mid s)]$, also measure relative scores and need only the probabilities of the queried actions. Pairs that connect all actions determine the centered score error, which bounds the mismatch.

Proposition 6.1 (From score error to mismatch)**.** *Let* $x_s = \widehat{F}_{\theta,\mathcal{I}}(s,\cdot) - \bar{F}^*_{\mathcal{I}}(s,\cdot)$ *and* $\varepsilon_d^2 = \mathbb{E}_{d_{\mathcal{I}}^{\Pi_{\theta,\mathcal{I}}}} \|x_s\|_2^2$*. For positive semantic action probabilities,*

$$M(\theta,\mathcal{I}) \le \frac{\varepsilon_d^2}{4\tau(1-\gamma)}. \tag{12}$$

Once this bound falls below the threshold in Corollary 5.2, the adapted pair is guaranteed to outperform the initial pair. Appendix E gives the proof and a lower bound on mismatch.

6.2 Finite Measurements Predict Return Changes

We test whether finite measurements predict return changes on tasks and configurations that played no role in calibration. For each configuration, an oracle supplies the ceiling and the optimal action distributions, and the predictor estimates a candidate's return change relative to its baseline as $\widehat{\Delta J} = \Delta C - \Delta \widehat{M}$. Exact returns serve as ground truth.

We first deploy the final H0-trained and H1-adapted checkpoints through H0's actions under held-out configurations that vary click reliability, click cost, and presentation, and compare each configuration with its baseline. The ceiling alone fails because presentation changes move mismatch and leave the ceiling fixed. Subtracting measured mismatch reduces the prediction error about a hundredfold and predicts the direction of every change not close to zero. A frequency rollout, which estimates action probabilities from frequencies at the same query budget, is similarly accurate here.

At each saved H1-adapted checkpoint, we then predict recovery and ranking, which are differences of a few hundredths in return. Here the mismatch-based prediction is about ten times more accurate than the frequency rollout. The predicted ranking reverses at the same checkpoint as the observed one, and the predicted recovery comes one saved checkpoint later. The frequency rollout overestimates both differences and predicts the reversal too early. Figure 5 shows both predictions, and Appendices F.5 and F.6 give the sampling budget and all numerical results.

6.3 Ceiling and Mismatch of Coding-Agent Harnesses

We deploy a frozen Qwen3.5-9B through the native configurations of five coding-agent harnesses: DeepAgents, Pi coding-agent, DeepSeek Harness, Terminus-2, and mini-SWE-agent. With the model fixed, the harness moves Terminal-Bench Lite accuracy from 47.9% to 64.1%. We ask whether the ceiling and the mismatch explain this ordering.

Teacher-based estimates. Coding agents have open action spaces and no solvable optimum, so a teacher model, GPT-5.6-luna, takes its place. On three Terminal-Bench Lite tasks, we record every model decision, sample outputs from the target and the teacher through the unchanged harness, and map each output to an action category shared by all harnesses. The reference is the target's own category distribution, so the target's regularized return equals its success probability, and so does the exact ceiling minus mismatch. We estimate the ceiling by the teacher's regularized return, $\widetilde{C}$, and the mismatch by Proposition 6.1 with the teacher's log odds in place of the optimal scores, summed along the target's trajectories, $\widetilde{M}$. The teacher need not be optimal, so we test whether the estimates order the harnesses by accuracy. Appendix F.7 gives the protocol and estimators.

Table 2: Ceiling and mismatch of five coding-agent harnesses for a frozen Qwen3.5-9B. $\widetilde{C}$ and $\widetilde{M}$ are teacher-based estimates. The last row gives each column's Spearman correlation with accuracy.

Harness	$\widetilde{C}$	$\widetilde{M}$	$\widetilde{C}-\widetilde{M}$	Accuracy (%)
Terminus-2	0.431	0.134	0.297	64.1 ± 5.1
mini-SWE-agent	0.631	0.335	0.296	59.8 ± 7.8
DeepSeek Harness	0.590	0.382	0.208	54.7 ± 3.9
Pi coding-agent	0.383	0.206	0.177	51.3 ± 2.6
DeepAgents	0.261	0.429	-0.168	47.9 ± 3.9
Spearman ρ	0.7	-0.7	1.0	

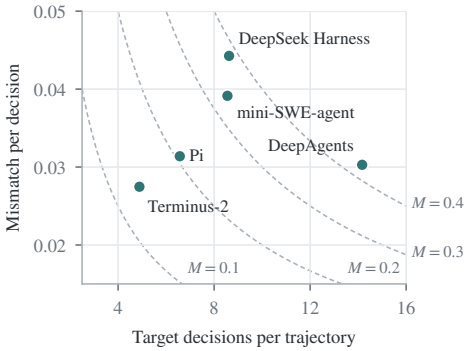

Figure 6: Mismatch accumulates over decisions. Each harness's mismatch splits into the number of target decisions per trajectory and the mismatch per decision. Dashed curves have constant $\widetilde{M}$.

$\widetilde{C}-\widetilde{M}$ orders the harnesses. Computed without the target's task outcomes, $\widetilde{C}-\widetilde{M}$ orders all five harnesses as accuracy does. Either term alone misorders them. The ceiling estimate alone ranks mini-SWE-agent first, and the mismatch estimate alone ranks Pi coding-agent second, while their difference ranks both correctly. Exact terms show the same pattern. When a frozen Qwen3.5-9B serves as its own reference under H0–H5 on MiniWoB++, the ceiling alone ranks the most successful harness last, and the mismatch alone ranks H0 below two less successful harnesses. Terminus-2 has a middle ceiling and leads through the smallest mismatch. DeepSeek Harness and mini-SWE-agent have the two highest ceilings and large mismatch, the profile of the batched-click harness before adaptation, so they are natural candidates for adapting the model. Table 2 lists the estimates, and Appendix F.8 gives the exact comparison.

Where mismatch comes from. By Proposition 5.1, mismatch accumulates along the model's own trajectories, so it grows with the number of decisions a harness asks of the model. DeepAgents has a mismatch per decision close to that of Terminus-2, but the target makes nearly three times as many model calls under it and accumulates the largest mismatch. DeepSeek Harness has the largest mismatch per decision. Under it, the teacher plans much more often than the target, and the target inspects the environment much more often. Figure 6 separates the two factors for each harness.

7 Discussion and Conclusion

A policy trained under a harness learns the harness's relative action scores and expresses them through its interface. Post-training makes these preferences harness-specific, and a new interface can change the model's choices even when every optimal decision stays the same. Because deployed return is the ceiling minus the mismatch, a harness that offers more can hurt a trained model and become the best choice after adaptation. Mismatch estimated from sampled actions predicts these changes when the optimum is known, and for coding agents, teacher-based estimates of ceiling minus mismatch order harnesses by accuracy. Harness evaluations should therefore report the ceiling next to frozen return, cross checkpoints with harnesses, and check how presentation changes move the model's choices. Harness distillation and joint adaptation of harnesses and weights (Ye et al., 2026; Chen et al., 2026; Kim et al., 2026) can use sampled-action measurements to track the ceiling and the mismatch separately. Appendix A discusses related work.

Reproducibility Statement

Section 2 and Appendix B state the control setting and the assumptions of all theoretical results. Appendices C, D, and E give complete proofs of all propositions and corollaries. Appendix F specifies each experiment, including task generation and splits, harness definitions, compiled task graphs, model and training settings with seeds, oracle computations, measurement estimators and their calibration, bootstrap procedures, and hardware. For the coding-agent study, Appendix F.7 gives the harness configurations, model settings, probe tasks, action categories, and estimators.

Ethics Statement

This work studies how the harness and the model's training together determine the performance of language-model agents. It involves no human subjects and no personal data. All experiments use public benchmarks (MiniWoB++ and Terminal-Bench Lite) and publicly available models. The coding agents run all file and shell operations inside the benchmark's task containers. Our measurements help practitioners choose a harness for a given model and decide when adapting the model to a new harness is worthwhile.

AI Use Statement

We used generative AI tools only to edit the text of this paper for clarity and readability and to refine the visual style of its figures. The research ideas, theory and proofs, experimental design, implementation, and analysis of results are our own work. We did not use generative AI tools for any task whose disclosure is required by the ICLR 2027 AI Policy for Authors. We reviewed all AI-assisted text and figures, checked that the edited text keeps its intended meaning, and checked that every figure shows the recorded results. We take responsibility for the final content of this work.

References

Huan ang Gao, Jiayi Geng, Wenyue Hua, Mengkang Hu, Xinzhe Juan, Hongzhang Liu, Shilong Liu, Jiahao Qiu, Xuan Qi, Yiran Wu, Hongru Wang, Han Xiao, Yuhang Zhou, Shaokun Zhang, Jiayi Zhang, Jinyu Xiang, Yixiong Fang, Qiwen Zhao, Dongrui Liu, Qihan Ren, Cheng Qian, Zhenhailong Wang, Minda Hu, Huazheng Wang, Qingyun Wu, Heng Ji, and Mengdi Wang. A survey of self-evolving agents: What, when, how, and where to evolve on the path to artificial super intelligence. Transactions on Machine Learning Research, 2026. URL https://arxiv.org/abs/2507.21046.

Haoyang Cao, Samuel N. Cohen, and Lukasz Szpruch. Identifiability in inverse reinforcement learning. In Advances in Neural Information Processing Systems, volume 34, pp. 12362–12373, 2021. URL https://arxiv.org/abs/2106.03498.

Zhengyu Chen, Teng Xiao, Huaisheng Zhu, Yige Yuan, Luan Zhang, and Jingang Wang. Co-Harness: Co-evolving harnesses and model weights for LLM agents. arXiv preprint arXiv:2607.22688, 2026. URL https://arxiv.org/abs/2607.22688.

Jinyuan Fang, Yanwen Peng, Xi Zhang, Yingxu Wang, Xinhao Yi, Guibin Zhang, Yi Xu, Bin Wu, Siwei Liu, Zihao Li, Zhaochun Ren, Nikos Aletras, Xi Wang, Han Zhou, and Zaiqiao Meng. A comprehensive survey of self-evolving ai agents: A new paradigm bridging foundation models and lifelong agentic systems, 2025. URL https://arxiv.org/abs/2508.07407.

Chrisantha Fernando, Dylan Banarse, Henryk Michalewski, Simon Osindero, and Tim Rocktäschel. Promptbreeder: Self-referential self-improvement via prompt evolution, 2023. URL https://arxiv.org/abs/2309.16797.

Matthieu Geist, Bruno Scherrer, and Olivier Pietquin. A theory of regularized Markov decision processes. In Proceedings of the 36th International Conference on Machine Learning, volume 97 of Proceedings of Machine Learning Research, pp. 2160–2169. PMLR, 2019. URL https://proceedings.mlr.press/v97/geist19a.html.

Adam Gleave, Michael Dennis, Shane Legg, Stuart Russell, and Jan Leike. Quantifying differences in reward functions, 2021. URL https://arxiv.org/abs/2006.13900.

Tuomas Haarnoja, Haoran Tang, Pieter Abbeel, and Sergey Levine. Reinforcement learning with deep energy-based policies, 2017. URL https://arxiv.org/abs/1702.08165.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor, 2018. URL https://arxiv.org/abs/1801.01290.

Prannay Hebbar, Yogendra Manawat, Samuel Verboomen, Alesia Ivanova, Selvam Palanimalai, Kunal Bhatia, and Vignesh Baskaran. SIA: Self improving AI with harness & weight updates. arXiv preprint arXiv:2605.27276, 2026. URL https://arxiv.org/abs/2605.27276.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In International Conference on Learning Representations, 2022. URL https://openreview.net/forum?id=nZeVKeeFYf9.

Shengran Hu, Cong Lu, and Jeff Clune. Automated design of agentic systems. In International Conference on Learning Representations, 2025. URL https://arxiv.org/abs/2408.08435.

Zixuan Ke, Austin Xu, Yifei Ming, Xuan-Phi Nguyen, Caiming Xiong, and Shafiq Joty. MAS-ZERO: Designing multi-agent systems with zero supervision. In NeurIPS 2025 Workshop on Scaling Environments for Agents, 2025. URL https://openreview.net/forum?id=j5VXzWyoyW.

Haechan Kim, Yoonho Lee, Gisang Lee, Chelsea Finn, and Kangwook Lee. WHALE: A simple recipe for joint harness-weight optimization. arXiv preprint arXiv:2609.00196, 2026. URL https://arxiv.org/abs/2609.00196.

Sergey Levine. Reinforcement learning and control as probabilistic inference: Tutorial and review. arXiv preprint arXiv:1805.00909, 2018. URL https://arxiv.org/abs/1805.00909.

Evan Zheran Liu, Kelvin Guu, Panupong Pasupat, Tianlin Shi, and Percy Liang. Reinforcement learning on web interfaces using workflow-guided exploration. In International Conference on Learning Representations, 2018. URL https://arxiv.org/abs/1802.08802.

Jian Mu, Chaoyun Zhang, Chiming Ni, Lu Wang, Bo Qiao, Kartik Mathur, Qianhui Wu, Yuhang Xie, Xiaojun Ma, Mengyu Zhou, Si Qin, Liqun Li, Yu Kang, Minghua Ma, Qingwei Lin, Saravan Rajmohan, and Dongmei Zhang. GUI-360°: A comprehensive dataset and benchmark for computer-using agents. arXiv preprint arXiv:2511.04307, 2025. URL https://arxiv.org/abs/2511.04307.

Andrew Y. Ng and Stuart J. Russell. Algorithms for inverse reinforcement learning. In Proceedings of the Seventeenth International Conference on Machine Learning (ICML 2000), pp. 663–670. Morgan Kaufmann, 2000.

Andrew Y. Ng, Daishi Harada, and Stuart J. Russell. Policy invariance under reward transformations: Theory and application to reward shaping. In Proceedings of the Sixteenth International Conference on Machine Learning (ICML 1999), pp. 278–287. Morgan Kaufmann, 1999.

Brendan O'Donoghue. On the connection between Bregman divergence and value in regularized Markov decision processes. arXiv preprint arXiv:2210.12160, 2022. URL https://arxiv.org/abs/2210.12160.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Stefano Ermon, Christopher D. Manning, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. In Advances in Neural Information Processing Systems, 2023. URL https://arxiv.org/abs/2305.18290.

Rafael Rafailov, Joey Hejna, Ryan Park, and Chelsea Finn. From r to Q^*: Your language model is secretly a Q-function. In Conference on Language Modeling, 2024. URL https://arxiv.org/abs/2404.12358.

Maxime Robeyns, Martin Szummer, and Laurence Aitchison. A self-improving coding agent, 2025. URL https://arxiv.org/abs/2504.15228.

Stuart Russell. Learning Agents for Uncertain Environments (Extended Abstract). In Annual Conference on Computational Learning Theory, pp. 101–103, 1998. doi: 10.1145/279943.279964. URL https://mlanthology.org/colt/1998/russell1998colt-learning/.

Andreas Schlaginhaufen and Maryam Kamgarpour. Towards the transferability of rewards recovered via regularized inverse reinforcement learning. In Advances in Neural Information Processing Systems (NeurIPS), volume 37, pp. 21461–21501, 2024. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/2628d4d3b054c2d7ad33ab03435204f4-Abstract-Conference.html.

John Schulman, Xi Chen, and Pieter Abbeel. Equivalence between policy gradients and soft Q-learning. arXiv preprint arXiv:1704.06440, 2017. URL https://arxiv.org/abs/1704.06440.

Melanie Sclar, Yejin Choi, Yulia Tsvetkov, and Alane Suhr. Quantifying language models' sensitivity to spurious features in prompt design or: How I learned to start worrying about prompt formatting. In International Conference on Learning Representations, 2024. URL https://arxiv.org/abs/2310.11324.

Joar Skalse, Matthew Farrugia-Roberts, Stuart Russell, Alessandro Abate, and Adam Gleave. Invariance in policy optimisation and partial identifiability in reward learning. In Proceedings of the 40th International Conference on Machine Learning (ICML), 2023. URL https://proceedings.mlr.press/v202/skalse23a.html.

Charlie Snell, Dan Klein, and Ruiqi Zhong. Learning by distilling context. arXiv preprint arXiv:2209.15189, 2022. URL https://arxiv.org/abs/2209.15189.

Gokul Swamy, Sanjiban Choudhury, Wen Sun, Zhiwei Steven Wu, and J. Andrew Bagnell. All roads lead to likelihood: The value of reinforcement learning in fine-tuning. arXiv preprint arXiv:2503.01067, 2025. URL https://arxiv.org/abs/2503.01067.

Hongliang Wei, Xiaobing Tu, Yinggui Wang, Zhengxi Liu, Rongkun Xue, Jinkui Ren, Xiantao Zhang, Debin Zhao, and Xiaopeng Fan. HarnessBandit: Joint learnability-transferability scheduling for multi-harness agentic reinforcement learning. arXiv preprint arXiv:2609.13739, 2026. URL https://arxiv.org/abs/2609.13739.

Georg Wölflein, Dyke Ferber, Daniel Truhn, Ognjen Arandjelovic, and Jakob Nikolas Kather. LLM agents making agent tools. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pp. 26092–26130, Vienna, Austria, July 2025. Association for Computational Linguistics. doi: 10.18653/v1/2025.acl-long.1266. URL https://aclanthology.org/2025.acl-long.1266/.

John Yang, Carlos E. Jimenez, Alexander Wettig, Kilian Lieret, Shunyu Yao, Karthik Narasimhan, and Ofir Press. SWE-agent: Agent-computer interfaces enable automated software engineering. In Advances in Neural Information Processing Systems, 2024. URL https://arxiv.org/abs/2405.15793.

Haoran Ye, Yuxing Lu, Haonan Dong, Zhaochen Su, and Guojie Song. Harness-Zero: Harness distillation via agent-as-harness. arXiv preprint arXiv:2609.24974, 2026. URL https://arxiv.org/abs/2609.24974.

Zhou Yu, Bin Bi, Shiva Kumar Pentyala, Shubham Mehrotra, Sougata Chaudhuri, Shilpa Bhagavath, Zeyuan Chen, Ran Xu, Phil Mui, James Zhu, and Sitaram Asur. Co-evolving harnesses and models: On-policy correction helps weaker models catch up where imitation fails. arXiv preprint arXiv:2609.09134, 2026. URL https://arxiv.org/abs/2609.09134.

Chaoyun Zhang, He Huang, Chiming Ni, Jian Mu, Si Qin, Shilin He, Lu Wang, Fangkai Yang, Pu Zhao, Chao Du, Liqun Li, Yu Kang, Zhao Jiang, Suzhen Zheng, Rujia Wang, Jiaxu Qian, Minghua Ma, Jian-Guang Lou, Qingwei Lin, Saravan Rajmohan, and Dongmei Zhang. UFO2: The desktop agentOS. arXiv preprint arXiv:2504.14603, 2025a. URL https://arxiv.org/abs/2504.14603.

Jenny Zhang, Shengran Hu, Cong Lu, Robert Lange, and Jeff Clune. Darwin gödel machine: Open-ended evolution of self-improving agents. In The International Conference on Learning Representations (ICLR), 2026. URL https://iclr.cc/virtual/2026/poster/10007327.

Jiayi Zhang, Jinyu Xiang, Zhaoyang Yu, Fengwei Teng, Xionghui Chen, Jiaqi Chen, Mingchen Zhuge, Xin Cheng, Sirui Hong, Jinlin Wang, Bingnan Zheng, Bang Liu, Yuyu Luo, and Chenglin Wu. AFlow: Automating agentic workflow generation. In International Conference on Learning Representations, 2025b. URL https://arxiv.org/abs/2410.10762.

A Related Work

Regularized control and decision geometry. Control as inference and RL with a maximum entropy objective express optimal policies through rewards, action values, and reference distributions (Levine, 2018; Haarnoja et al., 2017; 2018). Regularized control supplies the Gibbs policy and the geometry of the value gap (Geist et al., 2019; O'Donoghue, 2022). The relation between policy gradients and soft Q-learning provides another view of these coordinates (Schulman et al., 2017). DPO and its sequential extension relate language-model policies to rewards and action values during post-training (Rafailov et al., 2023; 2024).

Our optimal action score combines the optimal semantic action value with the semantic reference prior. These coordinates let us compare the decisions induced by different interfaces and measure a deployed policy's gap to each optimum. The analysis of fine-tuning objectives by Swamy et al. (2025) likewise uses a common geometric description to distinguish ideal equivalence from differences that emerge in practice. Here the distinction concerns the interface's optimal decisions and the choices that a particular checkpoint makes through it.

Behavioral identification and reward transfer. Inverse RL recovers rewards from behavior subject to ambiguities (Ng & Russell, 2000; Russell, 1998). Reward shaping using a potential function preserves optimal policies (Ng et al., 1999). Analyses of identifiability under entropy regularization and of partial identifiability characterize the corresponding reward classes (Cao et al., 2021; Skalse et al., 2023). EPIC compares reward functions while accounting for transformations that preserve policies (Gleave et al., 2021). Work on the transferability of rewards recovered through regularized inverse RL further studies changes in transition laws (Schlaginhaufen & Kamgarpour, 2024).

Our equivalence class for a single interface removes statewise constants from the scores. Appendix C distinguishes this behavioral object from the nonunique decomposition into prior, reward, and transition. The transfer analysis then compares the target energy with the actual deployed readout, weighting their discrepancy by deployment occupancy.

Interface and agent design. Interface design and prompt formatting affect agent behavior (Yang et al., 2024; Sclar et al., 2024; Zhang et al., 2025a). GUI-360° reports higher action prediction accuracy for GPT-4o and lower accuracy for fine-tuned Qwen after adding accessibility metadata and replacing coordinate arguments with element IDs (Mu et al., 2025). ADAS and AFlow search over agent implementations and workflows (Hu et al., 2025; Zhang et al., 2025b). Darwin Gödel Machine and Self-Improving Coding Agent revise scaffolds and assess them through execution (Zhang et al., 2026; Robeyns et al., 2025). Promptbreeder, ToolMaker, and MAS-ZERO respectively explore prompts, tools, and designs involving multiple agents (Fernando et al., 2023; Wölflein et al., 2025; Ke et al., 2025). Surveys place these approaches within agent systems with mechanisms for self improvement (Fang et al., 2025; ang Gao et al., 2026).

These design procedures can change action opportunities, realizations, or both. Our analysis separates each candidate's optimal return from the evaluated checkpoint's gap to that optimum. This distinction connects execution feedback to two possible sources of improvement: a more useful decision problem and a policy that better realizes its available decisions.

Context and harness distillation. Context distillation trains a model to reproduce, without the prompt, the behavior that the prompt elicits (Snell et al., 2022). Harness-Zero distills a specialized harness into the weights by letting a harnessing agent correct the student's responses in the target harness's action space (Ye et al., 2026). Our identification result states which decision information such training can carry, the relative action scores of the training harness. The readout study shows that the learned preferences are still expressed through the deployment interface, so a distilled policy can change its choices when the interface changes.

Joint adaptation of harnesses and model weights. Joint adaptation takes several forms: SIA selects harness and weight updates, Co-Harness trains from trajectories collected under revised interfaces, and WHALE alternates training with harness search (Hebbar et al., 2026; Chen et al., 2026; Kim et al., 2026). Yu et al. (2026) find that imitating expert trajectories can hurt a weaker model under an evolved harness while on-policy correction helps, and HarnessBandit schedules training across harnesses by learnability and transferability (Wei et al., 2026).

These methods can use the ceiling and the mismatch to tell whether a candidate harness offers too little or is not yet used well. Our adaptation threshold specifies the mismatch reduction needed to make a revision with a higher ceiling beneficial. Eq. (24) isolates how a weight update changes a revision's relative value. Comparisons across checkpoints and interfaces therefore track both recovery after a revision and changes in interface rankings. The controlled protocol measures this interaction separately from gains under either interface alone.

B Semantic Control: Objects, Abstraction, and Objectives

B.1 Notation and Standing Conditions

Table 3: Notation. Energies and vector operations are statewise unless an expectation is shown.

Symbol	Meaning
$\mathcal{I} = (c_{\mathcal{I}}, d_{\mathcal{I}}, e_{\mathcal{I}})$	Context encoder, semantic decoder, and fixed execution protocol.
$\pi_\theta,\ \Pi_{\theta,\mathcal{I}}$	Policy over token sequences and its executed semantic pushforward.
$P_{\mathcal{I}}, r_{\mathcal{I}}, \pi_{\mathcal{I}}^{\mathrm{ref}}$	Induced dynamics, task reward including execution costs, and semantic reference prior.
ρ_0, γ, τ	Initial state distribution, discount, and KL regularization coefficient.
$F_{\mathcal{I}}^*,\ \bar{F}_{\mathcal{I}}^*$	Optimal action score and its centered representative.
$\widehat{F}_{\theta,\mathcal{I}}$	Readout, the centered action scores $\tau P_\perp \log \Pi_{\theta,\mathcal{I}}$.
$d_{\mathcal{I}}^{\Pi}$	Normalized discounted state occupancy of the evaluated policy.
$C(\mathcal{I}), M(\theta, \mathcal{I})$	Ceiling (optimal regularized return) and mismatch.
e_1, g_{12}, η_{12}	Training residual, target shift, and readout drift.
Φ, D_Φ	Log partition function and its Bregman divergence.

The formal results use finite state and action spaces, bounded rewards, $0 \le \gamma < 1$, $\tau > 0$, and a reference prior that is positive on every available action. The initial distribution, reward units, and regularization coefficient are shared in a comparison across interfaces. State/action alignments are fixed before the comparison. Results involving $\log \Pi_{\theta,\mathcal{I}}$ additionally require positive probabilities for the actual policy on the probed actions. The gap identity for a fixed interface itself allows zero policy probabilities, with $0 \log 0 = 0$.

We use a reference checkpoint distinct from the evolving evaluated checkpoint. When its sequence distribution is π_{ref}, a natural semantic reference is

$$\pi_{\mathcal{I}}^{\mathrm{ref}}(a \mid s) = \sum_{y : d_{\mathcal{I}}(y;s) = a} \pi_{\mathrm{ref}}(y \mid c_{\mathcal{I}}(s)). \tag{13}$$

Consequently a new encoding can change both the target induced by the reference and the evaluated model's readout. Experiments that hold the target fixed must verify or explicitly control this reference.

B.2 Decision Epochs and Controller State

A semantic state includes every variable needed to predict the state at the next decision: the task state, visible history or a sufficient summary, tool results, budget counters, and controller memory. An action denotes a fully specified operation, such as invoking a particular tool with particular arguments, or submitting a particular candidate. Two output strings belong to one semantic action only when the decoder and executor execute the same operation at that state.

After a semantic action is selected, $e_{\mathcal{I}}$ can execute a fixed tool, perform a deterministic check, or apply a stochastic transition independent of the policy. Its expected costs enter $r_{\mathcal{I}}$, and its outcomes enter $P_{\mathcal{I}}$. If a retry, verifier, or stopping rule calls the evaluated model for another decision, that call is another decision epoch in the expanded state space. This keeps the transition kernel fixed while the policy varies.

An invalid string is mapped to an explicit failure or recovery action. Similarly, when an empirical action vocabulary groups outputs into an "other" category, that category must have a defined execution interpretation. Discarding unmodeled outputs changes the policy being measured.

B.3 When Training over Tokens Induces Semantic Optimality

Let $Y_a(s)$ be the set of complete output realizations decoded as action a. Write a sequence policy and reference as

$$p(y \mid s) = p_A(a \mid s)p(y \mid s, a), \qquad \rho(y \mid s) = \rho_A(a \mid s)\rho(y \mid s, a), \quad y \in Y_a(s).$$

For any such distributions, the KL chain rule gives

$$\mathrm{KL}(p(\cdot \mid s)\|\rho(\cdot \mid s)) = \mathrm{KL}(p_A(\cdot \mid s)\|\rho_A(\cdot \mid s)) + \mathbb{E}_{a\sim p_A}\mathrm{KL}(p(\cdot \mid s, a)\|\rho(\cdot \mid s, a)). \tag{14}$$

Suppose reward, transition, and discount are identical within each fiber. Then the return from choosing y depends only on a, while the second term in Eq. (14) is minimized by retaining the reference conditional distribution inside that fiber. Optimization over complete sequence actions therefore reduces exactly to the semantic control problem regularized by KL divergence relative to ρ_A.

For an autoregressive model, probabilities of complete sequences include termination. The expected sum of token KL divergences equals sequence KL when the same terminating sequence measure and matching token conditionals are used. Rewards that depend on tokens, discounts that depend on sequence length, or additional model decisions within a proposed temporally extended action require the expanded model above.

C Behavioral Identification and Mechanism Ambiguity

C.1 Soft Bellman Optimality

For a fixed interface, the optimal Bellman operator is

$$(\mathcal{T}V)(s) = \tau \log \sum_a \pi^{\mathrm{ref}}(a \mid s) \exp\left(\frac{r(s,a) + \gamma \mathbb{E}_{P(\cdot|s,a)}V(s')}{\tau}\right).$$

Monotonicity of the log partition function and its translation identity imply $\|\mathcal{T}V - \mathcal{T}W\|_\infty \le \gamma \|V - W\|_\infty$. Thus it has a unique fixed point V^*, and $Q^* = r + \gamma P V^*$ is uniquely determined. At each state, maximizing $\sum_a \pi_a Q^*_a - \tau \sum_a \pi_a \log(\pi_a/\rho_a)$ over the simplex gives

$$\pi^*_a = \frac{\rho_a e^{Q^*_a/\tau}}{\sum_b \rho_b e^{Q^*_b/\tau}}.$$

Strict concavity gives uniqueness. At the fixed point, $V^*(s) = \tau \log \sum_b \rho_b e^{Q^*_b/\tau}$, so the optimal policy takes the Gibbs form

$$\Pi^*(a \mid s) = \pi^{\mathrm{ref}}(a \mid s) \exp\left(\frac{Q^*(s,a) - V^*(s)}{\tau}\right). \tag{15}$$

Substituting $F^*_a = Q^*_a + \tau \log \rho_a$ produces the softmax representation in the main text.

C.2 Proof of Proposition 3.1

If $F(s,\cdot) = G(s,\cdot) + c(s)\mathbf{1}$, the factor $e^{c(s)/\tau}$ cancels in softmax. Conversely, equal softmax probabilities imply

$$F(s,a) - \Phi(F(s,\cdot)) = G(s,a) - \Phi(G(s,\cdot))$$

for every action. Their difference is therefore constant across actions. Applying this equivalence to optimal action scores proves the first assertion. Under $\Pi_{\theta,\mathcal{I}_{\mathrm{tr}}} = \Pi^*_{\mathcal{I}_{\mathrm{tr}}}$, the Gibbs identity gives

$$\tau \log \Pi_{\theta,\mathcal{I}_{\mathrm{tr}}}(a \mid s) = F^*_{\mathcal{I}_{\mathrm{tr}}}(s,a) - V^*_{\mathcal{I}_{\mathrm{tr}}}(s).$$

Projection onto $\mathbf{1}^\perp$, or subtraction between two actions, removes $V^*_{\mathcal{I}_{\mathrm{tr}}}(s)$ and proves Eq. (4). □

C.3 Mechanism Nonidentifiability

Proposition C.1 (Mechanism nonidentifiability)**.** *A fixed optimal score field can be realized by a continuous family of different reference priors and rewards, and even by different transition laws. Consequently, optimal behavior alone does not uniquely determine the interface mechanism.*

Proof. Fix a finite score field F, any reference ρ' with full support, and any stochastic kernel P'. Define

$$V(s) = \Phi(F(s,\cdot)), \qquad Q'(s,a) = F(s,a) - \tau \log \rho'(a \mid s), \qquad r' = Q' - \gamma P' V.$$

Then $Q' = r' + \gamma P' V$, and

$$\tau \log \sum_a \rho'(a \mid s) e^{Q'(s,a)/\tau} = \tau \log \sum_a e^{F(s,a)/\tau} = V(s).$$

By uniqueness of the Bellman fixed point, these are the optimal values of the constructed control problem. Its optimal policy is $\mathrm{softmax}(F/\tau)$. Varying ρ' or P' gives a continuous family of mechanisms with the same scores and policy. □

D Performance Geometry and Readout Across Interfaces

D.1 Proof of Proposition 5.1

Fix an interface and suppress its subscript. For a policy Π, write

$$(\mathcal{T}^{\Pi} V)(s) = \mathbb{E}_{a \sim \Pi(\cdot|s)} \left[r(s,a) + \gamma P V(s,a) - \tau \log \frac{\Pi(a \mid s)}{\pi^{\mathrm{ref}}(a \mid s)} \right].$$

Rearranging the optimal-policy expression in Eq. (15) gives $Q^* - V^* = \tau \log(\Pi^*/\pi^{\mathrm{ref}})$. Substituting this relation into the regularized Bellman update yields the statewise gap:

$$V^*(s) - (\mathcal{T}^{\Pi} V^*)(s) = \tau \mathrm{KL}(\Pi(\cdot \mid s) \| \Pi^*(\cdot \mid s)) =: k_{\Pi}(s). \tag{16}$$

Subtracting the policy Bellman equation $V^{\Pi} = \mathcal{T}^{\Pi} V^{\Pi}$ yields

$$V^* - V^{\Pi} = k_{\Pi} + \gamma P^{\Pi}(V^* - V^{\Pi}) = (I - \gamma P^{\Pi})^{-1} k_{\Pi}.$$

The normalized occupancy satisfies $(d^{\Pi})^{\top} = (1-\gamma)\rho_0^{\top}(I - \gamma P^{\Pi})^{-1}$. Multiplication by $\rho_0^{\top}$ proves Eq. (6), in which both the KL direction and the occupancy belong to the evaluated policy. □

D.2 Bregman Representation of the Decision Loss

For $\Phi(F) = \tau \log \sum_a e^{F_a/\tau}$, the following identity relates score differences to the policy divergence.

Lemma D.1 (Bregman–KL identity)**.** *For any score vectors F and G, the Bregman divergence generated by Φ satisfies*

$$D_{\Phi}(F,G) = \Phi(F) - \Phi(G) - \langle \nabla\Phi(G), F - G \rangle = \tau \mathrm{KL}\big(\mathrm{softmax}(G/\tau) \| \, \mathrm{softmax}(F/\tau)\big). \tag{17}$$

Proof. Let $p_F = \mathrm{softmax}(F/\tau)$, so that $\nabla\Phi(F) = p_F$ and $\tau \log p_F = F - \Phi(F)\mathbf{1}$. Then

$$\begin{aligned} \tau \mathrm{KL}(p_G \| p_F) &= \sum_a (p_G)_a \big[G_a - \Phi(G) - F_a + \Phi(F) \big] \\ &= \Phi(F) - \Phi(G) - \langle p_G, F - G \rangle = D_{\Phi}(F,G). \end{aligned}$$

For any constants c, d, $\Phi(F + c\mathbf{1}) = \Phi(F) + c$ and $\nabla\Phi(G + d\mathbf{1}) = \nabla\Phi(G)$. These identities show $D_{\Phi}(F + c\mathbf{1}, G + d\mathbf{1}) = D_{\Phi}(F,G)$. For a positive policy, let $F_{\Pi} = \tau P_{\perp} \log \Pi$. Substitution in Proposition 5.1 gives the performance gap in centered scores:

$$J_{\mathcal{I}}(\Pi^*_{\mathcal{I}}) - J_{\mathcal{I}}(\Pi) = \frac{1}{1-\gamma} \mathbb{E}_{s \sim d^{\Pi}_{\mathcal{I}}} D_{\Phi}(F^*_{\mathcal{I}}, F_{\Pi}). \tag{18}$$

For $\Pi = \Pi_{\theta,\mathcal{I}}$, its centered score vector is $\widehat{F}_{\theta,\mathcal{I}}$, giving the score form of mismatch through Eq. (7).

D.3 General Transfer and the Case with Zero Drift

Eq. (5) follows by adding and subtracting the source target and source readout in the same centered coordinates. Substitution into the exact gap gives

$$M(\theta, 2) = \frac{1}{1-\gamma}\mathbb{E}_{d_2^{\Pi_{\theta,2}}} D_\Phi\big(\bar{F}_2^*, \bar{F}_1^* + e_1 + \eta_{12}\big). \tag{19}$$

For an optimal source policy whose readout is preserved by the swap, $e_1 = \eta_{12} = 0$, so

$$M(\theta, 2) = \frac{1}{1-\gamma}\mathbb{E}_{d_2^{\Pi_{\theta,2}}} D_\Phi(F_2^*, F_1^*) \qquad (e_1 = 0,\ \eta_{12} = 0). \tag{20}$$

The model continues to express the source preferences, whose discrepancy from the new optimum determines the deployment gap. For a quantitative perturbation bound, define $\varepsilon_g, \varepsilon_e, \varepsilon_\eta$ as the respective root mean square norms of g_{12}, e_1, η_{12} under the same occupancy of the deployed policy in the target interface. Proposition 6.1 and Minkowski's inequality give

$$M(\theta, 2) \le \frac{(\varepsilon_g + \varepsilon_e + \varepsilon_\eta)^2}{4\tau(1-\gamma)}. \tag{21}$$

D.4 Correspondence Controls Readout Drift

To relate readout stability to model behavior, suppose T_s bijectively matches output realizations of each semantic action across interfaces. Define the model's sensitivity to this correspondence by

$$\delta_{\mathrm{tok}}(s) = \sup_y \left|\log \pi_\theta(T_s y \mid c_2(s)) - \log \pi_\theta(y \mid c_1(s))\right|. \tag{22}$$

Proposition D.2 (Correspondence controls readout drift)**.** *For a bijection that preserves semantic action fibers, with finite* $\delta_{\mathrm{tok}}(s)$*,*

$$|\eta_{12}(s,a) - \eta_{12}(s,b)| \le 2\tau\delta_{\mathrm{tok}}(s), \qquad \|\eta_{12}(s,\cdot)\|_2 \le \tau\sqrt{n}\,\delta_{\mathrm{tok}}(s). \tag{23}$$

Proof. At a fixed state, let $p_i(y) = \pi_\theta(y \mid c_i(s))$ and $\delta = \delta_{\mathrm{tok}}(s)$. The bijection that preserves action fibers gives

$$\Pi_{\theta,2}(a \mid s) = \sum_{y \in Y_a^{(1)}(s)} p_2(T_s y).$$

Since $e^{-\delta}p_1(y) \le p_2(T_s y) \le e^{\delta}p_1(y)$, summing over the fiber yields

$$e^{-\delta}\Pi_{\theta,1}(a \mid s) \le \Pi_{\theta,2}(a \mid s) \le e^{\delta}\Pi_{\theta,1}(a \mid s).$$

Write $v_a = \log \Pi_{\theta,2}(a \mid s) - \log \Pi_{\theta,1}(a \mid s)$. Then $|v_a| \le \delta$, and $\eta_{12} = \tau P_\perp v$. Pairwise subtraction gives $|\eta_a - \eta_b| \le 2\tau\delta$. Orthogonal projection is nonexpansive, so $\|\eta_{12}\|_2 \le \tau\,\|v\|_2 \le \tau\sqrt{n}\,\delta$. □

D.5 A Two-Action Illustration of Reversal and Adaptation

Consider a bandit with two actions and a uniform reference. The source execution protocol gives rewards $(1, 0)$, and its optimal policy chooses the first action with probability $p = e^{1/\tau}/(1 + e^{1/\tau})$. A revised protocol lowers the first action's utility to 0 and raises the second action's utility to $u > 0$. Holding the semantic policy fixed isolates the effect of this change in action values.

The two ceilings are

$$C_1 = \tau \log \frac{e^{1/\tau} + 1}{2}, \qquad C_2 = \tau \log \frac{1 + e^{u/\tau}}{2}.$$

Hence $C_2 > C_1$ if and only if $u > 1$. Holding the actual policy fixed also holds its reference KL fixed. Its returns satisfy

$$J_2 - J_1 = u(1-p) - p.$$

The policy gains u when it takes the second action and loses 1 when it takes the first. Its return therefore improves only when $u > p/(1-p) = e^{1/\tau}$. Throughout $1 < u < e^{1/\tau}$, the revised

interface has a higher ceiling and a lower deployed return: the unchanged policy still favors the action preferred under the source optimum.

Because the source policy is optimal, Corollary 5.2 makes the revision beneficial once the remaining mismatch is smaller than $C_2 - C_1$. Reaching the new optimum realizes this entire gain, positive for every $u > 1$. Figure 7 plots the frozen policy's return change and the ceiling gain at $\tau = 0.5$, with the vertical distance between the curves giving the mismatch under the revised interface. All quantities here are analytical.

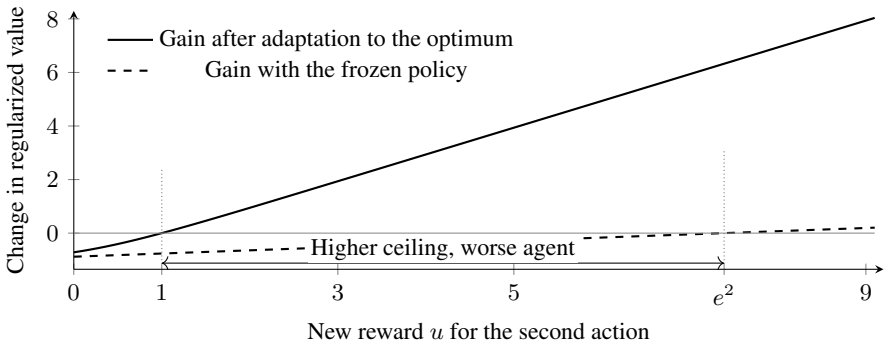

Figure 7: Adaptation can realize an initially harmful interface improvement. Analytical curves for the two-action bandit with $\tau = 0.5$. The frozen policy exceeds the original pair only for $u > e^2$ (dashed), while the ceiling gain is positive for $u > 1$. Reaching the new optimum realizes the ceiling gain $C_2 - C_1$ (solid). Corollary 5.2 gives the mismatch threshold for a benefit before full adaptation.

D.6 Policy Adaptation and Interface Rankings

Proof of Corollary 5.2. Keep the evaluation objective and reference fixed for each interface. Substituting $J(\theta, \mathcal{I}_j) = C_j - M(\theta, \mathcal{I}_j)$ gives

$$J(\theta_0, \mathcal{I}_2) - J(\theta_0, \mathcal{I}_1) = \Delta C + M(\theta_0, \mathcal{I}_1) - M(\theta_0, \mathcal{I}_2),$$
$$J(\theta_1, \mathcal{I}_2) - J(\theta_0, \mathcal{I}_1) = \Delta C + M(\theta_0, \mathcal{I}_1) - M(\theta_1, \mathcal{I}_2).$$

The initially harmful revision makes the first difference negative. Since $\Delta C + M(\theta_0, \mathcal{I}_1) = C_2 - J(\theta_0, \mathcal{I}_1)$, the second is positive exactly under Eq. (10). Combining the initial harm and subsequent improvement gives the equivalent chain

$$M(\theta_1, \mathcal{I}_2) < \Delta C + M(\theta_0, \mathcal{I}_1) < M(\theta_0, \mathcal{I}_2).$$

The initial mismatch $M(\theta_0, \mathcal{I}_1)$ allows for a source checkpoint that has not reached its optimum. For an optimal source policy, the threshold equals the ceiling gain ΔC, and reaching the new optimum realizes the entire gain. □

Interaction between weight and interface updates. For $j \in \{1, 2\}$, define the gain from the weight update in interface j by

$$A_j = J(\theta_1, \mathcal{I}_j) - J(\theta_0, \mathcal{I}_j) = M(\theta_0, \mathcal{I}_j) - M(\theta_1, \mathcal{I}_j).$$

Write $\Delta J(\theta) = J(\theta, \mathcal{I}_2) - J(\theta, \mathcal{I}_1)$ for the relative value of the revision. The interaction across the four combinations is

$$\Omega = \Delta J(\theta_1) - \Delta J(\theta_0) = A_2 - A_1. \tag{24}$$

The ceiling terms cancel because the objectives are fixed across checkpoints. A positive Ω means the revision becomes relatively more valuable after the weight update. If $\Delta J(\theta_0) < 0$, its ranking reverses exactly when $\Omega > -\Delta J(\theta_0)$. This compares the two interfaces at the updated checkpoint, while Corollary 5.2 compares the adapted pair with the initial pair. Reporting A_1 and A_2 also identifies the source of the interaction: deterioration under the source interface ($A_1 < 0$) can increase Ω even when $A_2 = 0$.

E MEASUREMENT THEORY

E.1 CURVATURE ON THE DECISION QUOTIENT

For $p = \mathrm{softmax}(F/\tau)$,

$$\nabla^2 \Phi(F) = \frac{1}{\tau}\big(\mathrm{diag}(p) - pp^\top\big). \tag{25}$$

The categorical covariance annihilates $\mathbf{1}$. Its spectral norm is at most $1/2$: the absolute row sum of row i is $2p_i(1-p_i) \le 1/2$, and the matrix is symmetric. If $p_i \ge p_{\min}$ and $z \perp \mathbf{1}$, then

$$\begin{aligned} z^\top\big(\mathrm{diag}(p) - pp^\top\big)z &= \min_c \sum_i p_i (z_i - c)^2 \\ &\ge p_{\min} \min_c \sum_i (z_i - c)^2 = p_{\min} \|z\|_2^2 . \end{aligned}$$

Thus the upper Hessian constant is $1/(2\tau)$ and the lower constant on the centered subspace is $p_{\min}/\tau$.

E.2 BOUNDS ON THE PERFORMANCE GAP

Proposition 6.1 bounds mismatch through centered score error. An upper bound below the threshold in Corollary 5.2 guarantees recovery of the initial pair's return. Under a condition on probability overlap, the same score error also provides a lower bound on mismatch. We state the complementary bound and prove both results below.

Corollary E.1 (Complementary lower bound)**.** *With x_s and ε_d as in Proposition 6.1, let $p_{\min} > 0$ be a uniform lower bound on action probabilities along the score interpolation between the actual and optimal policies. Then*

$$\frac{p_{\min}\varepsilon_d^2}{2\tau(1-\gamma)} \le M(\theta, \mathcal{I}) \le \frac{\varepsilon_d^2}{4\tau(1-\gamma)}. \tag{26}$$

Proof of both bounds. At a fixed state, let $x = \widehat{F} - \bar{F}^*$, with both arguments centered. Taylor's integral remainder gives

$$D_\Phi(\bar{F}^*, \widehat{F}) = \int_0^1 (1-t) x^\top \nabla^2 \Phi\big(\widehat{F} + t(\bar{F}^* - \widehat{F})\big) x \, \mathrm{d}t.$$

The curvature bounds above and $\int_0^1 (1-t)\,\mathrm{d}t = 1/2$ imply

$$\frac{p_{\min}}{2\tau} \|x\|_2^2 \le D_\Phi(\bar{F}^*, \widehat{F}) \le \frac{1}{4\tau} \|x\|_2^2 .$$

Taking the actual occupancy expectation and dividing by $1-\gamma$ proves both bounds. □

A lower probability bound at both endpoint policies is sufficient along their score interpolation: the logarithm of softmax is concave in the score vector, so each interpolated log probability is at least the linear interpolation of its endpoint log probabilities.

The quadratic upper rate and constant are locally tight. For two actions around the uniform distribution, the categorical covariance has eigenvalue $1/2$ on $\mathbf{1}^\perp$, so $D_\Phi(F, F+z) = \|z\|_2^2/(4\tau) + o(\|z\|_2^2)$ for centered z. No positive quadratic lower constant independent of the action probabilities is possible: near a policy concentrated on one action, the nonconstant curvature can approach zero.

F EXPERIMENTAL DETAILS

We describe the experiments in the order of the main text: the MiniWoB++ setup and post-training study, the readout study, the H0-to-H1 adaptation study, the adaptation across harness swaps, the calibration of finite measurements, the independent prediction study, the coding-agent harness study, and the exact comparison of harness scores with task success.

F.1 MINIWOB++ SETUP AND POST-TRAINING

The controlled study pairs language-model post-training with a finite control model of each harness. This gives every checkpoint a fixed optimal policy to compare against, while retaining task instructions, parameterized actions, and controller feedback in the model's input.

We compare six harnesses, H0–H5, using H0 as the direct baseline.

Tasks and observations. We use three tasks from MiniWoB++ (Liu et al., 2018). In `click-checkboxes`, the agent toggles checkboxes to match the requested labels and submits the selection. In `click-checkboxes-soft`, the requested words and checkbox labels are matched by synonym meaning. In `ascending-numbers`, the agent clicks five tiles in numerical order, with the remaining numbers hidden after the first click. We retain the initially visible layout in the public observation, together with spatial tile IDs, the remaining tiles, and the number of completed clicks. Checkbox observations contain the instruction, each label and its checked status. All actor observations also expose the remaining decision budget and the harness's controller state.

The reward is one for exact task completion and zero otherwise, including partial completion. All explicit action and tool costs are zero. Each episode allows at most ten actor decisions, with discount $\gamma = 0.95$ per decision. We compile the physical transitions, remaining budget, and controller memory into a finite acyclic graph. For every retained task, we verify a successful run in the native browser and separately compare the compiled transitions and termination rules with the page behavior. Training and exhaustive policy evaluation use these compiled graphs.

Harnesses. Table 4 specifies the six interfaces. The program verifier approves an action when it avoids an incorrect terminal outcome, using the public instruction, page state, and the benchmark's fixed synonym vocabulary. Verification targets are checkbox submission and each of the five tile clicks. H4 uses a separate frozen Qwen3.5-2B reviewer whose prompt contains the public task and physical page state together with the proposed action, omitting the actor's remaining budget and controller memory. It greedily selects between single-token approve/reject codes. For H3 and H4, verification feedback becomes part of the next observation and is cleared by a native action. The H1 batch executes two ordered clicks within one actor decision and stops at native termination, so it receives only one discount step. All ordered pairs of click IDs are available, including repeated IDs.

Table 4: MiniWoB++ harnesses. Every menu includes the complete parameters of each action. We measure the common budget in actor decisions.

ID	Interface	Actions, execution, and controller state
H0	Direct	One native click or checkbox submission per decision.
H1	Batched clicks	Adds every ordered pair of clicks as a single menu option, executed without an intervening actor call.
H2	Automatic gate	Checks a commit action before execution. Rejection leaves the page unchanged, consumes the decision, and disables subsequent checks. Memory stores whether the gate is enabled.
H3	Verify tool	Adds an optional program check of a proposed native action. Memory stores the checked action and its advisory verdict.
H4	Reviewer tool	Uses H3's action menu and memory, with advice supplied by a frozen language-model reviewer.
H5	Revised instructions	Uses H0's actions, observations, dynamics, and budget, with additional system instructions for matching labels, tracking tile IDs, correcting selections, and deciding when to submit.

Data splits. We generate tasks with the native pages and deduplicate them by full public task content before assigning splits. Table 5 gives the fixed counts. Checkbox instances contain fewer than five boxes, and in the number task the first required number lies in one of the first three spatial positions. The 72-task training probe is fixed before training and is a subset of the training set. The six harnesses share the same task contents and splits. Test tasks are held out from parameter updates.

All MiniWoB++ experiments reported in this paper use the same fixed test set of 576 tasks generated from MiniWoB++, with 192 tasks per family. Calibration and training-probe tasks remain separate.

Table 5: MiniWoB++ task allocation. Splits have disjoint public contents, except that the training probe comes from training. Metrics average tasks within each family and then weight the families equally.

Split	Per family	Total	Use in post-training
Training	480	1,440	Parameter updates
Calibration	96	288	Full learning curves
Training probe	24	72	Fixed training subset for learning curves
Test	192	576	Transfer evaluation

Table 6: Post-training settings. All harnesses share the same update and task-draw budgets, with computation depending on the number of reachable states and menu entries.

Parameter	Setting
Base / reference / H4 reviewer	Qwen3.5-2B with frozen base, reference, and reviewer weights
Trainable adapter	LoRA rank 16, scale $\alpha = 32$, dropout 0, no bias updates
LoRA projection targets	Attention: `q/k/v/o_proj` Linear attention: `in_proj_qkv`, `in_proj_z`, `in_proj_b`, `in_proj_a`, `out_proj` MLP: `gate/up/down_proj`
Optimizer	AdamW with a constant learning rate 10^{-5}, $\beta = (0.9, 0.999)$, $\epsilon = 10^{-8}$, weight decay 0, gradient norm clipping at 1
Training seed and budget	Seed 42 and 1,024 updates per harness at four tasks per update (4,096 task draws)
Checkpoint schedule	Initialization and every 64 updates (17 checkpoints at the full budget)
Numerical precision	BF16 for model forward passes FP32 for trainable parameters and optimizer states FP64 for categorical log-softmax and dynamic programming
State microbatch	Eight states grouped by prompt length and menu size
Prompt length	At most 2,048 tokens without truncation

Policy readout and reference. The actor receives a system message specifying the task and harness rules and a JSON observation listing every available action. Each complete action, including its arguments, maps bijectively to one case-sensitive token from `A–Z/a–z`. We normalize the next-token logits over this full menu at sampling temperature one, using the model's chat template with thinking disabled. This categorical distribution defines $\Pi_{\theta,\mathcal{I}_h}$ throughout training and evaluation. The base model and tokenizer are Qwen3.5-2B. An independent frozen copy, without the trained adapter, reads the same messages to define $\pi^{\mathrm{ref}}_{\mathcal{I}_h}$ at each state. The reference stays fixed across checkpoints. Because H5 changes the reference model's input as well as the actor's, we solve its regularized optimum separately from H0's even though their execution dynamics coincide.

Training distribution and objective. Each harness starts from the same base weights and initial LoRA adapter (Hu et al., 2022), with training seed 42. To retain correction states as successful behavior becomes more frequent, training uses

$$\mu_h = \tfrac{1}{2}\delta_{s_{0,h}} + \tfrac{1}{2}\nu_h, \qquad J_{\mu_h}(\theta) = \mathbb{E}_{\mu_h,\Pi_{\theta,\mathcal{I}_h},P_h}\sum_{t\geq 0}\gamma^t\left[r_h(s_t,a_t) - \tau\log\frac{\Pi_{\theta,\mathcal{I}_h}(a_t \mid s_t)}{\pi^{\mathrm{ref}}_{\mathcal{I}_h}(a_t \mid s_t)}\right], \quad (27)$$

with $\tau = 0.1$. Here $s_{0,h}$ is the native task start. We form ν_h from all other reachable nonterminal states, stratifying by correction category or completion progress, remaining-budget bin, and controller memory. Checkbox categories are correct, missing selections only, extra selections only, and both missing and extra selections. For the number task, categories are defined by the number of completed clicks. Budget bins are 1–3, 4–6, and 7–9 remaining decisions. We assign equal mass to nonempty strata and then to states within each stratum. Each start has a legal action prefix from the native state, and the objective begins at that start with its remaining budget. This gives every reachable decision state positive training mass while leaving the statewise dynamics, rewards, and optimal action preferences fixed.

Exact-PG updates. Each update samples four tasks with replacement, first choosing a task family uniformly and then an instance uniformly within that family. For each task, dynamic programming evaluates the current policy's regularized value and advantage, and propagates discounted visitation from μ_h. Exact-PG sums the policy gradient over this task's reachable states and complete action menu. Gradients for the four tasks are evaluated at the same parameters and averaged once per optimizer step. The objective thus weights tasks equally regardless of graph size. A separate soft Bellman solve provides $\Pi^*_{\mathcal{I}_h}$ and $F^*_{\mathcal{I}_h}$ as evaluation targets. Table 6 gives the optimization settings.

Each training run uses one RTX 4090 GPU.

Geometry and return measurements. At initialization and each saved checkpoint, we read the complete action distribution at every reachable nonterminal state of all calibration and training-probe tasks. For a task u, let $\mathcal{S}_{h,u}$ denote this fixed state bank. Its geometry error is

$$\mathrm{MSE}_{h,u}(\theta) = \frac{1}{|\mathcal{S}_{h,u}|} \sum_{s \in \mathcal{S}_{h,u}} \frac{\left\| \tau P_\perp \left[\log \Pi_{\theta,\mathcal{I}_h}(\cdot \mid s) - \log \Pi^*_{\mathcal{I}_h}(\cdot \mid s) \right] \right\|_2^2}{|\mathcal{A}_{h,u}(s)|}. \tag{28}$$

We average this quantity within task families and then across the three families. The fixed bank makes the trajectory of this error comparable across checkpoints, including decisions that become rare under the learned policy. Alongside it, we compute native-start success probability, J, C, and the squared score error weighted by the evaluated policy's discounted occupancy. Success is the undiscounted probability of exact completion, evaluated by dynamic programming. Mismatch is computed independently as $\tau/(1-\gamma)$ times the occupancy-weighted $\mathrm{KL}(\Pi_{\theta,\mathcal{I}_h} \| \Pi^*_{\mathcal{I}_h})$, allowing comparison with $C - J$. We sum over nonterminal states without renormalizing their occupancy mass.

Transfer and uncertainty. For transfer, we evaluate the six trained actors at the common final update 1,024 under all six deployment harnesses on the fixed test set, which gives the 6×6 matrix in Figure 3. Every cell uses its deployment harness's state graph, menu, frozen reference, and optimal target. Comparisons within a column therefore hold the deployment control problem fixed.

The learning curves in Figure 3 use all 288 calibration tasks at initialization and every 64 updates through update 1,024. For both success and geometry MSE, we form pointwise 95% percentile intervals from 2,000 bootstrap resamples of tasks within each family. The same task draws are shared across metrics and checkpoints, with equal weight on each family. Changes from initialization use paired differences under these same draws. The intervals describe task-sampling uncertainty for the single training seed 42. All measured checkpoints are plotted without smoothing. We make the same measurements on the 72-task training probe. From initialization to update 1,024, the geometry error falls by 69% to 90% across the six harnesses, both on the training probe and on the calibration tasks, which play no role in parameter updates. For each harness, the two reductions differ by less than one percentage point. Post-training thus improves the geometry on held-out and training tasks alike.

F.2 Readout Study

Tasks and frozen checkpoint. The main-text comparison uses the H0-trained Qwen3.5-2B checkpoint at post-training update 1,024, with training seed 42. Its weights remain frozen throughout evaluation. We evaluate it on the fixed test set in Table 5, the same 576 tasks used by the adaptation and prediction studies below. R0 therefore reproduces the initial H0 results in Table 8.

Presentation variants. R0 uses the canonical H0 system message, JSON observation, and action-code mapping. R1 reverses the order of top-level observation fields while preserving their contents and the nested action dictionary. R2 replaces the action dictionary with a list of records containing the same code and action fields. R3 shifts the action-code assignment cyclically by one position, R4 reverses it, and R5 uses a fixed random permutation for each menu size. These three variants preserve the order and full arguments of the semantic actions and update the displayed code assignment and decoder together. We align every probability table by semantic action before comparison. The system message and all task information are unchanged, and inputs are evaluated without truncation.

Shared reference and exact evaluation. We fix the semantic reference $q_0(a \mid s)$ to the frozen base model's distribution under R0 and assign this same probability to the corresponding code in

each variant. The evaluated actor reads each variant normally. All conditions use H0's native actions and dynamics, ten model decisions, $\gamma = 0.95$, $\tau = 0.1$, and zero explicit costs. The full legal single-token menu is evaluated at sampling temperature one. Model forward passes use BF16 with state microbatches of eight, followed by FP64 probability normalization and control calculations. We enumerate reachable states, solve the optimum separately in each code ordering, and compute return and success from the measured semantic policy by dynamic programming. Mismatch is computed independently from occupancy-weighted $\mathrm{KL}(\Pi \| \Pi^*)$. The largest ceiling difference is 2.2×10^{-16} and the largest optimal-action probability difference is 8.3×10^{-16} across variants.

Measurements and uncertainty. Choice change is the fraction of states where the most probable semantic action differs from R0. Readout RMS is the square root of the mean squared entries of $\eta = \tau P_{\perp}(\log \Pi_{\theta,R_v} - \log \Pi_{\theta,R_0})$. We average state-level measurements within each task, then average tasks within each family and give the three families equal weight. All variants are evaluated on the same tasks. We form percentile intervals from 2,000 paired task bootstrap resamples within families. These intervals quantify task variation at the fixed checkpoint. Table 7 reports the readout and performance measurements.

Table 7: H0 readout results, with 95% paired-task bootstrap intervals below each mean. RMS intervals apply the square root to the mean-squared drift interval endpoints. The target drift is numerically zero in every presentation.

Variant	Readout RMS	Mismatch M	Return J	Success (%)
R0	0.000	0.015	0.423	85.0
	[0.000, 0.000]	[0.010, 0.021]	[0.409, 0.436]	[84.0, 86.1]
R1	0.200	0.484	−0.046	19.2
	[0.198, 0.203]	[0.460, 0.507]	[−0.072, −0.020]	[17.0, 21.4]
R2	0.054	0.023	0.415	86.0
	[0.053, 0.054]	[0.018, 0.028]	[0.401, 0.428]	[85.1, 86.9]
R3	0.181	0.148	0.289	57.1
	[0.179, 0.183]	[0.143, 0.155]	[0.274, 0.303]	[55.9, 58.3]
R4	0.181	0.130	0.308	68.2
	[0.179, 0.182]	[0.124, 0.136]	[0.292, 0.323]	[66.9, 69.5]
R5	0.168	0.110	0.328	67.6
	[0.166, 0.170]	[0.104, 0.116]	[0.312, 0.342]	[66.2, 68.9]

F.3 Adaptation After an Interface Revision

Revision and source checkpoint. We study H0-to-H1 adaptation using the model, task families, action decoder, and harness definitions of Appendix F.1. H1 retains H0's native actions and adds ordered pairs of clicks executed within one actor decision, stopping at native termination. The source checkpoint θ_0 is the H0-trained Qwen3.5-2B model at post-training update 1,024, with source training seed 42.

Tasks and splits. Training reuses the 1,440 training tasks and the coverage mixture μ_h in Eq. (27). The evaluation reuses all 576 tasks in Table 5, with 192 instances per family. These tasks are held out from parameter updates and checkpoint selection. We reuse the same task graphs, action menus, and reference probabilities as the post-training study. All deployment metrics start from the native task starts ρ_0.

Training branches and budget. Both branches inherit the same source LoRA weights and reset AdamW's first and second moments. One continues under H0, giving θ_s, and the other adapts under H1, giving θ_t. Both use adaptation seed 42 and paired task draws. Each receives 1,024 additional Exact-PG updates with four tasks per update, for 4,096 task draws. We retain the post-training settings in Table 6, including LoRA rank 16 and scale 32, constant learning rate 10^{-5}, BF16 forward passes, and state microbatches of eight. Each interface's dynamics, costs, frozen base-model reference, $\gamma = 0.95$, and $\tau = 0.1$ stay fixed throughout training and evaluation.

Evaluation schedule and measurements. We evaluate at additional updates 0, 32, 64, 128, 256, 512, and 1,024. The initial checkpoint is evaluated under both interfaces, and each later checkpoint from both branches is evaluated under both interfaces, giving 26 complete evaluations. The budget and checkpoint schedule were fixed before adaptation. The endpoint comparisons use update 1,024, and the curves include every scheduled checkpoint.

The oracle solves the soft Bellman equations for C_j and $\Pi^*_{\mathcal{I}_j}$. Complete action probabilities and the evaluated policy's forward discounted visitation determine M through Eq. (6). Terminal states contribute zero loss, and nonterminal visitation mass is retained without renormalization. Independently, Bellman policy evaluation computes J, while undiscounted policy evaluation gives task success. The residual in $J = C - M$ checks numerical consistency. These measurements use the compiled task graphs and full action distributions.

Comparisons. For each branch, we compare the target-interface mismatch with the fixed recovery threshold $C_2 - J(\theta_0, \mathcal{I}_1)$. We separately evaluate recovery beyond the initial pair, interface ranking for the updated checkpoint, and the return difference between the H1-adapted pair and continued training under H0. Evaluating both branches under H1 also measures the effect of the training interface at a common deployment interface. We report the component gains A_1, A_2 and their difference Ω from Appendix D.6, alongside regularized return and task success.

Uncertainty and computation. Metrics average tasks within each family and weight the families equally. We form pointwise 95% percentile intervals from 2,000 paired bootstrap resamples within families. The same resampling indices are shared across checkpoints, interfaces, metrics, and paired differences. These intervals describe task-sampling uncertainty given the fixed source and adaptation seed 42. Each branch trains on one RTX 4090 GPU, and a third GPU evaluates checkpoints.

Adaptation results. H1 has a higher ceiling than H0, and both stay fixed across checkpoints. The frozen swap increases mismatch enough to outweigh this gain and lower return. H1 adaptation then reduces mismatch below the recovery threshold, and the final adapted pair exceeds both the initial H0 pair and H0 continuation under H0. Table 8 gives the initial and final measurements. Rising return under H1 and falling return under H0 together reverse the ranking along the adaptation branch.

The interface ranking reverses before the adapted pair recovers the initial return. At the first recorded checkpoint with a positive mean recovery difference, its paired interval still includes zero. By the final budget, the paired intervals for recovery and improvement over H0 continuation both lie above zero. Table 9 reports these endpoint comparisons, and Figure 5 shows the recovery and ranking trajectories. In the notation of Appendix D.6, $\mathcal{I}_1$ is H0 and $\mathcal{I}_2$ is H1.

Table 8: Initial and final deployment outcomes. Both training branches receive 1,024 additional updates. J is regularized return, M is occupancy-weighted decision loss, and success is the undiscounted probability of completing the task.

Checkpoint	Deployment	J	M	Success (%)
Initial	H0	0.42289	0.01485	85.05
Initial	H1	0.31285	0.13737	72.90
H0 continuation	H0	0.42294	0.01481	83.37
H0 continuation	H1	0.30404	0.14619	75.66
H1 adaptation	H0	0.38642	0.05132	78.31
H1 adaptation	H1	0.43133	0.01889	88.17

F.4 Adaptation Across Harness Swaps

Among the 30 ordered swaps between H0–H5, a frozen comparison rejects 22, and 13 of those have $T \leq 0$, which leaves nine. Each of the nine starts from the checkpoint post-trained under its source harness in Appendix F.1 and adapts under the target harness with the protocol of Appendix F.3. We recorded the predictions for the eight swaps other than H0→H1 before adapting them.

Table 10 lists the nine swaps. For the eight swaps recorded in advance, $T - r_2$ orders the gains with Spearman $\rho = 0.95$, the ceiling gain alone with $\rho = 0.86$, and the frozen return change with

Table 9: Paired comparisons with 95% task bootstrap intervals. Final comparisons use the H1-adapted checkpoint at 1,024 additional updates. Recovery compares it under H1 with the initial H0 pair. Interface ranking compares H1 and H0 at the same adapted checkpoint. Gain over H0 continuation compares the two branches at the same update budget, each under its training harness.

Quantity	Mean [95% CI]
Ceiling gain $C_2 - C_1$	0.01248 [0.00698, 0.01754]
Frozen mismatch increase	0.12252 [0.11487, 0.12961]
Frozen return change	-0.11004 [-0.11558, -0.10438]
Recovery threshold T	0.02733 [0.01984, 0.03520]
Final recovery difference	0.00844 [0.00135, 0.01447]
Final interface ranking	0.04491 [0.03653, 0.05254]
Final gain over H0 continuation	0.00840 [0.00254, 0.01411]
A_1: change under H0	-0.03647 [-0.04033, -0.03295]
A_2: change under H1	0.11848 [0.10964, 0.12658]
$\Omega = A_2 - A_1$	0.15495 [0.14430, 0.16493]

$\rho = 0.17$. Subtracting the frozen return change and the gain from T gives the target mismatch before and after adaptation. It falls from between 0.015 and 0.146 to at most 0.019. The observed gain exceeds $T - r_2$ in every swap, by 0.004 on average. The ranking predicted from the checkpoint post-trained under the target has the correct sign in all eight recorded swaps and orders the observed rankings with Spearman $\rho = 0.91$. After adapting from H1 to H0, the checkpoint prefers H0 by 0.041 and still falls short of the initial pair, so a ranking reversal alone does not show a net gain. Adapting from H0 to H5 also beats continued training under H0 by 0.085 [0.078, 0.092]. The 13 swaps with $T \leq 0$ include all five swaps from H5, with T between -0.136 and -0.016.

Table 10: The nine swaps that a frozen comparison rejects and that have $T > 0$, ordered by the observed gain. The ceiling gain ΔC, the frozen return change, T, and the predicted gain $T - r_2$ come from the post-trained checkpoints before adaptation. The gain is the adapted pair's return minus the initial pair's return after 1,024 updates, with its 95% paired task-bootstrap interval. Ranking is the adapted checkpoint's return under the target minus its return under the source. Success is the change in task success from the initial pair, in percentage points. The prediction for H0→H1 was computed after its adaptation.

Swap	ΔC	Frozen	T	$T - r_2$	Gain [95% CI]	Ranking	Success
H1→H5	0.068	−0.054	0.092	0.081	0.085 [0.079, 0.091]	0.167	+9.0
H0→H5	0.081	−0.004	0.095	0.085	0.085 [0.078, 0.091]	0.111	+8.7
H3→H1	0.078	−0.040	0.099	0.075	0.083 [0.071, 0.094]	0.169	+11.7
H4→H1	0.075	−0.040	0.089	0.066	0.071 [0.061, 0.081]	0.164	+12.6
H2→H5	0.038	−0.018	0.053	0.042	0.045 [0.039, 0.051]	0.084	−0.5
H1→H2	0.030	−0.055	0.054	0.039	0.042 [0.038, 0.047]	0.082	+9.8
H0→H1	0.012	−0.110	0.027	0.003	0.008 [0.001, 0.014]	0.045	+3.1
H4→H3	−0.003	−0.004	0.011	−0.009	−0.001 [−0.005, 0.002]	−0.001	+0.1
H1→H0	−0.012	−0.060	0.011	−0.003	−0.002 [−0.008, 0.004]	0.041	−2.5

F.5 Calibration of Finite Mismatch Measurements

Estimator. For each task and deployed pair, draw m independent pilot trajectories. For each trajectory, draw T with $\Pr(T = t) = (1 - \gamma)\gamma^t$ and retain its state s_T. Retaining one state per independent trajectory gives independent occupancy samples. A draw after termination contributes zero to the average over all m draws. At each nonterminal sampled state with K actions, draw N independent actions and form

$$\widetilde{p}(a \mid s) = \frac{n_a + \alpha}{N + K\alpha}. \tag{29}$$

These smoothed distributions enter the mismatch estimate in Eq. (11). Each nonterminal state draw uses the complete deployed action menu. Pilot and action-sampling queries are both counted.

Candidate budgets and smoothing. The grid uses $m \in \{64, 256\}$ and $N \in \{16, 64, 256, 1024\}$. Four smoothing rules use a pseudocount of 0.1 or 0.5 per action, or a total pseudocount $\beta \in \{0.5, 1\}$ divided across the K actions. This gives 32 candidates. Calibration uses 32 independent measurement pools, with shared nested prefixes across candidates. For each candidate, the calibration scorer computes the expected absolute-mismatch bias using exact occupancy and the binomial marginal distribution of each action count. The predictor receives only the sampled observations.

Selection criteria. A candidate must meet three criteria over H0 and H1 at every adaptation node: absolute expected mismatch bias at most 0.005 averaged over families, bias at most 0.01 within every family, and aggregate return-difference RMSE at most 0.01 for recovery and ranking across measurement pools. It must also reach RMSE at most 0.03 on the configuration calibration set. Among passing candidates, we select the lowest mean realized actor-query cost. Costs within 1% tie and are resolved by worst-node difference RMSE, m, N, and the fixed smoothing-rule order.

Four candidates pass, all with $m = 256$ and $N = 1024$. The selected rule uses $\alpha = 0.5$ per action and has configuration calibration RMSE 0.00887. Each test prediction uses a single measurement pool. Table 11 compares the initial and selected budgets, and Figure 8 shows the full sweep.

Table 11: Calibration of the measurement budget. Both columns use a pseudocount of 0.5 per action. Bias refers to the magnitude of the expected mismatch estimation error. Query counts include pilot actions and frequency samples at nonterminal states.

Calibration quantity	$m = 64, N = 16$	$m = 256, N = 1024$
Largest condition-level absolute bias in M	0.24805	0.00287
Largest family-level absolute bias in M	0.40320	0.00442
Worst-node return-difference RMSE	0.20259	0.00672
Mean actor queries per task and condition	356.9	42,553.3

Why the budget changes the estimate. Eq. (29) mixes the empirical action distribution with a uniform distribution of weight $K\alpha/(N + K\alpha)$. For a 30-action menu with $\alpha = 0.5$, this weight is 48.4% at $N = 16$ and 1.44% at $N = 1024$. Larger action budgets reduce this distortion, while more occupancy samples stabilize the aggregate estimate. On the test configurations of Table 12, increasing the budget reduces prediction RMSE from 0.01132 to 0.00230. The selected budget uses about 119 times as many actor queries on calibration.

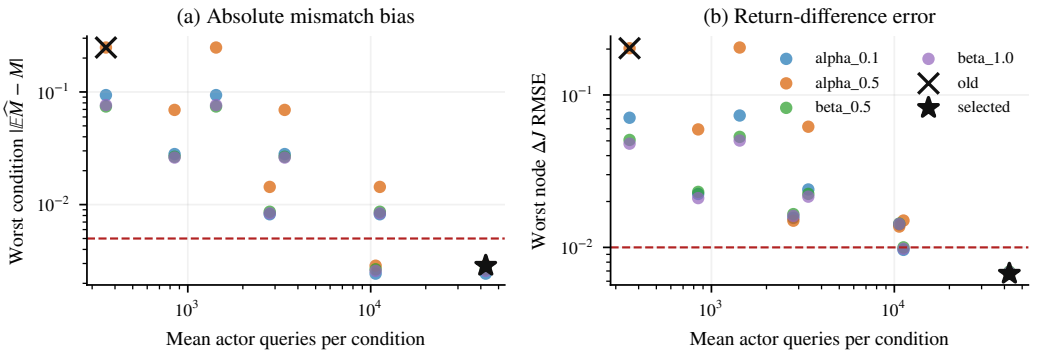

Figure 8: Measurement accuracy depends on the query budget and smoothing rule. Each point is a calibration candidate. (a) Largest absolute mismatch bias across deployment conditions. (b) Largest aggregate return-difference RMSE across adaptation nodes. Dashed lines mark calibration thresholds. The cross marks the initial budget and the star marks the selected estimator.

F.6 Independent Prediction of Deployment Outcomes

Tasks and checkpoints. The prediction study uses the fixed test set in Table 5, with 576 tasks divided equally across the three MiniWoB++ families. Estimator calibration uses a separate set of

72 tasks, 24 per family, generated in the same way as the splits in Table 5 and disjoint from all of them. This set is fixed before estimator selection and plays no role in the reported test results. We use the H0 checkpoint at post-training update 1,024 and the H1-adapted checkpoints from Appendix F.3. All weights are frozen for this study. Source training and adaptation both use seed 42. The control objective uses $\gamma = 0.95$, $\tau = 0.1$, and a horizon of ten model decisions.

Execution and encoding sweep. Both final checkpoints are deployed through H0's native action menu. We cross execution settings with three encodings: the canonical presentation R0, reversed top-level observation fields R1, and a cyclic permutation of action codes R3. The encodings preserve the semantic actions and their execution. The reference is the frozen base model's distribution under R0, aligned to the same semantic actions in all three presentations. Prompts are held fixed across execution settings and omit numerical reliability and cost values.

The baseline has click reliability $p = 0.8$, click cost $c = 0.01$, and encoding R0. One sweep varies $p \in \{0.5, 0.6, 0.7, 0.75, 0.8, 0.85, 0.9, 0.95, 1\}$ at fixed $c = 0.01$. A second varies $c \in \{0, 0.0025, 0.005, 0.0075, 0.01, 0.015, 0.02, 0.03, 0.05\}$ at fixed $p = 0.8$. Failed clicks leave the physical page unchanged and consume the decision and its cost. Submission retains the native task semantics. Calibration uses $p \in \{0.5, 0.8, 1\}$ and $c \in \{0, 0.01, 0.05\}$ in their respective sweeps. Crossing these settings with the encodings gives 15 calibration and 36 test configurations per checkpoint. The two checkpoints yield 72 test comparisons with their respective baselines.

Prediction and scoring. For each task and configuration, an oracle solves the control problem for C and the queried optimal action distributions. The predictor obtains independent pilot trajectories and finite categorical samples from the deployed policy. Eq. (11) gives $\widehat{M}$ for the candidate and baseline. Their predicted return difference is

$$\widehat{\Delta J} = \Delta C - \Delta \widehat{M}. \tag{30}$$

Appendix F.5 specifies the estimator and calibration. All test predictions are saved before scoring starts. The scoring process computes exact J and occupancy-weighted KL from full policy probabilities. The predictor's data access is limited to samples, pilot states, the reference, oracle ceilings, and queried optimal targets.

Adaptation comparisons. We evaluate the saved H1-adapted checkpoints at 0, 32, 64, 128, 256, 512, and 1,024 updates under native H0 and H1. Each interface retains its frozen base-model reference. At every node, the predictor estimates the recovery difference $J(\theta_t, \mathrm{H1}) - J(\theta_0, \mathrm{H0})$ and the ranking difference $J(\theta_t, \mathrm{H1}) - J(\theta_t, \mathrm{H0})$. The initial H0 baseline receives an independent measurement whose cost is included.

Baselines and access. Ceiling-only prediction sets $\Delta\widehat{M} = 0$. The frequency rollout baseline estimates regularized returns directly, using repeated queries at each visited state to estimate the selected action's log probability. Calibration selects 64 such samples per decision with additive smoothing $\alpha = 0.5$. This baseline receives the same actor-query allowance as the diagnostic, including its pilot queries and baseline measurement. Reference computation and oracle construction are recorded separately from actor queries.

Table 12: Prediction error on the fixed test tasks. Configuration RMSE averages over 72 checkpoint and configuration comparisons. Recovery and ranking RMSE average over seven saved adaptation checkpoints. No estimator reads exact action probabilities.

Estimator	Configuration RMSE	Recovery RMSE	Ranking RMSE
Ceiling minus measured mismatch	0.00230	0.00245	0.00286
Ceiling alone	0.23470	0.05744	0.05342
Frequency rollout, same query budget	0.00240	0.02672	0.02667

Prediction results. Figure 9 compares predicted and observed return changes across the configuration sweep. Table 12 reports errors in aggregate return differences, where each comparison first averages returns equally across task families. In the configuration sweep, the mismatch-based prediction identifies the direction of all 70 changes outside a 0.01 tie region and orders all 1,195

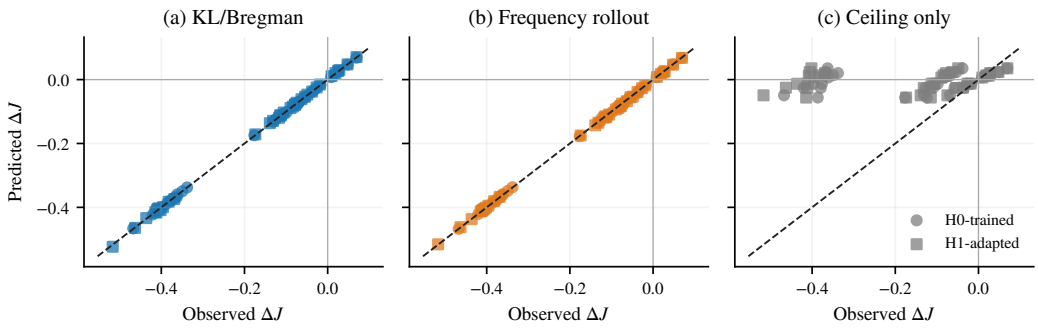

Figure 9: Measured mismatch predicts the return realized under a harness revision. Each point compares a candidate with its baseline on the fixed test tasks. The diagonal marks agreement between predicted and observed return changes. Circles and squares denote the H0-trained and H1-adapted checkpoints. (a) Ceiling minus measured mismatch. (b) Direct rollouts with action probabilities estimated from frequencies at the same actor-query budget. (c) Ceiling alone.

configuration pairs outside that region within each checkpoint. Classifying changes as harmful, tied, or beneficial gives 71 correct predictions out of 72.

In the adaptation study, at the final update 1,024, the exact recovery and ranking differences are 0.00844 and 0.04491, with intervals in Table 9. Their predictions are 0.00579 with 95% interval $[-0.00116, 0.01228]$ and 0.04195 with interval $[0.03371, 0.05015]$. Using the same 0.01 tie region, three-class predictions agree with the exact values at six of seven recovery nodes and all seven ranking nodes. Figure 5 in Section 6.2 shows the predicted and observed trajectories.

Uncertainty and resources. We average within each family and give equal weight to the three families. Intervals use 2,000 paired task bootstrap resamples within families. They describe task-sampling uncertainty for the fixed checkpoints. Finite action samples are simulated from cached distributions of the actual language-model checkpoints. The new probability measurements took 2.19 GPU-hours.

F.7 Coding-Agent Harness Study

Harnesses and benchmark. The study uses the native configurations of DeepAgents, Pi coding-agent, DeepSeek Harness, Terminus-2, and mini-SWE-agent. All file and command tools run inside the benchmark task container. The framework owns its loop, tool feedback, delegation where configured, compression, retries, and termination. Terminus-2 retains tmux, completion confirmation, and summarization. Mini-SWE-agent uses its official tool-call profile. Terminal-Bench Lite contains 39 tasks, scored by the official verifier. The agent never sees scoring data or gold patches.

Accuracy. Each harness has three independent runs. Accuracy is the mean of the three single-run success rates over all planned tasks, and the reported spread is the sample standard deviation across runs. Timeouts and failures remain in the denominator as unresolved tasks.

Models and budgets. The target is Qwen3.5-9B with thinking enabled, temperature 0.6, an output limit of 32,768 tokens, and a context window of 65,536 tokens. The teacher is GPT-5.6-luna with the same output and context limits and high reasoning effort. Each trajectory is capped at 50 top-level decisions and uses the task's own time limit.

Recorded states and sampling. At each model call, we record the request context, tool definitions, controller state, and remaining budgets. Summary, subagent, and retry calls count as decisions. To sample a decision, we resend the recorded request and decode each sample with the harness's parser, without executing tools. Samples measure the next action distribution at the recorded state.

Probe tasks and trajectories. The probes use three Terminal-Bench Lite tasks: `acl-permissions-inheritance`, `analyze-access-logs`, and `bank-trans-filter`. For each harness, the target and the teacher each contribute three trajectories per task, taken from separate runs of their own. A trajectory enters an estimate only when every recorded state in it has a complete probe. The teacher estimates use 7 of 9 trajectories for Pi coding-agent and all 9 for the other harnesses. The target estimates use 7, 6, and 8 of 9 trajectories for Pi coding-agent, DeepAgents, and DeepSeek Harness, and all 9 for Terminus-2 and mini-SWE-agent.

Common action categories. Each decoded output maps to one of 13 categories: inspect, modify, verify, setup, run, plan, delegate, wait, stop, context update, return, opaque, and invalid. File reads, listings, and searches map to inspect, and writes, edits, and deletions to modify. Todo and planning operations map to plan, and subagent calls to delegate. Shell commands, including Terminus-2 keystrokes at a shell prompt, are parsed, and a command maps to its dominant operation. Commands whose program the parser cannot resolve map to opaque, and responses without a countable native action map to invalid. The categories merge actions with different arguments and effects, so the estimates compare decisions at the level of these categories.

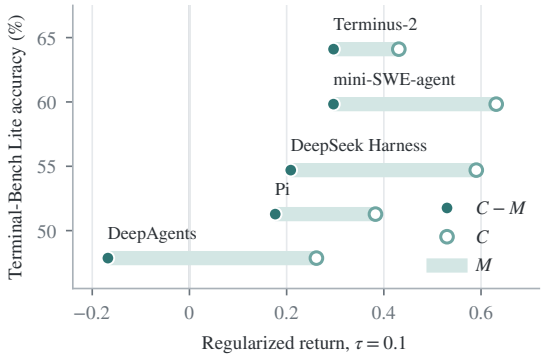

Figure 10: Ceiling and mismatch of five coding-agent harnesses for a frozen Qwen3.5-9B. Estimates of Table 2 on three Terminal-Bench Lite probe tasks with $\tau = 0.1$. Each harness sits at its accuracy over all 39 tasks. The open circle marks $\widetilde{C}$, the filled circle marks $\widetilde{C} - \widetilde{M}$, and the bar between them has length $\widetilde{M}$.

Estimators. At a recorded state s under harness h, let $p_h(\cdot \mid s)$ and $q_h(\cdot \mid s)$ be the target's and the teacher's category distributions, estimated from 128 and 64 samples with a pseudocount of 0.5 per category. The objective uses the reference p_h, a binary terminal reward Y, and no discount. The ceiling estimate averages the teacher's regularized return over its complete trajectories $s_0, \ldots, s_{L-1}$:

$$\widetilde{C}_h = \mathbb{E}_{q_h}\Big[Y - \tau \sum_{t=0}^{L-1} \mathrm{KL}\big(q_h(\cdot \mid s_t) \,\big\|\, p_h(\cdot \mid s_t)\big)\Big]. \tag{31}$$

The mismatch estimate applies Proposition 6.1 with q_h in place of the optimum. Without discounting, the factor $1/(1-\gamma)$ becomes a sum over the decisions of each complete target trajectory:

$$\widetilde{M}_h = \frac{1}{4\tau}\, \mathbb{E}_{p_h}\Big[\sum_{t=0}^{L-1} \Big\| \tau P_\perp \log \frac{p_h(\cdot \mid s_t)}{q_h(\cdot \mid s_t)} \Big\|_2^2\Big]. \tag{32}$$

We use $\tau = 0.1$. With p_h as the reference, the target's regularized return equals its success probability, so the exact ceiling minus mismatch also equals it. The estimates put a teacher that need not be optimal in place of the optimum, so $\widetilde{C}_h$ estimates a lower bound on the ceiling and $\widetilde{M}_h$ measures distance to the teacher. We therefore compare the ordering of $\widetilde{C}_h - \widetilde{M}_h$ with the ordering of accuracy. The mismatch per decision in Figure 6 is $\widetilde{M}_h$ divided by the mean number of decisions per complete target trajectory. Table 13 lists the three estimates and the decisions per trajectory, and Figure 10 plots the estimates against accuracy.

Table 13: Terminal-Bench Lite probe results at $\tau = 0.1$. Decisions are the mean number of model calls per complete target trajectory.

Harness	Decisions	$\widetilde{C}_h$	$\widetilde{M}_h$	$\widetilde{C}_h - \widetilde{M}_h$
Terminus-2	4.9	0.431	0.134	0.297
mini-SWE-agent	8.6	0.631	0.335	0.296
DeepSeek Harness	8.6	0.590	0.382	0.208
Pi coding-agent	6.6	0.383	0.206	0.177
DeepAgents	14.2	0.261	0.429	−0.168

Action shares. Table 14 gives the mean share of each category at the recorded states of complete target trajectories, for the target and for the teacher sampled at the same states. The remaining categories each hold at most 7.2% for either model. Across all five harnesses, the target inspects more often than the teacher. It also marks the task complete less often, except under Terminus-2, where the two do so at similar rates.

Table 14: Action category shares (%) at the states of complete target trajectories on the Terminal-Bench Lite probe tasks, target / teacher.

Harness	States	Inspect	Modify	Plan	Stop	Opaque	Invalid
Terminus-2	44	37.7 / 30.3	15.8 / 20.8	0.0 / 0.0	38.3 / 35.1	8.2 / 6.4	0.0 / 0.1
mini-SWE-agent	77	59.7 / 43.5	14.1 / 11.3	0.0 / 0.0	11.7 / 22.4	14.1 / 18.9	0.1 / 0.0
DeepSeek Harness	69	33.3 / 20.0	12.1 / 15.0	15.1 / 26.5	11.2 / 14.8	27.7 / 18.6	0.1 / 2.7
Pi coding-agent	46	48.3 / 37.0	20.4 / 22.4	0.0 / 0.0	12.9 / 21.0	18.0 / 16.6	0.0 / 0.4
DeepAgents	85	27.9 / 24.8	11.8 / 16.8	11.1 / 11.4	3.9 / 8.4	38.2 / 32.4	3.5 / 0.0

F.8 Exact Harness Scores and Task Success

Tasks and harnesses. We evaluate H0–H5 from Table 4 on the fixed test set of 576 tasks, with 192 tasks from each of the three MiniWoB++ families in Appendix F.1. We reuse their compiled execution graphs, whose transitions and successful task completion were checked against the native browser. The horizon is ten actor decisions, with remaining budget included in the state. H4 retains the archived decisions of its separate frozen Qwen3.5-2B reviewer.

Frozen model and semantic policy. The actor is off-the-shelf Qwen3.5-9B, with fixed weights and no project adapter. Each prompt exposes the harness's complete legal action menu, with a single-token code for every parameterized semantic action. We obtain the action distribution by normalizing the corresponding token logits at temperature one, using BF16 inference, FP64 normalization, and a state microbatch of eight. Thinking is disabled. The objective instruction asks the model to maximize exact task completion within its decision budget. Actor probabilities are computed anew for this model under every harness. We set each harness's reference to this same frozen semantic distribution and hold it fixed when solving for the optimal policy.

Objective and exact computation. We use the finite-horizon counterpart of Eq. (2), with $\gamma = 1$, $\tau = 0.1$, binary terminal success reward, and zero explicit execution costs. This objective and reference define a new control problem for each harness, so the ceilings differ from those of the post-training study in Section 3.2. Backward induction on the acyclic graph gives $\Pi^*_{\mathcal{I}}$ and the initial-state optimal value C. Let $\nu^{\Pi}_{\mathcal{I}}(s) = \sum_{t=0}^{9} \Pr^{\Pi}_{\mathcal{I}}(s_t = s)$ be the expected number of visits to a nonterminal state. Forward propagation under the evaluated policy supplies these visitation weights, giving

$$M(\theta, \mathcal{I}) = \tau \sum_{s} \nu^{\Pi_{\theta,\mathcal{I}}}_{\mathcal{I}}(s) \mathrm{KL}(\Pi_{\theta,\mathcal{I}}(\cdot \mid s) \| \Pi^*_{\mathcal{I}}(\cdot \mid s)) \,. \tag{33}$$

The Bellman residual telescopes over the finite horizon to give $C - J = M$. Since the evaluated policy equals its fixed reference, its regularization penalty is zero, and J equals task success probability. We compute J by forward reward accumulation and success by a separate backward recursion. These independent computations agree with $C - M$ to numerical precision. The largest observed identity residual is 7.22×10^{-16}.

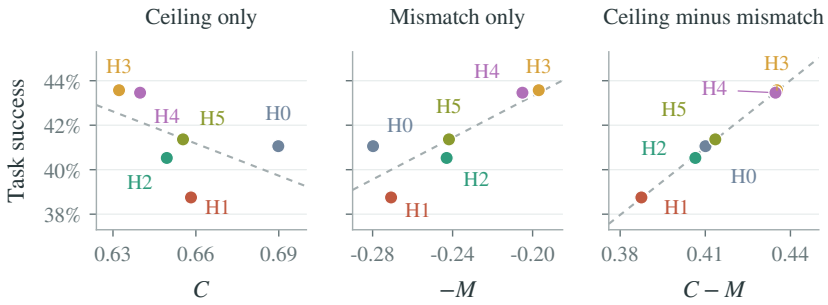

Figure 11: Harness scores compared with task success. Each point is one harness evaluated with the same frozen Qwen3.5-9B on a shared task set. The panels compare C, $-M$, and $C - M$ with rollout success. Dashed lines show linear fits.

Rollout evaluation and aggregation. For each task and harness, we draw 256 independent trajectories from the measured actor distribution on the compiled graph. The fraction that reaches native task success supplies the vertical coordinate in Figure 11.

Scores and success are averaged within each family and then equally across families. Each point therefore compares the same task distribution. The three selection rules maximize C, $-M$, and $C - M$, respectively. The dashed lines are unweighted linear fits through the six displayed harness means. All actor evaluations run on four RTX 4090 GPUs.

For these deterministic task graphs with binary terminal reward, an individual task's reference success probability S gives $C = \tau \log(1 + S(e^{1/\tau} - 1))$. This is increasing in S, but averaging this nonlinear quantity across tasks can order harnesses differently from averaging success. In the aggregate comparison, ceiling selects H0, while both negative mismatch and $C - M$ select H3. Negative mismatch ranks H1 above H0, whereas exact and sampled success rank H0 above H1. Combining ceiling and mismatch gives the complete ordering of the exact mean success probabilities.