PROPOSE BEFORE YOU PLAN: INFERRING CHANGED LEGALITY RULES FROM REFUSALS

Anonymous authors
Paper under double-blind review

ABSTRACT

When a deployed policy must learn which actions are legal from refusals, the usual lever is better probing: choosing informative actions and planning them ahead. On legality rules from third-party codebases, a policy frozen across a rule change trails a specialist retrained on the new rule. Our deployable policy closes 0% of that gap on average. We split the policy into proposing rule hypotheses, updating a belief from refusals and choosing actions, and vary one stage at a time. Planning probes ahead adds nothing measurable over pricing them one at a time, even with exact belief updates. An explicit information term adds nothing significant over exploiting the belief. We test proposal by planting a chosen rule among the hypotheses. Pooled over rule shapes, a near-copy of the deployed rule, wrong in one machine or command class, retains 60% of what the deployed rule itself gains over a wrong rule. The policy's own hypotheses stay further from the deployed rule than that near-copy, which places the bottleneck at proposal. This pooled result holds under a grammar that can express the deployed rules. For such policies, better proposal should come before better planning, and planting gives proposers a measurable target.

1 INTRODUCTION

Deployed systems refuse actions by rules the acting policy never sees. A tool server refuses a call whose permission check fails, and an agent runtime refuses a shell command that a safety rail flags, such as a fetch-then-execute command on a guarded machine. The policy learns about the rule only through one bit per attempt, accepted or refused, and a refused attempt still spends a step. When the rule changes at deployment, the policy has to work out the new rule from those bits. We take rules from three independently authored codebases: a tool server's permission checks, an agent runtime's safety rails and a rule-inference game. Each source supplies rules of several logical forms, or shapes, such as a conjunction of two conditions. On each of them, a specialist retrained on the new rule beats a policy trained under the old one, the generalist, by up to 22.3 percentage points (pp) of balanced accuracy (Section 2.3). We grade a policy that runs without retraining by the share of this gap it closes.

The usual remedy is to make the policy a better experimenter. Active learning, cost-sensitive value of information and Bayes-adaptive planning treat the choice of query as the lever (Krause & Guestrin, 2005; Golovin & Krause, 2011; Ross et al., 2007). They pick the action whose outcome is most informative, price it by its cost, and plan several queries ahead when their order matters. Our setting looks like the case where planning should matter most, because each query is a played action that changes which actions remain. Yet the deployable policy we build on this idea closes 0% of the gap on average (Section 4.1). The loss could sit in two other places. The policy may fail to act on what it has learned, or it may lack a hypothesis close to the deployed rule.

We locate the loss by splitting a deployable policy into the three stages of Figure 1 and changing one stage while we hold the other two fixed. The policy's weighted rule hypotheses form its belief, a posterior over rules. Its decision stage is a lookahead that plans probes over belief and state together, and we compare it with pricing each probe one step at a time. To test proposal we *plant* a chosen rule among the hypotheses and protect it from being replaced. Because we grade all variants of the

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

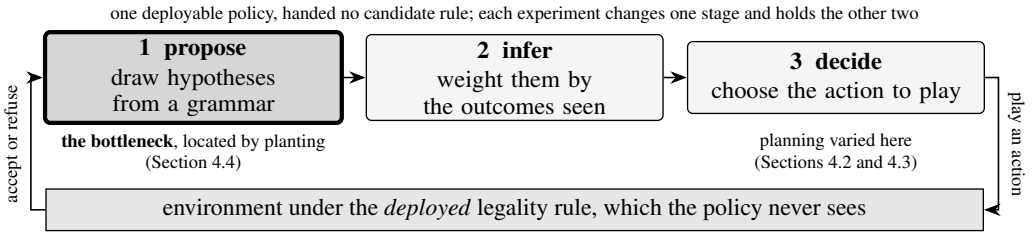

Figure 1: **Pooled over six rule shapes, the deployable policy loses at proposal.** Each comparison changes one stage of the loop and holds the others fixed. Varying the decide stage shows no measurable gain from planning the probes. Planting rules at the propose stage shows that a hypothesis near the deployed rule would help, and that the policy's own hypotheses are further off.

policy on the same states, visited by a random policy, a difference in score reflects what a variant has inferred about the rule, with rule shapes as the statistical unit.

For the planners, grammars and episode lengths we test, the policy loses at proposal. Planning the probes two steps ahead gives no measurable gain over pricing each probe on its own, even with exact belief updates that revise each hypothesis after each imagined outcome. The same runs detect the lookahead's gain over random play, which picks actions uniformly at random, and their interval against one-step pricing excludes any planning gain half that size. An explicit information term, which rewards probes on which the hypotheses disagree, adds nothing significant over exploiting the belief by playing the action of highest expected value under it. Pooled over six shapes, a planted copy of the deployed rule with one guarded machine or command class swapped retains 60% of what planting the deployed rule itself gains over planting a wrong rule. A hypothesis close to the deployed rule would thus carry most of that gain, and the policy's own hypotheses stay further from the deployed rule on both rule sources. Proposal remains the limit under a richer grammar that can express the deployed rules. Our contributions:

- Planting, a diagnostic that places the loss at proposal over six pooled shapes (Section 4.4).
- Exploiting the belief keeps 90–93% of the gain over a belief-blind ablation (Section 4.3).
- Two-step probe planning gains nothing measurable over one-step pricing (Section 4.2).
- ACTIONSET-SHIFT: 38 legality-rule shifts from three codebases (Section 2).

2 LEGALITY RULES REVEALED BY REFUSALS

2.1 INTERFACE AND SCORE

The policy predicts which actions are legal while seeing only whether its own attempts are accepted. An episode draws a hidden vector of k bits, $z \in \{0,1\}^k$, for instance which of $k = 12$ machines are in an outage. At each of H steps the policy picks one of the available actions. A legal action succeeds and pays its value $v(a)$, which the policy sees, and an illegal one is refused and pays nothing. Legality is a deterministic function $\mathrm{legal}(a, z, s)$ of the action, the hidden vector and the observable state s. The policy sees s and its own refusals, never z or the rule. A refusal carries only the bit, without an error message. An accepted action is consumed and a refused one stays available, and either attempt spends one of the H steps. We map the two rule sources of the policy experiments (Section 2.2) onto one world of machines, command classes and outage bits, with 36 actions per episode, three on each of the twelve machines.

The object to recover is the feasible set, the available actions that are currently legal. At each step the policy predicts legality for each available action. We score balanced accuracy, the mean of the true-positive and true-negative rates, over the steps at which the available actions include both a legal and an illegal one. Chance is 50%. Numbers below are this quantity for episodes of $H = 40$ attempts unless stated, averaged over the episodes of each (rule, seed) pair, where a seed fixes the randomness of one run.

rule source	rules taken from	legality decided by	shapes	shifts	rate-matched
permission	GitHub's MCP server	encoded predicates	3	12	yes
safety rails	OpenHands agent SDK	executed upstream code	3	12	yes
authored rules	GOHR corpus	its authors' rule files	–	14	no
all				38	

Table 1: **ACTIONSET-SHIFT holds 38 ordered legality-rule shifts from three third-party codebases.** A shift pairs a training rule with a deployed rule of the same shape; a rate-matched pair refuses the same fraction of actions. The permission and safety-rail sources carry the policy experiments, and the authored rules of the Game of Hidden Rules (GOHR) only measure the gap (Appendix C).

rule source *(unit)*	scored where it acts	value-greedy trajectory	uniform-random trajectory
permission, 12 pairs *(pair)*	+22.3 [+19.0, +25.4]	+15.8 [+11.7, +19.8]	–
permission, 9 rules *(shape)*	+21.8 [+17.4, +24.6]	+17.4 [+13.5, +20.5]	+18.0 [+10.5, +22.3]
GOHR, 14 changes *(file)*	+9.7 [+8.4, +11.4]	+13.7 [+11.6, +15.5]	+18.2 [+16.4, +21.1]
rails, 12 pairs *(shape)*	+20.7 [+17.8, +22.8]	+19.8 [+17.3, +22.6]	+16.7 [+15.0, +19.0]
rails, tuned network *(shape)*	+21.3 [+20.1, +22.5]	+17.7 [+15.0, +19.8]	+17.5 [+15.7, +19.7]

Table 2: **A policy frozen across a legality-rule change trails a retrained specialist on each rule source and under each grading we ran.** Each cell is the specialist minus the generalist in pp at $H = 40$, with a 95% interval that resamples the unit in italics: a rule pair, a rule shape or a GOHR rule file. The first column scores each predictor on the states its own actions reach; the other two score all predictors on one trajectory that no predictor chose, played by a policy that always tries the highest-value action (value-greedy) or a uniformly random one. The second rail row (tuned network) uses a network architecture selected on training loss alone; a dash marks a grading not run.

On the safety-rail rules, an action is a shell command sent to one of the twelve machines, in one of four command classes, and each machine offers three such commands per episode. Whether a rule refuses the command depends on the machine, the class and whether that machine is in a hidden outage, and each rule names the machines and classes it guards. A rule change keeps the rule's shape and changes those sets. The policy sees the machine, the class and its refusals, and it outputs one legal-or-illegal prediction per available action, scored against the verdict of the upstream rail code.

2.2 THREE THIRD-PARTY RULE SOURCES

The rules come from three independently authored codebases (Table 1). We encode the permission predicates of one, execute the safety-rail code of another in our loop, and take the third's rule files as written. From GitHub's Model Context Protocol (MCP) server (GitHub, 2025), a tool server for language-model agents, we read permission predicates of three shapes, a single lookup, a conjunction and a disjunction. From the OpenHands software agent SDK (Wang et al., 2025) we take three shapes of safety rail, fetch-then-execute, raw-disk and catastrophic-delete. On both sources the two rules of a train-and-deploy pair refuse the same fraction of actions, which keeps a change of rule from being a change of strictness. The released shifts pair consecutive rules of a shape, twelve pairs per source, and the policy experiments deploy nine held-out rules per source, three per shape. From the Game of Hidden Rules (GOHR) corpus (Pulick et al., 2022), a rule-inference board game, we take fourteen changes between consecutive rules of the authors' own experimental sequences. Its rules differ in how often they refuse, and we use GOHR only to measure the gap.

2.3 THE COST OF A RULE CHANGE

On all three sources, a policy frozen across a rule change trails a policy retrained on the new rule (Table 2). The *generalist* is trained under a shift's training rule and deployed under its deployed rule, and the *specialist* is retrained under the deployed rule. Both use the same transformer, observation interface and seeds, and both train on the same amount of data from a policy that tries actions uniformly at random. Their gap therefore measures what training-time exposure to the deployed rule is worth. The specialist is a learned reference, which a policy handed the right candidate rules can

exceed (Appendix F). A method's share of the gap is its score minus the generalist's, divided by the specialist's score minus the generalist's, computed for each (rule, seed) pair and averaged.

Scored on the states each predictor reaches, the gap runs from $+9.7$ pp on the authored GOHR changes to $+22.3$ pp on the permission rules. All intervals in Table 2 exclude zero, including those that score all predictors on one trajectory that no predictor chose. The gap survives seven ways of strengthening the generalist, among them a memory carried across episodes and a larger network (Appendix D).

2.4 A COMMON SCORING TRAJECTORY

We grade all variants of the policy on the same states. Accuracy is scored on the actions still available, and two variants that act differently would otherwise be graded on different states. Each variant plays its own actions in one instance of the episode and updates its belief from the refusals there. A second instance with the same hidden vector is advanced by a uniformly random policy, identically for all variants, and we call its states the common *scoring trajectory*. Because the belief is over rules, it predicts legality in any state, and at each step we score it on that instance's available actions.

We pool the six shapes of the permission and safety-rail sources, which leaves five degrees of freedom for a Student interval over shapes. We call a difference significant only when two 95% intervals exclude zero, a bootstrap over shapes and a Student t interval over the shape means. Appendices C and E verify the rule encoding against the executed rail code and check the grading pipeline.

3 A DEPLOYABLE POLICY IN THREE STAGES

The policy infers the deployed rule without retraining or a candidate list. Its three stages follow, and each comparison in Section 4 changes one of them.

Propose. The policy is handed the rule grammar, a set of primitives, each a condition on one attribute of the action or of the hidden vector. A hypothesis refuses an action when a conjunction of primitives holds. The grammar is the only knowledge of the deployed rule the policy is given. At the start of each episode it draws 24 fresh hypotheses, each a conjunction of two or three primitives, where each primitive tests membership in a random subset of its attribute's values, such as a set of machines. A deployed rule refuses when any of its one to four conjunctions holds, some of them a single primitive. On five of the six shapes no single hypothesis can express the deployed rule, and on the sixth only a hypothesis that names exactly its guarded sets can (Appendix C). Section 4.4 repeats the diagnosis with a richer grammar that can express all six. After each block of five steps, a resampling step replaces the hypotheses below the median likelihood, mostly by perturbed copies of well-fitting ones and otherwise by fresh draws.

Infer. A hypothesis h_j decides an action's legality either outright or through one hidden bit, the outage bit z_m of the action's machine m. We write $\ell_j(a, b)$ for its verdict when $z_m = b$, 1 if legal. For each hypothesis the policy keeps a weight w_j and a posterior $q_j(m)$ that $z_m = 1$, which starts at the world's known outage rate. The probability that action a is legal is

$$p(a) = \sum_j w_j \big[\, q_j(m)\, \ell_j(a, 1) + (1 - q_j(m))\, \ell_j(a, 0) \,\big]. \tag{1}$$

After an outcome at a, Bayes' rule updates w_j by the likelihood marginalised over z_m and updates $q_j(m)$ by the same outcome. The likelihood allows probability 0.05 of an unexplained outcome, which keeps each hypothesis's weight above zero. The policy predicts an action legal when $p(a) > 1/2$.

Decide. Only this stage differs among the variants of Sections 4.2 and 4.3. Let $\sigma(a) = \min\{p(a), 1 - p(a)\}$ measure how evenly the hypotheses split on a, and let $c(a) = 1 + (1 - p(a)) \max\{v(a), 0\}$ be the expected cost of trying it, one for the step it spends plus the value it fails to collect on that step if refused. The split $\sigma(a)$ follows the query criterion of the constraint-acquisition algorithm QuAcq2 (Bessiere et al., 2023); it is not an expected information gain over the rule space. The *belief-blind ablation* plays $\arg\max_a v(a)$, the highest-value action, and ignores the belief, and after a refusal it plays the same action again. It keeps the inference and removes

informed action choice. It is also the value-greedy policy of Table 2. The exploit strategy plays $\arg\max_a p(a)v(a)$, the action of highest expected value, with no information term. One-step pricing maximises σ/c, uncertainty priced by cost in the spirit of cost-sensitive value of information (Krause & Guestrin, 2005). Our method, the lookahead, adds the value of the best probe that remains after each outcome of the current one:

$$L_d(a) = \frac{\sigma(a)}{c(a)} + \sum_{y \in \{\text{accept},\, \text{refuse}\}} P(y \mid a) \max_{a'} L_{d-1}(a' \mid a, y), \qquad L_0 = \sigma/c. \tag{2}$$

Here $P(\text{accept} \mid a) = p(a)$, and $L_{d-1}(\cdot \mid a, y)$ is the same score under the posterior that outcome y induces, over the actions still available. Depth d plays $\arg\max_a L_d(a)$ and so looks d probes past the current one. We run depth 2 unless stated, and at each depth a node expands only the six of the 36 actions with the highest one-step score, which keeps depth 3 affordable.

The exact-update lookahead revises each hypothesis's weight and hidden-bit posterior after each hypothetical outcome by the rule above. The default lookahead is cheaper. It treats a verdict that depends on a hidden bit as legal with probability one half and leaves that hypothesis's weight unchanged. Both hold each hypothesis's verdicts fixed at the current state and carry forward only which actions remain available as imagined probes are accepted (Appendix A). A last variant, random play, chooses actions uniformly at random while its belief still updates. We fixed the scoring trajectory, the rule by which Section 4.4 reads the planting experiment and five success criteria, among them that the lookahead would beat one-step pricing, before reading the results they grade. Appendix F gives the five criteria and their outcomes.

4 Where the gap is lost

4.1 Share of the gap closed

No deployable variant reaches the retrained specialist. The lookahead closes 0% of the gap, averaging its share over (rule, seed) pairs (Appendix F), and its difference from the generalist has a Student interval of $[-8.9, +12.4]$ pp (Figure 2).

On the safety-rail rules alone the generalist scores above the lookahead (Table 3 in Appendix F). Both comparisons with the trained predictors are unpaired, because we measured the generalist and specialist in separate runs whose episodes offer 120 available actions, against 36 for the policy variants.

Ports of two published in-context learners score below the generalist on the safety-rail rules (Table 11 in Appendix J). They are a recurrent policy with memory across episodes, RL^2 (Duan et al., 2016), and a policy conditioned on a variational task posterior, VariBAD (Zintgraf et al., 2020), each trained on its own objective. A planted control shows that the VariBAD port fails even when the first legal action reveals the answer, and we therefore draw no conclusion about in-context learning from these ports.

4.2 Limited benefit from deeper probe planning

Planning the probes two steps ahead changes balanced accuracy by $+0.3$ pp over one-step pricing, with a Student interval of $[-0.8, +1.4]$ pp (Figure 2). The same runs detect the lookahead's gain over random play as significant (Section 4.3).

With exact belief updates inside the search, the lookahead chooses a different action from the default one in 47% of decisions (Appendix A). Its interval against one-step pricing is $[-1.0, +1.3]$ pp, while the exact-update lookahead's gain over random play in the same runs, $+3.2$ pp, is significant. The interval excludes planning gains larger than half the gain over random play and leaves smaller ones open.

Expanding all 36 actions keeps the exact-update lookahead within the same bound (Figure 2). At depth 3 on the safety-rail rules, the default lookahead is not significant against one-step pricing and takes about $40\times$ the one-step running time (Appendix G). Its probes also recover fewer bits of information about the rule than passive probing at random, on a rule generator of our own (Appendix H).

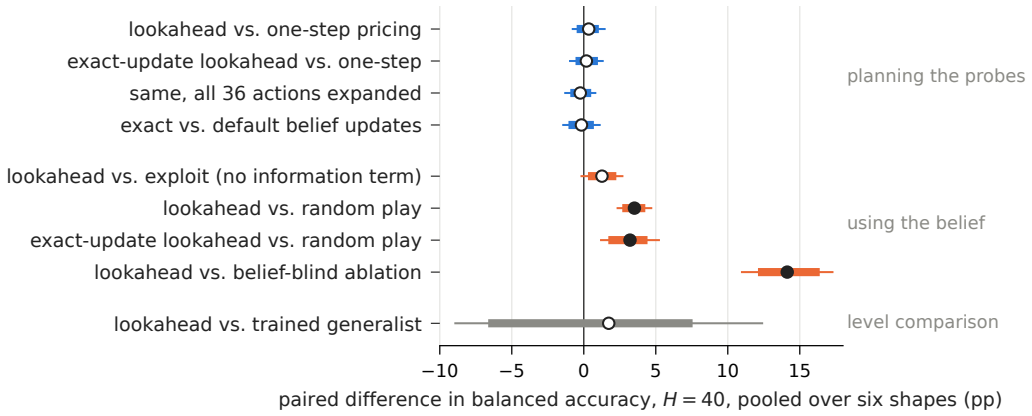

Figure 2: **Planning the probes adds nothing measurable over one-step pricing, even with exact belief updates; the significant gains are over references whose actions ignore the belief.** Paired differences in balanced accuracy at $H = 40$: dot = mean, thick bar = 95% bootstrap over shapes, thin bar = 95% Student t interval over the shape means; a filled dot is significant under both. The exact-update rows come from separate runs paired only with each other, and the generalist row compares levels from separate runs. Table 3 in Appendix F gives each value and each source's mean.

4.3 Acting on the belief

Choosing actions by the belief beats both references whose actions ignore it. Against the belief-blind ablation the lookahead gains $+14.1$ pp (Figure 2). Random play is a second reference whose actions also ignore the belief but vary instead of repeating a refused action, which tests whether any varied choice of action would do. Against random play the lookahead gains $+3.5$ pp, and against the generalist its difference is not significant.

The exploit strategy acts on the belief with no information term and keeps 93% of the lookahead's gain over the belief-blind ablation on the safety-rail rules and 90% on the permission rules, by the per-source differences of Table 3. The lookahead's remaining advantage over the exploit strategy is small, and its Student interval covers zero (Figure 2).

4.4 Planting locates the loss at proposal

Planting. As a diagnostic, planting hands the policy a chosen rule and measures how much it loses by having to propose its own hypotheses. We overwrite one of the policy's 24 initial hypotheses with the chosen rule and protect it from resampling, and we keep the other hypotheses and all random numbers identical. The planted rule is the deployed rule itself (the exact rule), a copy with one, two or four of its guarded machines or classes (its elements) swapped for unguarded ones, or a wrong rule that refuses exactly as often. Each is written directly as a rule and can take forms the grammar cannot draw, and we measure each gain against the wrong rule.

Suppose the copy with one element swapped retains, as a point estimate, at least half of the exact rule's gain, and its own gain is significant. Then a close hypothesis is nearly as useful as the exact one, and we read the limit as proposal. If the copy retains under a quarter, the limit is telling close rules apart. If the exact rule's own gain is not significant, the limit lies downstream of proposal, in inference or action choice.

Pooled over the six shapes, planting the exact rule and planting the copy with one element swapped both give significant gains (Figure 3a). The copy retains 60% of the exact rule's gain, with a bootstrap interval of [37%, 77%], which places the limit at proposal. Copies further off gain nothing significant, and the same reading holds under each strategy alone (Table 8 and Appendix I).

Each source alone has three shapes. At two degrees of freedom the exact rule's gain, which varies several-fold across shapes, has a Student interval that covers zero. By the same rule each source alone

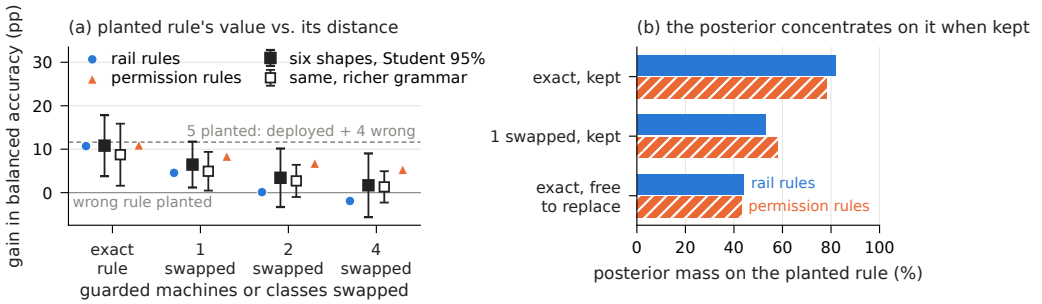

Figure 3: **A planted rule's value falls with its distance from the deployed rule under either grammar, and the posterior concentrates on the planted deployed rule when it is protected.** (a) Gain in balanced accuracy at $H = 40$ over the same policy with a planted wrong rule that refuses exactly as often, averaged over the exploit, one-step and lookahead strategies. Filled squares pool six shapes with a Student 95% interval; dots give each source's mean; hollow squares use the richer grammar of Section 4.4. The dashed line plants the deployed rule together with four wrong rules. (b) Posterior mass on the planted rule over the last ten steps, under the lookahead. Tables 8 and 10 in Appendix I give each value with its intervals.

places the limit downstream of proposal, under this grammar and under the richer one below. We therefore claim the proposal verdict only for the six pooled shapes.

The policy's own hypotheses stay further from the deployed rule than the copy with one element swapped. Over the last ten steps the closest of them disagrees with the deployed rule on 18% of the scored safety-rail actions, against 8% for that copy, and the permission rules show the same order (Appendix I). The policy's own resampling step can also discard a planted rule that it is free to replace (Table 8).

Once a rule is proposed and protected, the belief concentrates on it (Figure 3b). A planted deployed rule holds 82% of the posterior mass on the safety-rail rules. With the exact rule among its hypotheses, the policy still scores -4.1 pp against the same inference given only the deployed rule and four wrong ones, a cost of telling close rules apart.

A richer grammar. Proposal remains the limit when the grammar can express the deployed rules. In the richer grammar a hypothesis refuses when any of one to four conjunctions of one or two primitives holds, which covers all nine deployed rules of each policy source. We repeat planting under it, with exact belief updates in the lookahead, and compare each planted rule with a planted wrong rule under the same grammar and updates. The copy with one element swapped retains 56% of the exact rule's gain, with a bootstrap interval of [43%, 70%] (Figure 3a, hollow squares). Because we sized this grammar to the deployed shapes, its gain with nothing planted (Table 10) reflects prior knowledge of the rule family.

Count and targeting. Drawing more hypotheses is the one deployable change to proposal with a significant gain. Drawing 128 hypotheses instead of the default draw gains $+3.5$ pp on the safety-rail rules and $+4.4$ pp on the permission rules. The larger draw brings the policy close to the generalist, and a further fourfold increase is not significant, with an interval that does not show saturation (Table 9).

Changing which hypotheses are drawn gains nothing significant on the safety-rail rules under any of four operators, such as rejecting draws that past refusals contradict. The operators reach most of the belief, since hypotheses added by resampling hold 67% of the posterior mass (Appendix I). Consistent with the planting result, changing which refusal-consistent hypotheses the policy holds barely moves its predictions unless one of them is close to the deployed rule.

5 Related work

Methods that choose or plan queries assume that asking leaves the world unchanged, and methods that infer a constraint assume the target is among their hypotheses. Our setting breaks both assumptions. Each probe is a played action that changes what remains, and the policy must propose its own hypotheses.

Choosing and planning queries. Cost-sensitive value of information (Krause & Guestrin, 2005), adaptive submodularity (Golovin & Krause, 2011) and Weitzman's rule (Weitzman, 1979) price queries from a fixed set whose inspection leaves the world unchanged. Bayes-adaptive planning (Ross et al., 2007) and sample-based tree search (Silver & Veness, 2010; Guez et al., 2012) let queries move the state. Zhu et al. (2026) plan a few steps ahead over a belief, and Osband (2026) price exploration with a horizon term. In both, a query leaves the task state unchanged, while each of our probes moves it.

Inferring a constraint or a feasible set. Constraint acquisition, version spaces and membership queries identify a target from free queries (Mitchell, 1977; Bessiere et al., 2023; Angluin, 1988). Noisy generalized binary search assumes the target is among its hypotheses (Golovin et al., 2010), and planting measures what is lost when the policy must propose them itself. Inverse constraint learning needs demonstrations with known reward and dynamics (Scobee & Sastry, 2020; Malik et al., 2021). Action elimination learns from our refusal bit under one fixed notion of validity (Zahavy et al., 2018), and masking, affordance and stochastic-action-set methods take availability as given (Huang & Ontañón, 2022; Khetarpal et al., 2020; Boutilier et al., 2018; Chen et al., 2026). Seo et al. (2026) induce transition programs and name repairing proposals from rejected candidates as the remedy. Rejecting drawn hypotheses that past refusals contradict is one such repair, and it gains nothing significant here. Zha et al. (2026) refine a rule document from its errors on a labelled validation set, a richer signal than our one refusal bit per action.

In-context adaptation, and grading it. In-context learners adapt from a carried history (Duan et al., 2016; Zintgraf et al., 2020; Laskin et al., 2023), including to action sets unseen in training (Sinii et al., 2024), but none targets a legality rule revealed only by refusal. A change of grading distribution can reorder methods (Huang & Jiang, 2022; Suau et al., 2024), and we state which distribution grades each result. Appendix K places each line of work by what it assumes.

6 Limitations

Our findings hold inside a simulated world, action set and episode budget of our own design. The gap is measured on legality rules from three third-party codebases and the policy findings on two, and we drive no live agent runtime. The planning result covers depth-two lookaheads with default or exact belief updates and a depth-three lookahead with default updates on the safety-rail rules. All of them hold each hypothesis's verdicts at the current state, and sample-based tree search over the belief (Silver & Veness, 2010; Guez et al., 2012) lies outside the result. The planted copies keep the number of guarded elements but not the exact refusal rate. The proposal result concerns proposers that draw from a fixed grammar within one episode.

7 Conclusion

Pooled over the six rule shapes we test, what limits a policy that infers a changed legality rule from refusals is the set of hypotheses it proposes. Under both grammars we tested, planting places the limit at proposal, and the policy's own resampling can discard a good hypothesis once found. The planners we tested add nothing measurable to make up for it. For a builder, the finding suggests an order of work: keep the belief connected to the action, propose hypotheses near the rule and keep them once found, and only then refine which action to try next. Planting gives the next proposer a measurable target: how much of the exact rule's gain its own hypotheses recover.

AI USE STATEMENT

AI assistants based on large language models were used for part of the execution of this work, under the authors' direction: implementing the environment generator, the methods and the baselines. All research decisions were made by the authors, who verified every AI-assisted output and every reported result. The authors take full responsibility for the content of this paper.

REPRODUCIBILITY STATEMENT

Every result is reproducible from a seeded generator and the third-party rule sources, so the anonymized code supplement releases the rule shifts as code that rebuilds them, not as data files. It contains the rule generator, the code that builds the 38 rule shifts, and every policy variant, planting condition and comparator the paper reports. It also holds one configuration per experiment family with its seeds, and the scripts that emit every table and data figure from the scored results. Its README maps each table and figure to the commands that regenerate it. The upstream rail code and the authors' VariBAD are vendored at pinned revisions under their MIT licenses, and the GOHR files, which carry no license, are fetched and verified against checksums. Section 2 and Appendix B give the configuration, budgets, seeds and statistical units, Appendix A gives every decision strategy in full, and Appendix C describes each rule source and how its rules are encoded.

REFERENCES

Dana Angluin. Queries and concept learning. *Machine Learning*, 2:319–342, 1988.

Rohith Reddy Bellibatlu, Manpreet Singh, Deepak Parashar, and Rahul Joshi. Counterfactual fairness audits of multi-step clinical LLM agents require a measured per-action instability floor. arXiv preprint arXiv:2609.03221, 2026.

Christian Bessiere, Remi Coletta, Barry O'Sullivan, and Mathias Paulin. Query-driven constraint acquisition. In *International Joint Conference on Artificial Intelligence*, 2007.

Christian Bessiere, Remi Coletta, Emmanuel Hébrard, George Katsirelos, Nadjib Lazaar, Nina Narodytska, Claude-Guy Quimper, and Toby Walsh. Constraint acquisition via partial queries. In *International Joint Conference on Artificial Intelligence*, 2013.

Christian Bessiere, Clément Carbonnel, Anton Dries, Emmanuel Hébrard, George Katsirelos, Nadjib Lazaar, Nina Narodytska, Claude-Guy Quimper, Kostas Stergiou, Dimosthenis C. Tsouros, and Toby Walsh. Learning constraints through partial queries. *Artificial Intelligence*, 319:103896, 2023.

Craig Boutilier, Alon Cohen, Amit Daniely, Avinatan Hassidim, Yishay Mansour, Ofer Meshi, Martin Mladenov, and Dale Schuurmans. Planning and learning with stochastic action sets. In *International Joint Conference on Artificial Intelligence*, 2018.

Yi Chen, Rushuai Yang, Qiang Chen, and Dongyan Huo. Bellman-Taylor score decoding for Markov decision processes with state-dependent feasible action sets. arXiv preprint arXiv:2606.10979, 2026.

Simone Drago, Marco Mussi, and Alberto Maria Metelli. Sleeping reinforcement learning. In *International Conference on Machine Learning*, 2025.

Yan Duan, John Schulman, Xi Chen, Peter L. Bartlett, Ilya Sutskever, and Pieter Abbeel. RL^2: Fast reinforcement learning via slow reinforcement learning. arXiv preprint arXiv:1611.02779, 2016.

GitHub. GitHub MCP server. `https://github.com/github/github-mcp-server`, 2025.

Daniel Golovin and Andreas Krause. Adaptive submodularity: Theory and applications in active learning and stochastic optimization. *Journal of Artificial Intelligence Research*, 42:427–486, 2011.

Daniel Golovin, Andreas Krause, and Debajyoti Ray. Near-optimal Bayesian active learning with noisy observations. In *Advances in Neural Information Processing Systems*, 2010.

Arthur Guez, David Silver, and Peter Dayan. Efficient Bayes-adaptive reinforcement learning using sample-based search. In *Advances in Neural Information Processing Systems*, 2012.

Audrey Huang and Nan Jiang. Beyond the return: Off-policy function estimation under user-specified error-measuring distributions. In *Advances in Neural Information Processing Systems*, 2022.

Shengyi Huang and Santiago Ontañón. A closer look at invalid action masking in policy gradient algorithms. In *International Florida Artificial Intelligence Research Society Conference*, 2022.

Khimya Khetarpal, Zafarali Ahmed, Gheorghe Comanici, David Abel, and Doina Precup. What can I do here? A theory of affordances in reinforcement learning. In *International Conference on Machine Learning*, 2020.

Andreas Krause and Carlos Guestrin. Near-optimal nonmyopic value of information in graphical models. In *Conference on Uncertainty in Artificial Intelligence*, 2005.

Michael Laskin, Luyu Wang, Junhyuk Oh, Emilio Parisotto, Stephen Spencer, Richie Steigerwald, DJ Strouse, Steven Hansen, Angelos Filos, Ethan Brooks, Maxime Gazeau, Himanshu Sahni, Satinder Singh, and Volodymyr Mnih. In-context reinforcement learning with algorithm distillation. In *International Conference on Learning Representations*, 2023.

Shehryar Malik, Usman Anwar, Alireza Aghasi, and Ali Ahmed. Inverse constrained reinforcement learning. In *International Conference on Machine Learning*, 2021.

Vladimir Mikulik, Grégoire Delétang, Tom McGrath, Tim Genewein, Miljan Martic, Shane Legg, and Pedro A. Ortega. Meta-trained agents implement Bayes-optimal agents. In *Advances in Neural Information Processing Systems*, 2020.

Tom M. Mitchell. Version spaces: A candidate elimination approach to rule learning. In *International Joint Conference on Artificial Intelligence*, 1977.

Tom M. Mitchell. Generalization as search. *Artificial Intelligence*, 18:203–226, 1982.

Ian Osband. Delightful exploration. arXiv preprint arXiv:2605.13287, 2026.

Enze Pan. Tape: A cellular automata benchmark for evaluating rule-shift generalization in reinforcement learning. arXiv preprint arXiv:2601.04695, 2026.

Shishir G. Patil, Tianjun Zhang, Xin Wang, and Joseph E. Gonzalez. Gorilla: Large language model connected with massive APIs. In *Advances in Neural Information Processing Systems*, 2024.

Eric Pulick, Shubham Bharti, Yiding Chen, Vladimir Menkov, Yonatan Mintz, Paul Kantor, and Vicki M. Bier. The game of hidden rules: A new kind of benchmark challenge for machine learning. arXiv preprint arXiv:2207.10218, 2022.

Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, Sihan Zhao, Lauren Hong, Runchu Tian, Ruobing Xie, Jie Zhou, Mark Gerstein, Dahai Li, Zhiyuan Liu, and Maosong Sun. ToolLLM: Facilitating large language models to master 16000+ real-world APIs. In *International Conference on Learning Representations*, 2024.

Stéphane Ross, Brahim Chaib-draa, and Joelle Pineau. Bayes-adaptive POMDPs. In *Advances in Neural Information Processing Systems*, 2007.

Dexter R. R. Scobee and S. Shankar Sastry. Maximum likelihood constraint inference for inverse reinforcement learning. In *International Conference on Learning Representations*, 2020.

SeungWon Seo, DongHeun Han, SeongRae Noh, and HyeongYeop Kang. Baba in wonderland: Online self-supervised dynamics discovery for executable world models. arXiv preprint arXiv:2605.16725, 2026.

David Silver and Joel Veness. Monte-carlo planning in large POMDPs. In *Advances in Neural Information Processing Systems*, 2010.

Viacheslav Sinii, Alexander Nikulin, Vladislav Kurenkov, Ilya Zisman, and Sergey Kolesnikov. In-context reinforcement learning for variable action spaces. In *International Conference on Machine Learning*, 2024.

Miguel Suau, Matthijs T. J. Spaan, and Frans A. Oliehoek. Bad habits: Policy confounding and out-of-trajectory generalization in RL. In *Reinforcement Learning Conference*, 2024.

Éric D. Taillard. Benchmarks for basic scheduling problems. *European Journal of Operational Research*, 64:278–285, 1993.

Xingyao Wang, Simon Rosenberg, Juan Michelini, Calvin Smith, Hoang Viet Tran, Engel Nyst, Raman Malhotra, Xuhui Zhou, Valerie Chen, Robert W. Brennan, and Graham Neubig. The OpenHands software agent SDK: A composable and extensible foundation for production agents. arXiv preprint arXiv:2511.03690, 2025.

Martin L. Weitzman. Optimal search for the best alternative. *Econometrica*, 47:641–654, 1979.

Zheng Xiong, Luisa Zintgraf, Jacob Beck, Risto Vuorio, and Shimon Whiteson. On the practical consistency of meta-reinforcement learning algorithms. In *Workshop on Meta-Learning at the Conference on Neural Information Processing Systems*, 2021.

Shunyu Yao, Noah Shinn, Pedram Razavi, and Karthik Narasimhan. τ-bench: A benchmark for tool-agent-user interaction in real-world domains. In *International Conference on Learning Representations*, 2025.

Renos Zabounidis, Noah Siegelmann, Mohamad Qadri, Joseph Kim, Simon Stepputtis, and Katia Sycara. Overcoming valid action suppression in unmasked policy gradient algorithms. arXiv preprint arXiv:2603.09090, 2026.

Tom Zahavy, Matan Haroush, Nadav Merlis, Daniel J. Mankowitz, and Shie Mannor. Learn what not to learn: Action elimination with deep reinforcement learning. In *Advances in Neural Information Processing Systems*, 2018.

Junli Zha, Jinbo Wang, Chao Zhou, and Xiang Song. Trace2Policy: From expert behavior traces to self-evolving decision agents. arXiv preprint arXiv:2606.10457, 2026.

Yifan Zhu, Sammie Katt, and Samuel Kaski. Propose to learn, learn to propose: Evaluability-aware assistance under bounded rationality. arXiv preprint arXiv:2609.02242, 2026.

Luisa Zintgraf, Kyriacos Shiarlis, Maximilian Igl, Sebastian Schulze, Yarin Gal, Katja Hofmann, and Shimon Whiteson. VariBAD: A very good method for Bayes-adaptive deep RL via meta-learning. In *International Conference on Learning Representations*, 2020.

A THE DECISION STRATEGIES IN FULL

Every strategy scores the actions still available and plays the highest score; the one-step strategies break ties by adding 10^{-9} times the action's value. With $p(a)$ the posterior probability that a is legal and $v(a)$ its value, the scores are:

strategy	score of action a
belief-blind ablation	$v(a)$
exploit	$p(a)\,v(a)$
cost-only	$-c(a)$, with $c(a) = 1 + (1 - p(a))\max\{v(a), 0\}$
information-only	$\sigma(a) = \min\{p(a), 1 - p(a)\}$
one-step priced	$\sigma(a)/c(a)$
lookahead (the method)	$L_d(a)$ of Equation 2, default or exact updates
random play	uniform random over the available actions

Every strategy reads the same $p(a)$ of Equation 1. The belief it comes from is exact for the hypotheses held, and each hypothesis carries its own posterior over every machine's outage bit. An outcome at

a multiplies a hypothesis's weight by $q\,\lambda_1 + (1-q)\,\lambda_0$. Here q is its current posterior on the bit of a's machine, and λ_b is 0.95 if the hypothesis with that bit set to b predicts the outcome and 0.05 otherwise. The bit's posterior is updated by the same terms. The split σ is the criterion QuAcq2 uses to generate a query, restricted to the actions that can be played. The cost c is paid whatever comes back: the step is spent either way, and a refused action forgoes its value.

The lookahead computes the posterior after a hypothetical outcome in closed form on the hypotheses' log-weights. A hypothesis whose verdict on a does not depend on a hidden bit and disagrees with y is down-weighted by the noise level. The two forms differ only for a hypothesis whose verdict on a depends on a hidden bit. The exact lookahead applies the update above at every node: it multiplies the hypothesis's weight by $q\,\lambda_1 + (1-q)\,\lambda_0$ and updates q by the same terms, so the search carries each hypothesis's hidden-bit posterior. The default lookahead predicts that action with probability one half and leaves the hypothesis unchanged. The two choose different actions in 47% of decisions and score the same within noise (Table 3). In the default form $P(y \mid a)$ is $p(a)$ for an accept and $1 - p(a)$ for a refusal; in the exact form it is the belief's own predictive probability of y, which includes the noise level. A branch with probability below 10^{-6} is skipped.

The recursion makes three approximations. It expands only the six of the 36 actions with the highest one-step score at each node, and the rest keep their one-step score. Expanding all 36 changes the default lookahead's score by 0.0 pp on the safety-rail rules and the exact lookahead's score by -0.4 pp over the six shapes, a small loss whose Student interval, $[-0.6, -0.3]$ pp, excludes zero. It updates the belief on log-weights instead of cloning the set of hypotheses, which is what makes depth above one affordable. And it holds each hypothesis's responses at the current state inside the recursion: it carries exactly the removal of an accepted action from the available set, and it does not model the small change in other responses as the state advances. The method is therefore a shallow planner that is affordable at every step, not an optimal one.

B Configuration and statistical units

Interface. The trained predictors see $n = 3H$ actions per episode, so choosing the best H of $3H$ is a constant difficulty across budgets $H \in \{10, 20, 40\}$. The policy experiments see 36 actions at every budget, three on each of 12 machines. Every trained predictor uses one shared relational transformer with width 96, two layers and four heads, trained for six epochs by masked cross-entropy on $80{,}000$ transitions collected by one uniform prober, so the interaction budget is identical by construction. Evaluation is 720 episodes per (rule, seed) pair, and 240 in the count and planting experiments. None of the policy variants trains a network: they are inference policies over the module of Section 3 and run on CPU.

The inference module. The policy carries 24 hypotheses. Each is drawn by picking one of the grammar's templates (ten on the safety-rail rules), a conjunction of two or three primitives that always includes the hidden outage primitive, and grounding it at a width drawn uniformly from $[0.15, 0.85]$. The likelihood allows an unexplained outcome with probability 0.05. As a check that no candidate list leaks in, the policy's score is bit-identical whether or not the list handed to the QuAcq2 port contains the deployed rule. Every five steps the hypotheses below the median likelihood are replaced. About two thirds of the replacements perturb the width of a hypothesis drawn in proportion to its weight, and the rest are fresh draws. A replacement inherits its parent's likelihood, and a fresh draw enters at the current minimum, so resampling cannot win by recency. The lookahead expands six candidate actions per node, each with two outcomes. The richer grammar of Section 4.4 draws one to four conjunctions of one or two primitives over the same five primitive kinds. Its support contains every deployed rule, each deployed template drawn with probability between 1.6×10^{-6} and 0.025; resampling and the likelihood are unchanged.

Units and intervals. Rules of one shape are not independent, so the shape is the unit. Pooling the permission and rail sources gives six units and five degrees of freedom; one source alone gives three units and two. For each difference we report the mean over all paired runs, a bootstrap over shapes, and a Student interval over the shape means. A bootstrap over few units can only return values between the smallest and largest shape mean, so it is the narrower of the two, and we require both to exclude zero before calling a difference significant. Where noted we also give an interval over the nine deploy rules of a source. Pairing is within (rule, seed, decision strategy), so every difference

compares two variants on identical episodes. We fixed the six-shape pooling and the planting reading of Section 4.4 before running the planting experiments.

C THE RULE SOURCES

Permission predicates. The legality rules on this source are read from GitHub's MCP server (GitHub, 2025), an MIT-licensed tool server for language-model agents, at a fixed release. The predicate shapes and the state coordinates each predicate consults are theirs, and the source is read, not executed. Three shapes are encoded. Shape **T** is a single lookup, the check of whether a toolset is enabled. Shape **A** is a conjunction that admits a call only if every stage of the tool-enablement check passes. Shape **O** is a disjunction that admits content if any condition of the repository-access lockdown check passes. The distribution of the hidden vector, the action space, the budget, the network and the optimiser are ours. The encoding caps predicate arity at two, against the source's four, because every hidden primitive is keyed to the action's own coordinate and all experiments share that action representation. It also makes the second conjunct observable, because with both coordinates hidden the encoding would hand the predictor a value it is meant to infer. Rules are matched on their exact integer coordinate mass against a fixed reference world, so the two rules of a pair carry an identical closed-form illegality rate. The policy experiments use nine deploy rules, three per shape.

Safety rails. The second agent codebase is the MIT-licensed OpenHands software agent SDK (Wang et al., 2025), vendored byte-identical at a fixed release under checksums. Three rail shapes are read from its security policy. The fetch-then-execute rail is a conjunction of two coordinate sets (**RC**), the raw-disk-operation rail is a disjunction of two (**RD**), and the catastrophic-delete rail with the outer rail disjunction is an OR of two conjunctive rails (**RM**). Each shape supplies twelve rate-matched rules. The released shifts are twelve ordered pairs of consecutive rules within a shape, four per shape, and the policy experiments deploy nine held-out rules, three per shape. The shapes compile to a wider and deeper disjunctive normal form than the permission predicates reach, inside the same function class, so this source adds a second rule source and a second compiled structure, not more expressiveness.

The rail decisions are upstream's own. We execute the upstream rail policy in the loop: every judgement runs its own risk classifier, segment extraction, normalisation and rail evaluation, behind a thin wrapper that supplies four symbols the decision path never reads. A rail verdict depends only on the action's machine, its class and that machine's outage bit, which makes an exhaustive check possible. Over all 36 rules and the full per-rule input space, plus every state the twelve ordered pairs reach at $H = 40$, the executed policy and our encoding disagree on 0 of $464{,}256$ checks. The check has power: a renderer that suppresses one token disagrees on $42{,}320$ episode checks. Agreement also survives rendering each conjunct as realistic shell text, with slashes, device paths and flags under upstream's own pattern matching, at 0 disagreements. Every rail number here is therefore one that upstream's policy would produce; the world, the action language and the stepper remain ours.

Authored rules. On the GOHR corpus (Pulick et al., 2022), consecutive entries of the authors' own trial lists give fourteen train-to-deploy transitions over their authored rule files, read at a fixed release, and the deployed rule, the decoys and the boards are all upstream's. This source carries the gap of Table 2 and no experiment on the policy, because its rule files are not rate-matched to one another.

The form of the deployed rules. Every rule refuses an action when any of its conjunctive terms holds. A rail is checked on each of an action's two fields, the machine field and the class field, so a rail shape has twice as many terms as rails. The policy's grammar draws one term of two or three primitives, one of them the hidden outage bit. The deployed rules have two terms of two primitives (RC), four single-primitive terms (RD), four terms of two primitives (RM), one single-primitive term (T), two single-primitive terms (A), and one term of two primitives (O). Only shape O is a form the grammar can draw, and then only if a hypothesis names exactly its guarded sets. The richer grammar of Section 4.4 can draw every one of these forms. A planted rule is the compiled rule itself.

Test beds that carry no reported result. We also built the generator and the grading protocol on test beds that carry no reported result, and we name them here. Our own seeded rule generator carries

only the probing instrument of Appendix H. Three more carry no table, figure or claim in this paper. They are job-shop scheduling geometry from the Taillard benchmark instances (Taillard, 1993) and the Kubernetes scheduler's taint-toleration admission check, compiled and executed from two release tags. The third is the permission source with the server's verbose refusal, whose message names the failed condition.

D THE GAP UNDER EVERY WAY OF GRADING

Table 2 gives the gap at $H = 40$. It is paid in behaviour and not only in a score: on the permission rules the specialist earns $+25.94$ more episode return and attempts 10.60 fewer illegal actions per episode, every interval excluding zero. That axis is necessarily graded on each predictor's own trajectory, because on a shared trajectory the environment picks the action and both predictors record identical decisions.

The generalist has been strengthened seven ways, against inexperience, the training protocol, limited capacity, the lack of adaptation within an episode or across episodes, a narrow training distribution over rules, and by in-context learners on a second rule source. A specialist lead whose interval excludes zero survives each one on every shared trajectory: $+19.0$ pp on a value-greedy trajectory with a tuned network, $+15.0$ pp on a uniform-random one and $+19.3$ pp on one that deliberately probes. A memory carried across episodes and meta-trained over eleven rules is worth -0.1 pp over a plain network, against a specialist lead of $+18.3$ pp on the same rules. A network $3.9\times$ larger moves the gap to $+19.8$ pp against $+18.2$ pp, inside its own interval.

E GRADING A POLICY THAT CHOOSES ITS OWN ACTIONS

Two conventional ways of grading. Scoring each predictor on the states its own actions reach is the deployment distribution, but it carries a selection term: a policy that plays differently is graded on a different remainder of actions. Scoring every predictor on one shared trajectory, played by a policy that consults none of them, removes that term but takes the policy out of acting. The two disagree by more than a nuisance term. On the safety-rail rules, with a network selected on training loss alone, a predictor trained across the rule family recovers 72% of the gap, bootstrap [66%, 77%], on each predictor's own trajectory, against 10%, bootstrap [3%, 24%], on a value-greedy shared trajectory. The two are disjoint under both intervals and same-signed on each shape. On the permission rules the same contrast does not separate at nine deploy rules over three shapes. There the share of the gap that family training recovers varies with the predicate shape by a factor of about three, so at two degrees of freedom adding rules per shape cannot narrow the Student interval over shapes enough. This contrast between gradings therefore rests on the safety-rail rules; the policy comparisons of Section 4 pool both sources. A shared trajectory that generates more refusals does not credit an adaptive predictor more either. The value-greedy trajectory is refused on 39.05 of its 40 attempts and the uniform-random one on 22.12, and the second credits family training with more of the gap. The states a policy is graded on are part of what is being measured.

The common scoring trajectory. A policy whose claim is that it chooses informative actions needs both properties at once. Each variant therefore acts in a *probe* copy of the episode, advanced by its own decision strategy. A *scoring* copy with the same hidden vector is advanced by a uniform-random policy that never reads any variant. At every step the variant's current posterior predicts legality over the scoring copy's available actions, and those predictions are what is scored. The random-play variant consumes one random stream in both copies, so its score on this trajectory must equal its score on its own trajectory. It does, at a maximum absolute difference of exactly zero, on all 30 permission result files and all 15 rail result files, and the scorer stops on any non-zero value.

F SUCCESS CRITERIA AND EVERY LEVEL

We fixed five success criteria for the lookahead, run at depth 2, before reading any result. It succeeds if it (1) closes a significant share of the gap and (2) beats one-step pricing. (3) The share must reach 25% with episode return, the summed value of accepted actions, no more than 15% below the belief-blind ablation's. (4) It must match the share of an oracle-informed reference, the constraint-acquisition

algorithm QuAcq2 (Bessiere et al., 2023), ported so that its queries, free in the original, become played actions, and handed a candidate list containing the deployed rule. (5) The posterior must recover at least twice the information about the rule, in bits, that passive probing at random recovers, measured on a seeded rule generator of our own (Appendix H). The lookahead meets none of the five (Table 4), for the reasons given below.

	rail	permission	six shapes, both sources	
paired difference, $H = 40$	rules	rules	mean [bootstrap]	Student t, 5 df
lookahead − one-step priced	0.0	+0.7	+0.3 [−0.5, +1.0]	[−0.8, +1.4]
lookahead − exploit (no information term)	+1.0	+1.5	+1.3 [+0.3, +2.3]	[−0.2, +2.7]
lookahead − random play	+3.4	+3.6	**+3.5 [+2.7, +4.3]**	**[+2.3, +4.7]**
lookahead − belief-blind ablation	+13.5	+14.7	**+14.1 [+12.1, +16.4]**	**[+11.0, +17.3]**
lookahead − trained generalist	−1.4	+4.9	+1.7 [−6.6, +7.6]	[−8.9, +12.4]
with exact belief updates inside the search				
exact lookahead − one-step priced	+0.8	−0.4	+0.2 [−0.6, +1.0]	[−1.0, +1.3]
exact lookahead, all 36 actions − one-step	+0.3	−0.8	−0.2 [−0.9, +0.5]	[−1.3, +0.8]
exact − default lookahead	+0.8	−1.1	−0.2 [−1.1, +0.7]	[−1.4, +1.1]
exact lookahead − random play	+4.1	+2.4	**+3.2 [+1.7, +4.4]**	**[+1.2, +5.2]**

Table 3: **Every decision-stage difference behind Figure 2.** Differences in pp at $H = 40$ on nine held-out deploy rules per source, three per shape, five seeds, every variant scored on the common scoring trajectory. All rows are paired except the generalist row, a level comparison against separate runs with $3H = 120$ actions. The exact-lookahead rows come from separate runs with fewer episodes per pair and are paired only with each other. The source columns give each source's mean; the six-shape columns give the pooled mean with a 95% bootstrap over shapes and a 95% Student t interval over the shape means. **Bold** marks a significant difference.

Table 4 lists the criteria with the lookahead's measured values. Table 3 gives every decision-stage difference and Table 6 each source's intervals for them, and Figure 4 with Table 5 gives every level. Shares are fractions of the specialist gap at $H = 40$, computed for each (rule, seed) pair and averaged. As a ratio of the average differences, the lookahead's share is -9% on the safety-rail rules and 27% on the permission rules, because the gap varies about fivefold across pairs. Every strategy that weighs value or information by the posterior lands at about the same level. The one significant difference among them, the lookahead's $+0.4$ pp over the information-only strategy on the permission rules, is small, and only the cost-only strategy, which favours actions it is sure are legal, falls below the belief-blind ablation.

criterion	requires	permission rules	rail rules	met
1 closes a share	share > 0, both intervals	22% (bootstrap only)	-22%	no
2 beats one-step	difference > 0, both intervals	+0.7	0.0 (depth 2) ; -0.3 (depth 3)	no
with exact updates	as above	+0.2 over six shapes		no
3 share and return	share $\geq 25\%$, return within 15%	22%	-22%	no
4 matches incumbent	share $\geq$ incumbent's	22% vs 133%	-22% vs 132%	no
5 identifies the rule	bits $\geq 2\times$ passive (8.296)	3.454 bits, on the generator of App. H		no
with exact updates	as above	2.792 bits, same instrument		no

Table 4: **All five success criteria fail.** Criterion 2 compares the lookahead with the one-step priced strategy. The incumbent of criterion 4 is a QuAcq2 port handed a candidate list that contains the deployed rule. Differences are in pp; criterion 5 is measured in bits on the probing instrument of Appendix H.

Criterion 1, a share of the gap. Its first clause asks the method's share to exclude zero under both intervals. On the permission rules the method gains $+4.9$ pp over the generalist, bootstrap $[+1.5, +9.9]$ pp but Student interval $[-6.0, +15.8]$ pp, so the clause fails. On the safety-rail rules the method sits at -1.4 pp, with per-shape values of $+4.6$, -17.9 and $+9.0$ pp. A second clause, that the belief-blind ablation's share falls below the method's, holds in the method's favour on both sources. Pooled over the six shapes the method gains $+1.7$ pp, and its mean share is 0% with a Student interval of $[-74\%, +74\%]$, because the per-pair denominators vary about fivefold.

variant, balanced accuracy at $H = 40$	rail rules	permission rules
handed a list containing the deployed rule	93.0	92.3
retrained specialist (the target)	90.2	88.7
handed a rate-matched wrong list	78.4	74.9
trained generalist	73.7	70.7
lookahead (the method)	72.3	75.6
one-step priced strategy	72.3	74.9
information-only strategy	72.2	75.2
exploit strategy, no information term	71.3	74.0
random play chooses the action	68.9	72.0
belief-blind ablation (ignores the posterior)	58.8	60.8
cost-only strategy	54.4	54.2

Table 5: **Levels behind Figure 4.** Balanced accuracy in percent at $H = 40$, nine held-out deploy rules per source over three shapes, five seeds. The lower block shares one inference module and the common scoring trajectory; the upper block's generalist and specialist come from separate runs of the trained predictors (Section 4.1).

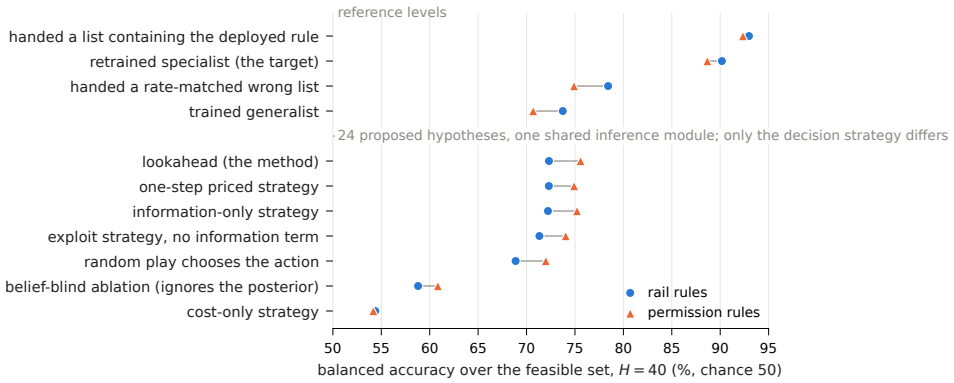

Figure 4: **Every decision strategy that weighs value or information by the posterior lands at about the same level: near the trained generalist, and far below both the retrained specialist and a policy handed a list that contains the deployed rule.** Balanced accuracy in percent on the common scoring trajectory at $H = 40$, nine held-out deploy rules per source, five seeds. The seven lower rows share one inference module over 24 proposed hypotheses and differ only in how the played action is chosen. The reference levels come from separate runs (Section 4.1).

Criterion 2, beating the one-step strategy. It fails on the safety-rail rules at 0.0 pp, with per-shape values of $+0.4$, $+1.0$ and -1.4 pp, again at depth 3 and with every action expanded, and over the six shapes at $+0.3$ pp. With exact belief updates inside the search it fails again, at $+0.2$ pp over the six shapes (Section 4.2).

Criterion 3, share and return. The share on the permission rules is 22% against the 25% required. Its return leg passes at $H = 40$ only, with episode return 18.29 against the belief-blind ablation's 9.74 on the safety-rail rules.

Criterion 4, the incumbent. The incumbent is a QuAcq2 port whose free query is replaced by a played action. Handed a candidate list containing the deployed rule, it closes, averaged over (rule, seed) pairs, 133% of the gap on the permission rules and 132% on the safety-rail rules, passing the specialist outright. The method closes 22% and -22%, so it fails the criterion. Because the port is handed its list, this comparison prices access to the true rule, not a better algorithm.

Criterion 5, identification. The posterior must recover at least twice the 4.148 bits that uniform passive probing recovers on the instrument of Appendix H, that is, 8.296 bits. The lookahead's own

paired difference, $H = 40$	mean [bootstrap, shapes]	Student t, shape means
rail rules		
lookahead − one-step priced, depth 2	0.0 [−1.4, +1.0]	[−3.1, +3.1]
lookahead − one-step priced, depth 3	−0.3 [−0.8, +0.1]	[−1.5, +0.9]
lookahead − information-only	+0.1 [−1.1, +1.4]	[−3.0, +3.2]
lookahead − exploit, no information term	+1.0 [+0.4, +1.7]	[−0.7, +2.7]
lookahead − random play	**+3.4 [+2.6, +3.9]**	**[+1.6, +5.3]**
lookahead − belief-blind ablation	**+13.5 [+10.5, +16.5]**	**[+6.1, +21.0]**
lookahead − trained generalist	−1.4 [−17.9, +9.0]	[−37.3, +34.4]
true-rule list − matched wrong list	**+14.5 [+13.6, +15.4]**	**[+12.4, +16.7]**
permission rules		
lookahead − one-step priced, depth 2	+0.7 [−0.4, +1.5]	[−1.7, +3.0]
lookahead − information-only	**+0.4 [+0.3, +0.4]**	**[+0.1, +0.6]**
lookahead − exploit, no information term	+1.5 [−0.6, +3.3]	[−3.4, +6.4]
lookahead − random play	+3.6 [+1.8, +4.7]	[−0.4, +7.5]
lookahead − belief-blind ablation	**+14.7 [+12.6, +18.8]**	**[+6.0, +23.4]**
lookahead − trained generalist	+4.9 [+1.5, +9.9]	[−6.0, +15.8]
true-rule list − matched wrong list	+17.4 [+3.3, +27.8]	[−14.1, +49.0]

Table 6: **Each source's decision-stage differences, with intervals over its three shapes.** Paired differences in pp at $H = 40$, nine held-out deploy rules per source, five seeds, common scoring trajectory. **Bold** marks a significant difference. The last row of each panel is the QuAcq2 port with and without the deployed rule in its handed list, which the method never gets.

probes, at depth 2 over the six best actions, recover 3.454, so the criterion fails for the method. With exact belief updates they recover 2.792. The one-step priced prober recovers 3.953 and fails too.

G THE COMPUTE BILL

seconds per (rule, seed) pair, $H = 40$	depth 2	depth 3
belief-state lookahead, mean ± sd	293.2 ± 9.3	1766.8 ± 11.0
one-step priced strategy	71.5	41.9
random play (control work)	71.0	42.2
lookahead over one-step	4.1×	42.2×
random play over one-step (node check)	1.0×	1.0×

Table 7: **The lookahead's price is time.** Nine rail (rule, seed) pairs at $H = 40$, each variant timed in the same process on the same node as its comparator. The columns are the depth-2 and depth-3 timing runs, and the one-step and random-play rows are the comparators timed alongside each. The node check divides random play's time by the one-step strategy's on identical control work.

Table 7 gives the timings. The raw wall-clock ratio of depth 3 to depth 2 is $6.0\times$, but the two runs used nodes of different speed: the one-step strategy's identical work ran $1.7\times$ faster on the depth-3 node. Scaled to one node speed, the extra step costs $10.3\times$, about the raw ratio times the node-speed factor. Peak memory, read from job accounting, is lower for the depth-3 timing jobs than for the full depth-2 experiment, because the number of variants differs and each belief node is only a weight vector over 24 hypotheses. The gain from more hypotheses (Section 4.4) is bought at a cost that grows about linearly in their number.

H THE INFORMATION ONE REFUSAL BIT CARRIES

Passive probing. We drew 4096 hypotheses from the policy's grammar at its sampling density on our own rule generator. Uniform passive probing at the 30 queries an episode of that generator affords recovers 4.148 of a 9.042-bit ceiling among the hypotheses rate-matched to the deployed rule,

leaving 354 of 705 in one indistinguishable block. A positive control on the same instrument recovers 5.305 of 6.044 bits where separation is known to exist, and 28% of draws are vacuous on the probe.

The lookahead's probes. Run as a prober on the same 4096 hypotheses, worlds and seed, the lookahead at depth 2 over the six best actions recovers 3.454 bits, with a standard deviation over trajectories of 2.412 bits. Expanding every available action gives 3.647 bits and depth 3 gives 3.543. The exact lookahead recovers 2.792 bits, with a standard deviation of 2.190. Expanding every action gives 3.073 and depth 3 gives 3.247. All of these fall below passive probing and far from the 8.296 of criterion 5. The same instrument with a uniform prober reproduces the passive 4.148 exactly.

One-step probing. The one-step priced prober recovers 3.953 bits against passive probing's 4.148, with a standard deviation over trajectories of 1.960 bits, so it buys no identification that this spread could detect. At the first step, with the posterior uniform over 4096 hypotheses by construction, the most balanced available action splits them 92% to 8%, which is 0.391 bits, and about 5.89 actions tie there. No available probe comes close to an even split, which is consistent with an information term that buys little. These measurements come from one hypothesis distribution on one rule algebra.

I THE PROPOSAL EXPERIMENTS IN FULL

	rail rules	permission rules	six shapes	
paired difference, $H = 40$	mean [bootstrap]	mean [bootstrap]	mean [bootstrap]	Student t, 5 df
exact deployed rule − wrong rule	+10.7 [+6.4, +18.0]	+10.9 [+3.7, +20.2]	**+10.8 [+6.3, +15.7]**	**[+3.8, +17.8]**
one element swapped − wrong rule	**+4.6 [+3.4, +5.7]**	+8.4 [+2.0, +16.1]	**+6.5 [+3.5, +10.5]**	**[+1.2, +11.7]**
two elements swapped − wrong rule	+0.1 [−6.7, +4.3]	+6.7 [+1.1, +12.8]	+3.4 [−1.4, +8.0]	[−3.3, +10.1]
four elements swapped − wrong rule	−1.9 [−10.7, +3.4]	+5.3 [+0.7, +10.5]	+1.7 [−3.9, +6.3]	[−5.6, +9.0]
deployed + four wrong rules planted − wrong rule	+11.7 [+7.4, +19.7]	+11.5 [+5.2, +20.0]	**+11.6 [+7.2, +16.2]**	**[+4.8, +18.5]**
wrong rule − nothing planted	+5.9 [+1.9, +13.4]	+2.1 [+0.7, +3.7]	+4.0 [+1.6, +8.0]	[−0.9, +8.9]
exact rule in the pool − handed list alone	**−4.5 [−5.1, −3.8]**	−3.6 [−4.4, −3.2]	−4.1 [−4.6, −3.5]	[−4.9, −3.2]
exact rule protected − unprotected	+7.7 [+4.7, +13.7]	+7.2 [+4.3, +12.0]	**+7.5 [+4.7, +10.6]**	**[+3.0, +11.9]**
exact deployed rule − nothing planted	+16.6 [+8.9, +31.4]	+13.1 [+7.3, +22.2]	**+14.9 [+8.7, +22.6]**	**[+4.6, +25.1]**
handed list alone − nothing planted	+21.1 [+14.0, +35.2]	+16.7 [+11.7, +25.5]	**+18.9 [+12.9, +26.5]**	**[+9.0, +28.8]**

Table 8: **Every planting difference, in pp.** Nine held-out deploy rules per source over three shapes, five seeds, $H = 40$, averaged over the exploit, one-step priced and lookahead strategies, common scoring trajectory. **Bold** marks a difference whose bootstrap and Student intervals both exclude zero; each source's Student interval, over three shapes, is not shown.

change to the proposal step	against	mean [bootstrap, shapes]	Student t, shape means
count: 24 → 128, rail rules	24 hypotheses	**+3.5 [+3.2, +4.1]**	**[+2.3, +4.8]**
count: 24 → 128, permission rules	24 hypotheses	**+4.4 [+3.0, +5.2]**	**[+1.5, +7.3]**
count: 128 → 512, rail rules	128 hypotheses	+0.3 [−0.4, +0.8]	[−1.3, +1.9]
target: templates, by the posterior	same seeds, off	0.0 [0.0, 0.0]	[−0.1, 0.0]
target: widths	same seeds, off	0.0 [−0.1, +0.1]	[−0.2, +0.2]
target: widths, twice as strongly	same seeds, off	0.0 [−0.1, +0.1]	[−0.3, +0.2]
target: reject refuted draws	same seeds, off	0.0 [−0.2, +0.2]	[−0.5, +0.5]
grammar: add 4- and 5-primitive conjunctions	same seeds, off	−0.3 [−0.6, +0.3]	[−1.5, +0.9]

Table 9: **Drawing more hypotheses buys one significant step on each source; targeting the draw buys no significant gain.** Differences in pp. Nine held-out deploy rules over three shapes per source, five seeds, $H = 40$. The rail count rows pool the lookahead and random play; the permission row pools the exploit, one-step priced and lookahead strategies. Every targeting and grammar row runs on the rail rules on the same random numbers as its baseline, which has the change switched off. Intervals as in Table 3, over three shapes, with **bold** marking a significant difference.

Planting. Each planted condition overwrites one of the 24 hypotheses of the identical first draw, or five of them for the handed list, and resampling never replaces or perturbs a planted hypothesis unless the condition says so. Table 8 gives every difference. Proposal is the limit under each strategy alone, with the copy one element off keeping between 58% and 61% of the exact gain. The swapped

copies keep the number of guarded machines, not the exact refusal rate. On the safety-rail rules the planted rule with one, two and four elements swapped disagrees with the deployed rule on 8%, 15% and 20% of the scored actions, against 21% for the wrong rule; on the permission rules on 5%, 9% and 11%, against 26%. Under the lookahead a planted rule one element off holds 53% and 58% of the posterior mass, and a planted exact rule is the most probable hypothesis 89% and 88% of the time. Without a planted rule the most probable hypothesis disagrees with the deployed rule on 29% and 23% of the scored actions. Left free to be replaced, the exact rule holds 44% and 43% of the mass, and the policy keeps about half its gain over nothing planted. Three checks pass at exactly zero on all 30 result files: with nothing planted the policy reproduces the default one, the planted exact rule disagrees with the deployed rule on no scored action, and every condition of a (rule, seed) pair shares one scoring trajectory.

	rail rules	permission rules	six shapes	
paired difference, $H = 40$	mean [bootstrap]	mean [bootstrap]	mean [bootstrap]	Student t, 5 df
exact deployed rule − wrong rule	+9.1 [+4.4, +18.0]	+8.4 [+2.1, +16.6]	**+8.7 [+4.1, +13.7]**	**[+1.6, +15.9]**
one element swapped − wrong rule	+4.0 [+1.5, +7.5]	+5.8 [+1.0, +12.2]	**+4.9 [+2.2, +8.2]**	**[+0.5, +9.4]**
two elements swapped − wrong rule	+1.0 [−0.9, +2.6]	+4.4 [+0.1, +9.0]	+2.7 [+0.4, +5.4]	[−1.0, +6.4]
four elements swapped − wrong rule	−0.4 [−3.3, +2.1]	+3.1 [−0.2, +6.9]	+1.3 [−1.1, +3.8]	[−2.3, +4.9]
deployed + four wrong rules planted − wrong rule	+10.1 [+5.2, +19.6]	+9.1 [+2.9, +17.5]	**+9.6 [+4.9, +14.8]**	**[+2.2, +17.0]**
wrong rule − nothing planted	+1.9 [0.0, +5.5]	+0.3 [−0.1, +0.7]	+1.1 [+0.1, +2.9]	[−1.1, +3.4]
exact rule in the pool − handed list alone	−4.0 [−4.7, −3.3]	−4.2 [−5.4, −3.3]	−4.1 [−4.7, −3.5]	[−5.0, −3.2]
exact rule protected − unprotected	+9.1 [+4.1, +19.0]	+7.2 [+2.2, +13.7]	**+8.1 [+3.7, +13.4]**	**[+1.1, +15.1]**
exact deployed rule − nothing planted	+11.0 [+4.4, +23.5]	+8.7 [+2.7, +17.0]	**+9.9 [+4.4, +16.5]**	**[+1.0, +18.7]**
handed list alone − nothing planted	+15.0 [+8.5, +27.3]	+12.8 [+6.5, +22.5]	**+13.9 [+8.2, +20.6]**	**[+4.8, +23.1]**
nothing planted: expressive − default grammar	**+5.8 [+4.4, +7.8]**	**+4.1 [+3.4, +5.4]**	**+4.9 [+3.9, +6.3]**	**[+3.2, +6.7]**

Table 10: **Every planting difference under the richer grammar, whose hypotheses can express every deployed rule.** As Table 8, averaged over the exploit, one-step priced and exact lookahead strategies. The last row compares the two grammars with nothing planted. **Bold** marks a significant difference.

The richer grammar. Table 10 repeats every planting difference with the grammar of Section 4.4 that can express every deployed rule. Its identity checks pass at exactly zero on all 30 result files, and the scorer, run on the default-grammar files, reproduces their differences to within 10^{-12}. With nothing planted, the closest of the richer grammar's own hypotheses disagrees with the deployed rule on 8% and 7% of the scored actions, against 18% and 11% under the default grammar. Resampling still discards a planted rule it is free to replace: protecting the exact rule adds $+8.1$ pp.

Rejecting refuted draws. The informed proposer rejection-samples every draw against the refusals so far, which is exact consistency with the evidence and nothing stronger. Its one-attempt version reproduces the default policy exactly over 90 paired runs. The filter is active: the informed proposer makes 8.35 grounding attempts per accepted hypothesis, so about 88% of draws are discarded as refuted. Of the hypotheses, 11% exhaust all 32 attempts, and 47% of unfiltered draws are already refuted when drawn; the few slots that exhaust their attempts dominate the mean. The decision does not move: -0.2, -0.1 and 0.0 pp at $H = 10, 20, 40$, each covering zero under both intervals, with a Student interval of [−1.2, +0.7] pp at $H = 10$. Rejection at draw time does what likelihood resampling already does, only earlier. Both can remove a hypothesis the data refute, and neither can produce one near the deployed rule.

Count. The step from 24 to 128 hypotheses is positive on all three shapes, at $+3.2$, $+3.3$ and $+4.1$ pp. The step from 128 to 512 has per-shape values of $+0.9$, -0.4 and $+0.5$ pp. In levels the default policy scores 70.7%, the 128-hypothesis policy 74.2% and the 512-hypothesis policy 74.5%, against the generalist's 73.7% from the decision-rule experiment. That level comparison crosses two experiments, while the differences in Table 9 are paired within one. Against the $+14.5$ pp that the true-rule list is worth on the same rules, the first step buys 24%. Split by strategy, the step is $+4.1$ pp under the lookahead and $+2.9$ pp under random play, each significant, and their difference of $+1.2$ pp is not; the step from 128 to 512 is significant under neither. On the permission rules the step from 24 to 128 is significant under each of the three strategies that act on the posterior, and under random play only its bootstrap excludes zero.

Targeting the templates. Reweighting the template draw by the policy's own posterior moved the template distribution to a mean total-variation distance of 0.323 from uniform and cut the number of distinct templates held from 5.88 to 5.27. The decision moved by less than a tenth of a point at $H = 10, 20, 40$; at $H = 40$ its bootstrap excludes zero while its Student interval does not. Three intervals across the targeting operators exclude zero, each at a magnitude below a tenth of a point, and we report them as effects of no substantive size.

Targeting the width. Reweighting the grounding width moved the distribution that widths are drawn from 0.109 in total variation away from uniform, and 0.214 at twice the strength. The mean drawn width moved only from 0.4658 to 0.4706 on a range of 0.15 to 0.85, because the policy's own posterior over widths is nearly flat. That flatness is itself a measurement: refusals do not identify the grounding width, and this operator shows that we could not target the width, not that targeting would not help.

Enlarging the grammar. Conjunction depths 4 and 5, which the default grammar cannot express, supply 25% of draws and move the decision by -0.5, -0.4 and -0.3 pp at $H = 10, 20, 40$, each covering zero under both intervals, with a Student interval as wide as $[-2.4, +1.3]$ pp at $H = 10$.

Why every baseline consumes the same random numbers. A baseline that differs from the default policy only in how its random numbers are consumed reads -0.6, -0.4 and -0.2 pp at the three budgets, covering zero, yet reaches 12.1 pp on a single (rule, seed) pair. The grammar experiment's own baseline reaches 10.4 pp on a single pair. Every operator effect is far smaller than this draw luck, so each operator is compared with its own baseline run on the same random numbers, never with the default policy.

How much of the posterior the operators reach. Over $302{,}400$ steps, hypotheses born in resampling hold 67% of the normalised posterior mass and fill 17.08 of the 24 places on average, and in 67% of steps the most probable hypothesis is one of them. An operator that changes what resampling draws therefore changes most of the belief.

Identity checks. Every result file of the count and targeting experiments passes the common scoring trajectory's identity check exactly. The default 24-hypothesis policy, which appears in three sets of runs, reproduces bit-identically across them. The width operator's baseline, run on the same random numbers with the operator off, reproduces the default policy exactly, and the template operator's baseline differs from it only in how random numbers are consumed.

J COMPARATORS

	80 000 transitions		800 000 transitions	
rail rules, 12 pairs, $H = 40$	level	carry − no carry	level	carry − no carry
RL^2-class, own return objective	52.7	-0.5 [-2.6, +1.6]	51.3	0.0 [-0.4, +0.3]
VariBAD-class, own objective	55.6	-0.2 [-0.7, +0.3]	52.4	+0.2 [-1.7, +2.1]
trained generalist, same pairs	60.1	–	61.9	–
retrained specialist, same pairs	80.0	–	80.8	–

Table 11: **Both in-context learners stay near chance, and a $10\times$ budget moves both down.** Rail rules, twelve ordered pairs over three shapes, five seeds, $H = 40$, scored on a value-greedy shared trajectory; levels in percent, differences in pp. The carry column differences each learner against a copy that sees one episode at a time; the interval resamples the shape.

In-context learners on their own objectives. Table 11 gives both learners' levels. The RL^2-class learner is a recurrent policy trained by REINFORCE with a detached baseline on episode return, with the carried context spanning several consecutive episodes of one task. The VariBAD-class learner fits an explicit variational task posterior and conditions its policy on it. Both see the same observations and budget as every other variant. A positive control detects the specialist's lead on these rules inside

the same experiment, at $+18.9$ pp, so a carry difference that covers zero reflects the learners and not an insensitive measurement. At ten times the $80{,}000$-transition training budget the RL^2-class learner clears chance by $+1.3$ pp, trivially small although its interval excludes zero, and the VariBAD-class learner by $+2.4$ pp with an interval covering zero. The VariBAD-class posterior moves with the refusals, more at the larger budget than at the published one, so the task posterior is live and the decision still does not move.

control	quantity	value
authors' benchmark (GridNavi)	training rise in return	+11.22 ($\pm$ 0.08 SE)
	episode 4 $-$ episode 1 of a task	+4.88 ($\pm$ 0.98 SE)
planted latent-class rule	class-aware oracle return	44.64
	best class-blind return	30.45
	their policy at 4×10^6 frames, rise	+1.57
	their policy at 4×10^6 frames, ep. 4 $-$ ep. 1	-0.44
	their policy at 2×10^7 frames, return	1.62 ($\pm$ 0.14 SE)
rail rules, three shapes	balanced accuracy (RC / RD / RM)	50.5 / 51.3 / 51.1

Table 12: **The authors' own VariBAD trains and adapts on its home benchmark, but not through our adapter, so it bounds nothing about this interface.** Three seeds per row, at the frame (environment-step) counts named in each row; balanced accuracy in percent.

The published implementation, end to end. Table 12 summarises this run. We vendored the authors' VariBAD unmodified and ran it at its own scale and optimiser: twelve runs of 2×10^7 environment steps, four per rail shape (three seeds at $H = 40$ and one at $H = 10$). The vendoring is sound on the authors' benchmark: training raises return by $+11.22$ and return rises by $+4.88$ from the first to the fourth episode of a task, with a no-adaptation control reading exactly zero before training. Through our adapter it fails a control built to be easy. On a planted test rule where the first legal action reveals the hidden class, an oracle that knows the class earns 44.64 and no policy that ignores refusals can exceed 30.45. Their policy returns 1.62 at the full budget, and its return never rises within a task. On the safety-rail rules it scores 50.5%, 51.3% and 51.1% on the three shapes. This is a failed port, not a measured comparator.

K Related work in full

Each line is placed by what it assumes. Our setting has five properties at once. Legality is a deterministic function of a *hidden* per-episode vector, and the only signal about it is a *refusal*. The probe is a *played* action whose refusal spends a step and advances the state. Deployment *replaces* the rule: with another authored rule on GOHR, and with a same-shape rule that guards different machines, classes or tools on the other two sources. And the object is the *whole* currently legal set.

Priced and planned information gathering. Cost-sensitive value of information (Krause & Guestrin, 2005), adaptive submodularity (Golovin & Krause, 2011), noisy generalized binary search (Golovin et al., 2010) and Pandora's rule (Weitzman, 1979) select from a fixed set of queries whose inspection does not move the world, which is what their greedy guarantees rest on. Bayes-adaptive POMDPs (Ross et al., 2007) drop that assumption, and Zhu et al. (2026) study shallow Bayes-adaptive planning over proposed hypotheses, with probes that both inform and earn reward. Osband (2026) prices exploration with a horizon term. We do not claim any of these as a contribution. We built the one-step priced strategy and a lookahead over it, measured the lookahead's increment with default and with exact belief updates, and found no significant increment for this shallow planner.

Constraint acquisition and version spaces. Candidate elimination (Mitchell, 1977; 1982), QuAcq and its successors (Bessiere et al., 2023; 2013; 2007) and exact learning with membership queries (Angluin, 1988) identify a target from queries that cost nothing and never change the network. They give exact convergence bounds that we do not. Their probe is a freely chosen assignment, and ours is a played action whose refusal advances the state; their target has no hidden part, and ours is gated by a hidden vector redrawn each episode. We port QuAcq2 as a variant and use it as the incumbent of criterion 4 (Appendix F).

Learning constraints from demonstrations. Maximum-likelihood constraint inference (Scobee & Sastry, 2020) and inverse constrained reinforcement learning (Malik et al., 2021) are closest by problem statement. Both need a demonstrator together with nominal reward and dynamics, and infer a fixed constraint offline, so neither runs online from refusals alone as a matched-information comparator.

Feasible sets from interaction. Action elimination (Zahavy et al., 2018) supervises an elimination network with exactly our per-action refusal bit, but in one environment under one fixed notion of validity, so there is no change in the rule's form to transfer across. Invalid-action masking (Huang & Ontañón, 2022), affordances (Khetarpal et al., 2020), stochastic action sets (Boutilier et al., 2018), sleeping-action formulations (Drago et al., 2025) and recent feasibility-aware value methods (Chen et al., 2026; Zabounidis et al., 2026) take availability as given or as a stable property to estimate. None measures what a policy loses when the rule generating it is replaced, and none recovers it.

Inducing rules from interaction. Seo et al. (2026) induce a program for the transition law from fully observed transitions and name proposal repair from rejected candidates as the remedy. Their evidence unit is a whole transition, ours one refusal bit, and rejecting refuted draws tests one such repair within one grammar. Pan (2026) isolate a hidden rule shift in the dynamics, not in legality.

In-context adaptation. RL^2 (Duan et al., 2016), VariBAD (Zintgraf et al., 2020) and Algorithm Distillation (Laskin et al., 2023) adapt from a carried history, and Headless-AD (Sinii et al., 2024) extends in-context learning to action sets unseen in training. None targets a hidden legality rule revealed only by refusal. Mikulik et al. (2020) show meta-trained agents implementing Bayes-optimal ones when the optimal policy is in the search space and training converges, and Xiong et al. (2021) report RL^2 and VariBAD failing to adapt out of distribution in their default form. We ran both classes on this interface and report where they land.

Grading distributions and null results. Huang & Jiang (2022) bound off-policy estimation under an arbitrary user-specified error-measuring distribution, and Suau et al. (2024) show that a policy-conditioned state distribution can confound what an agent learns. Both are reasons to name the grading distribution, which our common scoring trajectory does. Bellibatlu et al. (2026) require a measured instability floor before an audit's flip rate is read, which our nulls carry together with an in-design sensitivity check.

Tool-use agents. Benchmarks for tool-using agents (Yao et al., 2025; Qin et al., 2024; Patil et al., 2024) are where refusals under hidden entitlements are the everyday case. None replaces the legality rule between training and deployment, and none holds the refusal rate equal across such a change. They motivate the setting and are not evidence about it.