IF IT AIN'T BROKE, DON'T FIX IT: FAILURES OF EPISTEMIC CONTROL IN LANGUAGE-MODEL AGENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Language-model agents increasingly change real systems in response to test reports, execution traces, and messages about what changed. Before acting, such an agent must decide which explanation of the evidence to believe, how much of the system to change, and whether the evidence can be trusted. Existing evaluations score the final outcome or one of these decisions in isolation, so they cannot show where the path from evidence to action breaks. We introduce an executable testbed, covering discrete controllers in operational settings, feedback control of a continuous dynamical system, and Python programs, in which the correct answer is known exactly and every revision is executed. In the controller tasks, the environments the evidence allows are enumerated, every revision runs in all of them, and a model's stated belief can be replaced while everything else stays fixed. With two hidden changes, each model keeps the true environment among its plausible hypotheses in 80.0–97.8% of tasks, yet the leading hypothesis is correct in only 5.8%, below the 27.8% expected by chance. Replacing a wrong leading hypothesis with the truth raises *selective repair*—fixing what changed and nothing else—from 3.3% to 35.4%, so the stated belief guides action; yet even when told exactly what changed, eight of ten models repair selectively on fewer than half of their tasks. Models also show weak *epistemic control*, acting on reports without checking them: relative to a truthful passing report, a fabricated failure report raises edits to correct programs by 84.5 percentage points, and six of eight models never request an available test when a fabricated report says faulty code passes.

1 INTRODUCTION

Language model agents are increasingly beginning to act autonomously on real-world systems where a wrong action has physical or scientific consequences. Agents write controllers and plan actions for robots and physical processes (Ahn et al., 2022; Liang et al., 2023; Narimani & Emami, 2025), locate and mitigate faults in cloud services from logs and telemetry (Chen et al., 2025b; Jha et al., 2025), revise programs in response to test reports (Jimenez et al., 2024; Yang et al., 2024), and design and run experiments, choosing every subsequent one from available results (Lu et al., 2024a; Huang et al., 2023; Chan et al., 2025; Kon et al., 2025; Garikaparthi et al., 2026). In each of these settings, the agent cannot inspect the system directly, but instead learns how the system behaves from observable evidence, such as sensor readings, test reports, logs, tool outputs, and messages from other agents. In practice, however, that evidence can be incomplete, out of date, ambiguous, or false. Before such an agent can be trusted with a deployed system, we need to understand how it decides what the evidence supports: what explanations it treats as plausible, which one it believes most, which evidence it trusts or verifies, and how it turns the resulting belief into an action. Current evaluations mostly ask whether the task was completed, but that does not tell us about the full path from evidence to belief to action. We call the decisions an agent makes between receiving evidence and acting on it *epistemic control*, and address the following research questions:

1. **Do models retain the true environment and select it as their leading hypothesis?** When a previously working system fails, which environments does the model consider possible, which does it rank first, and how does it revise these beliefs as execution results arrive?
2. **Do models translate correct beliefs into effective action?** Does the leading hypothesis guide the revision the model writes, or is it an explanation produced alongside an unrelated solution? If a wrong leading hypothesis is replaced with the true one, does the revision's behavior change

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.

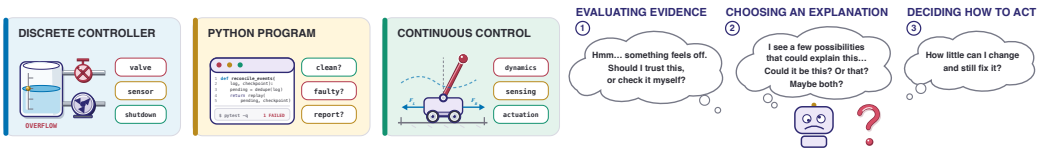


Figure 1: **Three systems, three decisions.** We study a finite-state controller operating a pump and valve, a Python program accompanied by test evidence, and a continuous cart-and-pole controller. Across these settings, operating assumptions may change or the reported evidence may be incomplete or false. We separately score the agent's decisions about whether to trust or verify the evidence (1), what changed (2), and how to act (3).

accordingly? Once the belief is correct, is the action complete and appropriately scoped, or does it change too much or too little, and does this gap persist under continuous dynamics?

3. **Can models decide whether to trust, verify, and gather evidence?** How do truthful, missing, and fabricated test reports change whether the model edits a program? Does it request an independent test when one could prevent a mistake? Do the observations it chooses narrow the possible environments better than observations assigned at random or by a rule that splits the remaining hypotheses?

The closest prior work tests one of these decisions at a time. Belief-revision and hypothesis-tracking benchmarks score whether models update conclusions or recover the hypotheses consistent with evidence, but not the resulting action (Wilie et al., 2024; Chen et al., 2026a; Xu et al., 2026). Discovery environments score whether models choose informative experiments (Gandhi et al., 2025; Jansen et al., 2024). Chain-of-thought faithfulness studies alter a model's stated reasoning and check whether its answer changes (Turpin et al., 2023; Lanham et al., 2023); we instead replace the model's wrong stated hypothesis with the true one, change nothing else, and measure the resulting change in executable behavior. Repair benchmarks evaluate whether a revision passes its tests (Jimenez et al., 2024) and, increasingly, whether it is narrowly scoped (Zhu et al., 2026; Yang et al., 2026a), but not whether it follows from the model's belief about what changed. Self-verification methods run a check at every step (Gou et al., 2024; Manakul et al., 2023), so the model never decides whether to check. Work on sycophancy, distracting context, conflicting knowledge, and indirect prompt injection shows that supplied text can override what a model would otherwise conclude or do (Sharma et al., 2023; Shi et al., 2023; Xie et al., 2024; Greshake et al., 2023).

To trace the full path from evidence to action, we ask the model to state its belief before it acts: the environments it still considers possible—its *plausible hypotheses*, and the one it considers most likely—its *leading hypothesis*. An *environment* is one combination of operating assumptions (e.g., whether a pump keeps running after the stop command; see Figure 1), and the model is either told which assumptions changed, or must infer them from the failure. Our testbed covers operational, physical, and software systems: discrete controllers for settings ranging from industrial pumps and measurement calibration to service failover and authentication; feedback control of a continuous dynamical system (Yuan et al., 2022); and Python programs whose test reports are truthful, missing, or fabricated (Section 2). In every task, we know which environments the evidence allows and run each revision in all of them, so both belief and action are scored exactly. We call an action a *selective repair* if it fixes what changed and nothing else: it completes the task in the true environment without depending on assumptions that no longer hold, still succeeds in the original environment, and leaves unchanged all behavior tied to assumptions that still hold (Section 2).

We make the following contributions:

1. We introduce an executable testbed for epistemic control. Each task specifies exactly which environments the evidence allows, every revision runs in all of them, and the model's stated belief can be replaced while everything else stays fixed, so we can test whether belief drives action and whether evidence is checked, against exact ground truth.
2. We show that models usually keep the true environment among their plausible hypotheses but rarely rank it first: with two hidden changes, it is plausible in 80.0–97.8% of tasks but leading in only 5.8%, below the 27.8% chance baseline.
3. We show that models act on the beliefs they state, but a correct belief is not enough. Replacing a wrong leading hypothesis with the truth raises selective repair from 3.3% to 35.4%, yet even when

told exactly what changed, models reach at most 36.1%, often rewriting more of the system than the change requires.
4. We show that models act on the evidence they are given but rarely check it. A fabricated failure report raises edits to correct programs by 84.5 percentage points, and six of eight models never request an available test when a fabricated report says faulty code passes.

We will release the task definitions, prompts, evaluators, model responses, execution traces, and analysis code needed to reproduce every reported result.

2 Measuring Epistemic Control

2.1 What do we measure?

What does the model believe changed? Does the model keep the true environment among the explanations it considers plausible, and does it select it as its leading hypothesis? Each task defines a finite set of possible environments Θ. An environment is one complete combination of the task's operating assumptions. For instance, in the pump control task, one environment is that the displayed tank level lags, the pump stops on time, and the valve opens immediately. The source environment $\theta^{(0)}$ captures the assumptions under which the original system operates correctly, and the true environment $\theta^\star$ describes the operating conditions currently in force. The assumptions that changed are

$$\Delta^\star = \{j : \theta^\star_j \neq \theta^{(0)}_j\},$$

where j indexes the task's operating assumptions. The model never observes $\theta^\star$ directly. Instead, it receives an evidence sequence $e_{1:t} = (e_1, \ldots, e_t)$ which may be complete, incomplete, absent, or false. We write $\Theta(e)$ for the environments the evidence has not ruled out. After seeing the evidence, the model reports the environments it still considers plausible, $\widehat{\Theta} \subseteq \Theta$, and selects a leading hypothesis, $\widehat{\theta} \in \widehat{\Theta}$. When more than one environment fits the evidence, the model should keep all of them in its plausible set. The leading hypothesis is correct when $\widehat{\theta} = \theta^\star$. When the leading hypothesis is incorrect, we replace it with $\theta^\star$ while holding the model, task, evidence, and source environment fixed. A change in the revised program's subsequent executable behavior would then indicate a causal relationship between the model's stated belief and its action.

When is a revision a selective repair? Does the model's revision affect only what changed, and leave everything else intact? After stating its belief, the model writes a revised program $\mathtt{P}$, and we run $\mathtt{P}$ from a fresh state in the true environment $\theta^\star$, in the source environment $\theta^{(0)}$, and on checks tied to each assumption outside $\Delta^\star$. We call $\mathtt{P}$ a *selective repair* when it works in the changed environment (*task completion*), removes behavior based on invalid assumptions (*obsolete-behavior removal*), continues to work in the original environment (*source compatibility*), and leaves behavior tied to assumptions that remain valid unchanged (*scope preservation*). In no-change tasks, $\Delta^\star$ is empty and intervention is unnecessary. Because the belief is stated before the revision, we can also find revisions that fail even though $\widehat{\theta} = \theta^\star$. In these cases the model identified what changed and did not act on it correctly. Between receiving evidence and acting on it, an agent decides what to believe changed, how to act on that belief, and which evidence to trust, verify, or gather. We measure these decisions along one path,

$$e_{1:t} \longrightarrow (\widehat{\Theta}, \widehat{\theta}) \longrightarrow \mathtt{P} \longrightarrow Y(\mathtt{P}),$$

in which the evidence $e_{1:t}$ leads to a stated belief $(\widehat{\Theta}, \widehat{\theta})$, the belief leads to a revision $\mathtt{P}$ of the system, and running $\mathtt{P}$ gives its outcome $Y(\mathtt{P})$. Each experiment changes one point on this path and holds the rest fixed. In the Python experiments the model does not state a belief, so they measure the path from evidence directly to action. Because each comparison changes a single point on the path, a difference in outcome can be attributed to that point.

Which evidence does the model trust, verify, and gather? Can a model decide whether a report deserves trust, when to verify it, and which observation to gather? We study trust and verification with Python programs whose true state we establish before the model sees them. Each program runs against its complete certified test suite, and it is correct if every test passes and faulty otherwise. The model then receives the program with a truthful test report, no report, or a fabricated report that states

the opposite result. The appropriate action depends only on the program's state. A correct program should be left unchanged (`NO_CHANGE`), and a faulty program should be replaced with one that passes the suite. We compare a condition in which the model must request a fresh test result with one in which it is supplied automatically. To study which observations a model gathers, we return to the controllers. Before stating a final belief and acting, the model chooses two diagnostic observations, e_{t+1} and e_{t+2}. We compare these choices with two observations assigned at random and with two chosen by a splitting rule. Without knowing the true environment, the rule picks the observation that splits the environments still in $\Theta(e)$ most evenly, that is, the one expected to leave the fewest of them possible. Separating a program's state from the report about it shows whether the model's action follows the report or the program, and comparing chosen with assigned observations shows whether the model's own choices make its evidence more informative.

2.2 Where can every decision be checked exactly?

Operational systems. We construct executable tasks across six operational settings: pump control, service failover, transaction recovery, authentication, stream replay, and measurement calibration. A finite state controller is a program that, at each step, reads an observation, updates a small internal state, and chooses an action. Each task has three binary assumptions, yielding eight possible environments. Because states and observations are finite, we know exactly which of the eight environments the evidence allows, and we can run any controller in all eight. Starting from a source program certified to work under the original assumptions, we change zero, one, or two assumptions. We use machine-checked certificates to verify that each changed task fails as intended and admits a selective repair. We experiment with three evidence conditions: (a) disclosed change, in which the model is told what changed; (b) hidden change, in which it must infer the change from an execution trace; and (c) no change, in which the source assumptions remain valid and intervention is unnecessary. See Appendix Section B.1 for concrete task examples, the exhaustive one-edit search for two-change tasks, artifact hashing details, and the full certification procedure.

Physical systems. We repeat the controller experiments in CartPole (Yuan et al., 2022), a physics simulation with continuously varying positions and forces. Each of 12 changed versions changes one or two of three properties, at one of two magnitudes: the pole's length and mass, a constant extra push from the motor, or a fixed error in the measured pole angle.

Software systems. We use 40 nonoverlapping BigCodeBench programs from PDB-Single (Zhuo et al., 2025; Zhu et al., 2026), each with certified correct and faulty versions, to compare actions under truthful, absent, and fabricated test reports.

Models and scoring. We evaluate a fixed panel of five to ten models per experiment. Each model must first complete format and atomic-task checks, and we report every qualification outcome (Appendix D). All analyses preserve the matched design, and averages across models weight each model equally. We report 95% confidence intervals (CIs) computed by resampling tasks with replacement (source programs for Python); every model and condition uses the same resampled tasks, so comparisons remain paired. Every revision is executed, and deterministic code performs all scoring; no language model judges another model's output.

3 Results

3.1 Models often retain the truth but rarely make it their leading hypothesis

We first ask whether models can identify which operating rules changed when a previously working system encounters a change in its environment—for example, when a valve that previously opened immediately begins opening only after a delay, or a sensor begins reporting with a delay. Each task defines three independent assumptions about actuation, sensing, and recovery, and changes one or two from the environment in which the controller originally worked. Before acting, the model is instructed to report a set of plausible hypotheses, $\widehat{\Theta}$, and identify its leading hypothesis, $\widehat{\theta} \in \widehat{\Theta}$. Under disclosed-change conditions, the model is told exactly which operating assumptions changed. Under hidden-change conditions, that information is withheld, and the model must infer the changes from the system's failure and the available execution evidence. The task and controller interface are otherwise identical.

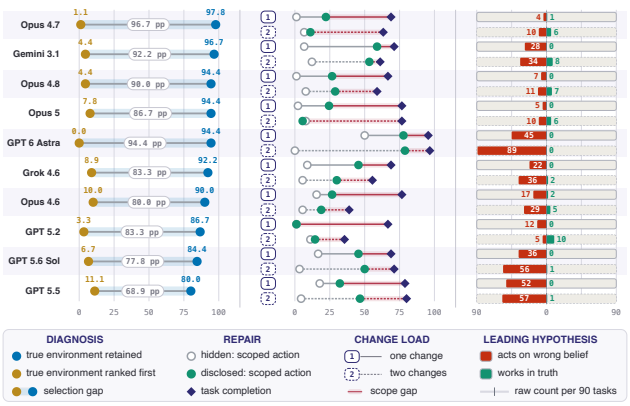


Figure 2: **From plausible hypotheses to action. Left:** hypotheses with two hidden changes. **Center:** selective repair with the changes hidden and disclosed, and task completion after disclosure; rows marked 1 and 2 show one and two changed assumptions. **Right:** with hidden changes and a wrong leading hypothesis, tasks whose controller works only in the environment of that hypothesis (left) or only in the true environment (right). All panels use first attempts over the same 90 tasks per row.

When models must infer *two hidden changes*, they usually retain the hypothesis matching the true environment in $\widehat{\Theta}$ but select a different leading hypothesis. Across models, the true environment remains among the plausible hypotheses in 80.0–97.8% of tasks but is selected as the leading hypothesis in at most 11.1% of tasks ($\theta^\star \in \widehat{\Theta}$ but $\widehat{\theta} \neq \theta^\star$). Figure 2, left, shows this gap for each model, and Table 6 in the appendix reports the quantities below for each model. $\widehat{\Theta}$ lists 5.4 of the 8 possible environments on average, although only 3.7 remain in $\Theta(e_1)$, the set of environments the initial failure e_1 leaves possible. Choosing $\widehat{\theta}$ uniformly at random from $\Theta(e_1)$ would give $\widehat{\theta} = \theta^\star$ in 27.8% of tasks; the models reach $\widehat{\theta} = \theta^\star$ in 5.8% on average, 22.0 pp lower (95% CI 19.6 to 24.2), and every model falls below 27.8%. The true environment always differs from the source in two assumptions, but some environments in $\Theta(e_1)$ explain the initial failure with a single change. Models choose one of these single-change environments as $\widehat{\theta}$ in 51.3% of tasks on average (24.4–77.8% across models), compared with 38.9% if $\widehat{\theta}$ were chosen at random from $\Theta(e_1)$. On average, models favor the simplest explanation, which is a reasonable default when the prompt does not say how many assumptions changed but always misses the truth on these tasks. Thus, we can conclude that *models often retain the truth within their set of plausible hypotheses without giving it priority over competing hypotheses.*

Revealing exactly which operating assumptions changed removes uncertainty about the cause of the failure, but models still fail to reliably translate that knowledge into action (Figure 2, center, and Appendix Table 4). Compared with hidden changes, disclosure increases the rate of selective repair for nine of ten models in each of the one-change and two-change conditions. For tasks with one changed assumption, GPT 5.2 shows no improvement from disclosure, while gains for the other models range from 11.1 to 52.2 pp. For tasks with two changed assumptions, gains range from 3.3 to 78.9 pp for nine models, while Claude 5 Opus declines by 2.2 pp. With two changes, the differences for Claude 4.7 Opus, GPT 5.2, and Claude 5 Opus are too small to distinguish from no change. After disclosure, every model's leading hypothesis correctly describes the changed environment on every task. Even so, eight of ten models produce a selective repair on fewer than half of their tasks when one environmental assumption changes, compared with seven of ten when two assumptions change. *Thus, difficulty identifying the true environment accounts for only part of the observed failures.*

Even when their leading hypothesis correctly identifies what changed, models often act beyond what the evidence justifies. A controller can work in the changed environment while retaining behavior that depends on an invalid assumption, failing in the original environment, or altering behavior supported by assumptions that remain valid. In disclosed one-change tasks, task completion exceeds selective repair for every model, with gaps ranging from 12.2 to 65.6 pp (Figure 2, center). Models often change controller behavior that the new environment did not require them to change. Broad rewriting appears for every model, affecting 47.5% to 99.7% of its first attempts, while changes to fallback behavior range from 19.7% to 91.7%. Examples include waiting after any pump observation

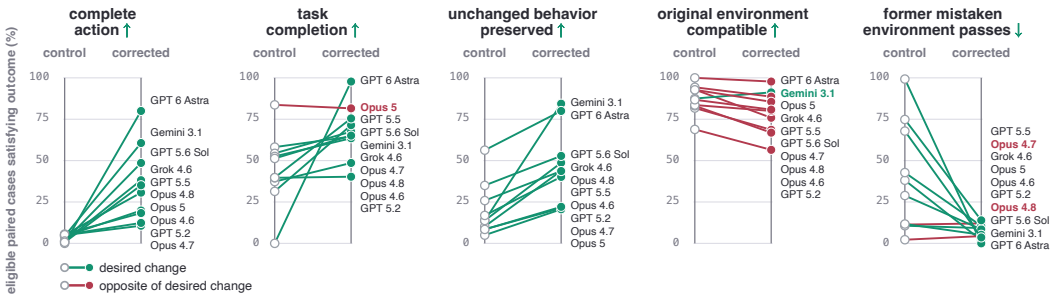


Figure 3: **Correcting the stated belief redirects action.** The unfilled endpoint shows the control action based on the model's original belief; the filled endpoint shows its action on the same task after the oracle supplies the true environment before an otherwise identical request. Green lines indicate movement in the desired direction, red lines in the opposite direction, and gray no change. The oracle intervention improves selective repair and preservation of behavior tied to unchanged assumptions for every model, and task completion for nine, but compatibility with the original environment declines for nine. Most actions produced after oracle input still fail at least one requirement.

rather than only a pending shutdown, reauthenticating across unrelated token states rather than only after a scope error, and cross-checking every sensor reading rather than only when sensors disagree. For nine of ten models, hiding what changed also increases the rate at which the controller adds contingency behavior for every assumption that actually remained valid. These labels describe differences observed by executing the controllers; they do not treat textual edit size as a proxy for behavioral scope. Together, these findings show that *a correct belief and successful task completion are not sufficient for action that remains within the bounds of the evidence.*

Effective agents acting in the real world—where operating environments continually evolve—should be able to learn from the outcomes of failed actions to revise their beliefs and decide what to do next. We test whether models learn from execution feedback under a fixed interaction budget of six attempts per episode. Most observed gains in the cumulative rate of selective repair occur within the first two to three attempts (Appendix Table 5), and by attempt 3, eleven of thirteen models show no further improvement (Claude 4.7 Opus gains another 2.2 pp, while DeepSeek V4 Pro gains another 1.1 pp). Within six attempts, the best model, GPT 6 Astra, produces a selective repair in 73 of its 90 hidden-change episodes (81.1%), and no other model exceeds the 27.8% of GPT 5.6 Sol. Even GPT 6 Astra ends 17 episodes with a controller that passes the visible test suite but is not a selective repair. Thus, models may revise their hypotheses repeatedly without producing an action that is both effective and supported by the evidence.

3.2 Models often fail to translate correct beliefs into effective action

Does changing what a model believes change what it does? A reported hypothesis could be generated alongside an independently produced action; if it instead guides the action, the resulting controller should behave appropriately for the environment that the hypothesis describes. The prompt asks the model to act on its diagnosis but cannot enforce it, so we first test whether stated belief and resulting action are in fact behaviorally aligned. We isolate tasks in which the environment described by the model's leading hypothesis, $\widehat{\theta}$, differs from the true environment, $\widehat{\theta} \neq \theta^{\star}$, and execute each controller in both $\widehat{\theta}$ and $\theta^{\star}$, counting it as working in an environment when it is a selective repair there.

With one hidden change, 605 of 900 tasks, summed over the ten models, have a wrong leading hypothesis and a readable controller. Among these cases, 228 controllers work only in the environment described by the mistaken leading hypothesis, three work only in the true environment, and 374 work in neither. With two hidden changes, 840 of 900 tasks meet both conditions. Among these cases, 337 controllers work only in $\widehat{\theta}$, 46 work only in $\theta^{\star}$, and 457 work in neither. This asymmetry holds for every model with one change and nine of ten models with two (Figure 2, right). The exception is GPT 5.2 with two changes, whose controllers work only in $\widehat{\theta}$ in 5 tasks and only in $\theta^{\star}$ in 10. These failures are structured: models often produce functioning controllers for $\widehat{\theta}$, even when $\widehat{\theta}$ is wrong. Simply including the hypothesis that matches the true environment in $\widehat{\Theta}$ is therefore not enough. The

Model	Pairs	Selective ↑	Target ↑	Scope ↑	Source ↑	Former wrong ↓
Claude 4.7 Opus	141	5.0→**10.6** (+5.7)	58.2→**64.5** (+6.4)	8.5→**21.3** (+12.8)	**92.9**→75.9 (−17.0)	**11.3**→12.1 (+0.7)
Gemini 3.1 Pro	160	4.4→**60.6** (+56.2)	54.4→**67.5** (+13.1)	13.8→**84.4** (+70.6)	87.5→**91.2** (+3.7)	38.1→**1.9** (−36.2)
Claude 4.8 Opus	137	5.1→**30.7** (+25.5)	52.6→**63.5** (+10.9)	10.2→**43.8** (+33.6)	**81.8**→68.6 (−13.1)	**2.2**→4.4 (+2.2)
Claude 5 Opus	141	4.3→**19.9** (+15.6)	**83.7**→81.6 (−2.1)	5.0→**20.6** (+15.6)	**94.3**→88.7 (−5.7)	10.6→**9.2** (−1.4)
GPT 6 Astra	135	0.0→**80.0** (+80.0)	0.0→**97.8** (+97.8)	56.3→**80.0** (+23.7)	**100.0**→97.8 (−2.2)	99.3→**0.0** (−99.3)
Grok 4.6	152	1.3→**38.2** (+36.8)	51.3→**65.1** (+13.8)	13.8→**48.7** (+34.9)	**92.8**→85.5 (−7.2)	42.8→**10.5** (−32.2)
Claude 4.6 Opus	142	5.6→**18.3** (+12.7)	37.3→**48.6** (+11.3)	16.9→**40.1** (+23.2)	**83.1**→66.9 (−16.2)	28.9→**8.5** (−20.4)
GPT 5.2	154	5.2→**12.3** (+7.1)	39.6→**40.3** (+0.6)	8.4→**22.1** (+13.6)	**68.8**→56.5 (−12.3)	11.7→**5.2** (−6.5)
GPT 5.6 Sol	140	1.4→**48.6** (+47.1)	31.4→**71.4** (+40.0)	35.0→**52.9** (+17.9)	**83.6**→80.0 (−3.6)	67.9→**3.6** (−64.3)
GPT 5.5	151	0.7→**35.1** (+34.4)	39.7→**75.5** (+35.8)	25.8→**43.7** (+17.9)	**86.8**→80.8 (−6.0)	74.8→**13.9** (−60.9)
Aggregate	–	3.3→**35.4** (+32.1) [28.8, 35.6]	44.8→**67.6** (+22.8) [18.5, 26.9]	19.4→**45.7** (+26.4) [23.1, 29.7]	**87.2**→79.2 (−8.0) [−11.3, −4.7]	38.8→**6.9** (−31.8) [−34.3, −29.4]

Table 1: **Paired outcomes under an oracle intervention on belief** (%). Each pair is one task in which the model's initial leading hypothesis does not match the true environment, evaluated under matched control and oracle conditions. Because eligibility depends on each model's initial errors, comparisons should be made within rows rather than across models. Each cell is control → oracle (Δ pp), where $\Delta = \text{corrected} - \text{control}$. Arrows indicate the desired direction, and the better endpoint is bold. "Former wrong" is the rate at which the controller passes in the environment the model originally but incorrectly ranked first. Green indicates improvement and red indicates decline; ties are unshaded. The aggregate row is reported with 95% CIs for Δ.

resulting action closely follows $\widehat{\theta}$. GPT 6 Astra shows the clearest example of this (Figure 2, left and right): with two hidden changes, $\widehat{\Theta}$ contains the hypothesis matching the true environment in 85 of 90 tasks, but the model never selects it as $\widehat{\theta}$. Its controller works only in the environment described by $\widehat{\theta}$ in 89 tasks and only in the true environment in none.

We next test whether this relationship is causal. In cases where the model's leading hypothesis is incorrect, we compare its original action with an action produced after an oracle supplies the true environment. The model, task, execution evidence, preceding conversation, and action request remain fixed; only the belief supplied before action changes. The oracle intervention redirects action for every model, increasing selective repair by 5.7 to 80.0 pp (Figure 3). Task completion improves for nine models, and preservation of behavior supported by unchanged assumptions improves for all ten. Eight models also become less likely to act on their former mistaken belief. Even with the correct belief, models may still take ineffective actions: after the oracle intervention, eight of ten models produce selective repairs on fewer than half of eligible tasks, and compatibility with the original environment declines for nine. Table 1 reports the paired rates for every model.

What do models learn from failed actions? A feedback round consists of a hypothesis report and system revision; if the revision fails the visible test suite, its execution result is returned before the next round. Appendix Figure 5 follows the 810 hidden-change episodes of the nine models that passed qualification (upper block of Appendix Table 5, Appendix D). Most belief revision occurs after the first returned result: averaged over models, the number of plausible hypotheses falls from 5.69 to 2.00, 87.6% of sets contract, and 184 of the 187 observed promotions of the true environment to leading hypothesis occur at this transition. Progress then stalls. We count an environment as ruled out once some system run so far would have produced a different execution result in it (outcome, score, progress, failure code, or trace) from the one returned. Averaged over models, 27.0% of later reports (10.9% to 69.7% across models) still include at least one ruled-out environment, and in 90.3% of later rounds that do not produce a selective repair, the model reports higher confidence than in the previous round. Because an episode ends after a visible test-suite pass, later rounds contain a smaller and more difficult subset; their lower action rates do not show that feedback causes performance to decline. Instead, the trajectories show that feedback can narrow models' beliefs without ensuring that they maintain its implications or translate them into effective action.

Does the gap between correct belief and effective action persist under continuous dynamics? We repeat the experiment in CartPole (Yuan et al., 2022), a physics simulation with continuously varying positions and forces. A controller, a short program, reads sensor measurements of a cart and a pole hinged on it, and decides how hard to push the cart, with two goals: keep the pole upright and stop the cart near the center of its track. The model receives a controller that meets both goals in the original system but fails in each of 12 changed versions. Each version changes one or two of three properties, at one of two magnitudes: the pole's length and mass, a constant extra push from

the motor, or a fixed error in the measured pole angle. The model then chooses a set of plausible hypotheses and a leading hypothesis from a list of candidate versions, and writes a new controller, with no feedback or second attempt. When the change is hidden, the only evidence available to the model is one or two readings of cart position and pole angle taken within the first 0.27 seconds. When it is disclosed, the prompt states exactly what changed. A new controller is a selective repair if it runs without error, meets both goals in the changed system within fixed limits on motion and force, respects those limits in every version the evidence allows, and still meets both goals in the original system, much as the original controller did.

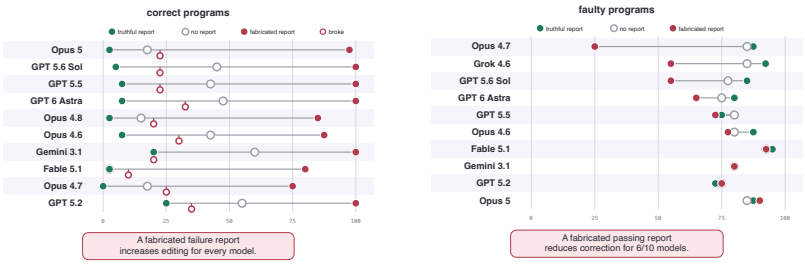
Appendix Table 11 reports each model's belief and action in both conditions. With the change hidden, GPT 6 Astra, Gemini 3.1 Pro, DeepSeek V4 Pro, Claude 5.1 Fable, and Qwen 3.8 write only 1 selective repair among their 60. With the change disclosed, every model's stated belief matches the true version in all 12 cases, but only 5 of 60 controllers are selective repairs: 4 of 12 for GPT 6 Astra, 1 for Gemini 3.1 Pro, and none for the other three. Qwen 3.8 returns no program in any of its 24 responses, hidden or disclosed. Of the 60, 21 run and 12 meet both goals in the changed system. All 12 keep the pole upright in the original system, but 7 leave the cart 0.5 to 1.1 m from the center—where the goal allows 0.1 m—because they apply a fixed correction for the disclosed change at every step, for example by subtracting 1 N from every push. Notably, Gemini 3.1 Pro's successful controller instead checks for and correctly estimates the force bias during execution, and promptly compensates for it. Similarly to the finite state tasks, models often identify the cause precisely yet still modify behavior that should remain intact.

3.3 Models act on evidence they rarely verify

Are models easily influenced by false evidence? Before the model sees a task, we run each program against the complete certified test suite to establish whether it passes or fails. A program receives `PASS` only if every test passes and `FAIL` otherwise. The experimental protocol then supplies either that verified result, no report, or a report that deliberately states the opposite outcome. We call the last case a fabricated report: a fabricated passing report accompanies a program that fails the suite, while a fabricated failure report accompanies a program that passes it.

We observe that relative to a truthful passing report, a fabricated failure report increases actual changes by 84.5 pp on average across models (95% CI 79.8 to 89.0; 75–95 pp for every model), and those changes break 10–35% of programs that previously worked correctly. The reverse manipulation also changes action, but less: when a fabricated passing report accompanies faulty code, models correct 12.8 pp fewer programs on average than they do without a report (95% CI 7.8 to 18.0). Most of this decline comes from three models: Claude 4.7 Opus (60.0 pp), Grok 4.6 (30.0 pp), and GPT 5.6 Sol (22.5 pp). None of the other seven changes by more than 10 pp. The two manipulations are therefore asymmetric. A fabricated failure report makes every model alter—and sometimes break—working code. A fabricated passing report persuades some models to leave faulty code unchanged. Thus, models tend to act on what they are told, even when it contradicts how the underlying system actually behaves (Figure 4). Examples are shown in Appendix Section D.5.

Can models recognize when evidence requires verification? Models are given the opportunity to request an independent test before deciding whether to change a program accompanied by a test report. They see the task description and the complete source program, but not the test definitions or an execution trace, and they cannot execute the program directly. The prompt allows the model to request a fresh execution result, act immediately, or leave the system unchanged. If a model requests verification, the benchmark runs the unchanged program against the complete certified test suite and returns only a fresh `PASS` or `FAIL` result before the model chooses its final action. Models rarely request verification, especially when a fabricated passing report says everything is fine. With a faulty program and a fabricated passing report, six of eight models never request verification in 40 opportunities, and the other two request it twice each. With a correct program and a fabricated failure report, models request it in 14.7% of opportunities on average (0 to 20 of 40 per model). For correct programs, the appropriate action is `NO_CHANGE`, and a fresh `PASS` directly contradicts the fabricated failure report. When this result is supplied automatically, models choose the appropriate action in 92.5% of cases on average, compared with 15.3% when verification is optional (difference 77.2 pp; 95% CI 62.8 to 86.6). For faulty programs, the appropriate action is a replacement that passes the certified tests. Most models already make one without verifying (60.0–87.5%), and supplying the result makes no detectable difference (75.0% versus 74.1%; 95% CI for the difference -18.5 to 17.4



(a) (b)

Figure 4: **Supplied evidence changes action.** One row per model. (a) **A false alarm prompts unnecessary action:** edit rates on correct programs after a truthful passing report, no report, or a fabricated failure report; hollow red circles show how often those edits break the program. (b) **False reassurance suppresses necessary action:** successful correction of faulty programs after a truthful failure report, no report, or a fabricated passing report. Rates for each model in Appendix Table 7.

pp). Models can therefore act on a fresh result showing that a program works, but rarely initiate the check required to obtain it (Appendix Table 8).

Can models gather useful evidence when uncertain? We then ask whether models can decide what evidence to gather when the environment remains uncertain. Models receive two opportunities to investigate it by selecting diagnostic tests, which the simulator executes from a fresh state and returns as new evidence, before reporting a final belief and acting on it. Before the first observation, the leading hypothesis is correct in 2 to 15 of 48 tasks per model. After two observations selected by the model, it is correct in 93.8% of tasks on average (41 to 47 of 48 per model), but selective repair occurs in only 47.1% (9 to 37 of 48; difference 46.6 pp, 95% CI 39.1 to 54.2). Better evidence therefore substantially improves beliefs without closing the gap between belief and action.

To determine whether this improvement comes from choosing good observations or merely receiving additional evidence, we compare observations selected by the model with observations assigned at random or by an oracle under the same budget of two observations. The oracle picks the observation that most evenly splits the remaining hypotheses, without knowing the true environment or planning the subsequent action. When models select the observations, selective repair occurs in 47.1% of tasks on average, compared with 49.7% under random assignment (difference -2.6 pp; 95% CI -7.6 to 2.6). Random assignment also makes the leading hypothesis correct more often (99.7% versus 93.8%; 95% CI for the difference 3.4 to 8.9 pp). Oracle assignment results in no detectable advantage over random assignment either (52.6% versus 49.7% complete action; 95% CI for the difference -1.6 to 7.3 pp). In summary, additional observations make the leading hypothesis more accurate, but letting models choose those observations improves neither beliefs nor actions relative to random assignment (Appendix Table 9). Overall, models respond strongly to the evidence they receive but struggle with epistemic control: judging if the evidence can be trusted, whether to verify it, and which observation is most informative to collect.

4 Conclusion

We set out to measure what happens between the moment an agent receives evidence that a working system has failed and the moment it changes that system. First, models often keep the true environment among their plausible hypotheses but choose a different leading hypothesis, and their revision follows that choice: replacing a wrong leading hypothesis with the truth raises selective repair from 3.3% to 35.4%. Second, a correct belief is not enough. With the change disclosed, most models still change more or less of the system than the change requires. Finally, models act on the reports they are given. A fabricated failure report leads every model to change working programs, and models rarely request an independent test. For anyone building or evaluating agents, task completion is therefore an incomplete measure of success. It should be accompanied by checks of which belief the agent acted on, whether its change stayed within what the evidence supports, and if the evidence deserved trust.

AI-Assistance Disclosure

We used large language model assistants to write and debug experiment and analysis code, render figures, audit reported numbers against result files, and edit parts of the text for grammar and repetition. The authors directed the research, reviewed and revised all assisted output, and take full responsibility for the content of the paper. No language model was used to score or judge the responses of the evaluated models.

References

A. Abdi et al. Llama See, Llama Do: A mechanistic perspective on contextual entrainment and distraction in LLMs. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics*, 2025. doi: 10.18653/v1/2025.acl-long.791. URL https://aclanthology.org/2025.acl-long.791/.

Michael Ahn et al. Do as I can, not as I say: Grounding language in robotic affordances. In *Proceedings of the 6th Conference on Robot Learning*, 2022. URL https://arxiv.org/abs/2204.01691.

Carlos E. Alchourrón, Peter Gärdenfors, and David Makinson. On the Logic of Theory Change: Partial Meet Contraction and Revision Functions, 1985. URL https://www.cambridge.org/core/journals/journal-of-symbolic-logic/article/abs/on-the-logic-of-theory-change-partial-meet-contraction-and-revision-functions/7ED837BAD5FB6D9A7C77906D73527F9C.

Pierre Andrews, Amine Benhalloum, Matteo Bettini, Virginie Do, Romain Froger, Emilien Garreau, Jean-Baptiste Gaya, Hugo Laurençon, Maxime Lecanu, Kunal Malkan, Dheeraj Mekala, Grégoire Mialon, Pierre Ménard, Gerard Moreno-Torres Bertran, Ulyana Piterbarg, Mathieu Rita, Andrey Rusakov, Thomas Scialom, and Mengjue Wang. ARE: Scaling up agent environments and evaluations, 2025. URL https://arxiv.org/abs/2509.17158.

Aitor Arrieta, Pablo Valle, and Shaukat Ali. Search-based automated program repair of CPS controllers modeled in Simulink-Stateflow, 2024. URL https://arxiv.org/abs/2404.04688.

Akari Asai, Zeqiu Wu, Yizhong Wang, Avirup Sil, and Hannaneh Hajishirzi. Self-RAG: Learning to retrieve, generate, and critique through self-reflection. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://arxiv.org/abs/2310.11511.

Ruairidh M. Battleday, Kai Sandbrink, Jimi Cullen-Drohan, Zihan Yan, Timothy Muller, Clare Maguire, Ales Kubicek, Fraser Greenlee-Scott, Sukrit Sumant, Tri Dao, Jürgen Schmidhuber, Michal Valko, Joshua Tenenbaum, Thomas L. Griffiths, Zeb Kurth-Nelson, and James C. R. Whittington. DiG-bench: Discovery in games, 2026. URL https://arxiv.org/abs/2608.12593.

Leonardo Bertolazzi, Katya Tentori, and Raffaella Bernardi. FALSIFYBENCH: Evaluating inductive reasoning in LLMs with rule discovery games, 2026. URL https://arxiv.org/abs/2606.04751.

David H. Blum, Javier Arroyo, Sen Huang, Ján Drgoňa, Filip Jorissen, Harald Taxt Walnum, Yan Chen, Kyle Benne, Draguna Vrabie, Michael Wetter, and Lieve Helsen. Building optimization testing framework (BOPTEST) for simulation-based benchmarking of control strategies in buildings. *Journal of Building Performance Simulation*, 14(5):586–610, 2021. doi: 10.1080/19401493.2021.1986574. URL https://escholarship.org/uc/item/7xb9g6hv.

Carlo Bosio and Mark W. Mueller. Synthesizing interpretable control policies through large language model guided search, 2024. URL https://arxiv.org/abs/2410.05406.

Lukas Brunke, Melissa Greeff, Adam W. Hall, Zhaocong Yuan, Siqi Zhou, Jacopo Panerati, and Angela P. Schoellig. Safe learning in robotics: From learning-based control to safe reinforcement learning. *Annual Review of Control, Robotics, and Autonomous Systems*, 5:411–444, 2022. doi: 10.1146/annurev-control-042920-020211. URL https://arxiv.org/abs/2108.06266.

Jun Shern Chan, Neil Chowdhury, Oliver Jaffe, James Aung, Dane Sherburn, Evan Mays, Giulio Starace, Kevin Liu, Leon Maksin, Tejal Patwardhan, Lilian Weng, and Aleksander Mądry. MLE-bench: Evaluating Machine Learning Agents on Machine Learning Engineering. In *International Conference on Learning Representations*, 2025.

Hung-Ting Chen et al. Rich knowledge sources bring complex knowledge conflicts: Recalibrating models to reflect conflicting evidence. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, 2022. doi: 10.18653/v1/2022.emnlp-main.146. URL https://aclanthology.org/2022.emnlp-main.146/.

Tingting Chen, Beibei Lin, Zifeng Yuan, Qiran Zou, Hongyu He, Anirudh Goyal, Yew-Soon Ong, and Dianbo Liu. HypoSpace: A diagnostic benchmark for set-valued hypothesis generation under underdetermination and sublinear coverage bounds. In *Proceedings of the 43rd International Conference on Machine Learning*, 2026a. URL https://openreview.net/forum?id=QpjtK65JHO. Spotlight.

Yimeng Chen, Piotr Piękos, Mateusz Ostaszewski, Firas Laakom, and Jürgen Schmidhuber. PhysGym: Benchmarking LLMs in Interactive Physics Discovery with Controlled Priors, 2025a. URL https://arxiv.org/abs/2507.15550.

Yinfang Chen, Manish Shetty, Gagan Somashekar, Minghua Ma, Yogesh Simmhan, Jonathan Mace, Chetan Bansal, Rujia Wang, and Saravan Rajmohan. AIOpsLab: A Holistic Framework to Evaluate AI Agents for Enabling Autonomous Clouds, 2025b.

Zhenhao Chen, Yongqiang Chen, Chenxi Liu, Junchi Yu, Xiangchen Song, Zijian Li, Jialin Li, Philip Torr, Bo Han, and Kun Zhang. CausalGame: Benchmarking Causal Thinking of LLM Agents in Games, 2026b. URL https://arxiv.org/abs/2607.04293.

Edoardo Debenedetti, Jie Zhang, Mislav Balunović, Luca Beurer-Kellner, Marc Fischer, and Florian Tramèr. AgentDojo: A dynamic environment to evaluate prompt injection attacks and defenses for LLM agents. In *Advances in Neural Information Processing Systems (Datasets and Benchmarks Track)*, 2024. URL https://arxiv.org/abs/2406.13352.

James J. Downs and Ernest F. Vogel. A plant-wide industrial process control problem. *Computers & Chemical Engineering*, 17(3):245–255, 1993. doi: 10.1016/0098-1354(93)80018-I.

Mohamad Fakih, Rahul Dharmaji, Yasamin Moghaddas, Gustavo Quiros Araya, Oluwatosin Ogundare, and Mohammad Abdullah Al Faruque. LLM4PLC: Harnessing large language models for verifiable programming of PLCs in industrial control systems. In *Proceedings of the 46th IEEE/ACM International Conference on Software Engineering: Software Engineering in Practice*, 2024. doi: 10.1145/3639477.3639743. URL https://arxiv.org/abs/2401.05443.

Aaron Fanous, Jacob N. Goldberg, Ank A. Agarwal, Jianhao Lin, Angela Zhou, Roxana Daneshjou, and Sanmi Koyejo. SycEval: Evaluating LLM Sycophancy, 2025. URL https://arxiv.org/abs/2502.08177.

Kanishk Gandhi, Michael Y. Li, Lyle Goodyear, Agam Bhatia, Louise Li, Aditi Bhaskar, Mohammed Zaman, and Noah D. Goodman. BoxingGym: Benchmarking Progress in Automated Experimental Design and Model Discovery, 2025. URL https://arxiv.org/abs/2501.01540.

Luyu Gao et al. RARR: Researching and revising what language models say, using language models. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics*, 2023. doi: 10.18653/v1/2023.acl-long.910. URL https://aclanthology.org/2023.acl-long.910/.

Aniketh Garikaparthi et al. ResearchGym: Evaluating Language Model Agents on Real-World AI Research, 2026. URL https://arxiv.org/abs/2602.15112.

Federico Germani et al. Source framing triggers systematic bias in large language models. *Science Advances*, 2025. doi: 10.1126/sciadv.adz2924. URL https://www.science.org/doi/10.1126/sciadv.adz2924.

Milapji Singh Gill, Javal Vyas, Artan Markaj, Felix Gehlhoff, and Mehmet Mercangöz. Leveraging LLM agents and digital twins for fault handling in process plants, 2025. URL https://arxiv.org/abs/2505.02076.

Zhibin Gou, Zhihong Shao, Yeyun Gong, Yelong Shen, Yujiu Yang, Nan Duan, and Weizhu Chen. CRITIC: Large language models can self-correct with tool-interactive critiquing. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://arxiv.org/abs/2305.11738.

Kai Greshake, Sahar Abdelnabi, Shailesh Mishra, Christoph Endres, Thorsten Holz, and Mario Fritz. Not what you've signed up for: Compromising real-world LLM-integrated applications with indirect prompt injection. In *Proceedings of the 16th ACM Workshop on Artificial Intelligence and Security*, 2023. URL https://arxiv.org/abs/2302.12173.

Chadi Helwe, Sultan AlRashed, and Francesco Orabona. ReviseQA: A benchmark for belief revision in multi-turn logical reasoning. In *ICML 2025 Workshop on Assessing World Models*, 2025. URL https://openreview.net/forum?id=Z4KBiAYXlI.

Qian Huang, Jian Vora, Percy Liang, and Jure Leskovec. MLAgentBench: Evaluating Language Agents on Machine Learning Experimentation, 2023.

Peter Jansen, Marc-Alexandre Côté, Tushar Khot, Erin Bransom, Bhavana Dalvi Mishra, Bodhisattwa Prasad Majumder, Oyvind Tafjord, and Peter Clark. DISCOVERYWORLD: A Virtual Environment for Developing and Evaluating Automated Scientific Discovery Agents, 2024. URL https://arxiv.org/abs/2406.06769.

Ben Jenkins. AGM-bench: Do large language models revise beliefs rationally? In *ICLR 2026 Workshop on Logical Reasoning of Large Language Models*, 2026. URL https://openreview.net/forum?id=2s1BujG84C.

Soyeong Jeong et al. Adaptive-RAG: Learning to adapt retrieval-augmented large language models through question complexity. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2024. doi: 10.18653/v1/2024.naacl-long.389. URL https://aclanthology.org/2024.naacl-long.389/.

Saurabh Jha, Rohan Arora, Yuji Watanabe, Takumi Yanagawa, Yinfang Chen, Jackson Clark, Bhavya Bhavya, Mudit Verma, Harshit Kumar, Hirokuni Kitahara, Noah Zheutlin, Saki Takano, Divya Pathak, Felix George, Xinbo Wu, Bekir O. Turkkan, Gerard Vanloo, Michael Nidd, Ting Dai, Oishik Chatterjee, Pranjal Gupta, Suranjana Samanta, Pooja Aggarwal, Rong Lee, Pavankumar Murali, Jae-wook Ahn, Debanjana Kar, Ameet Rahane, Carlos Fonseca, Amit Paradkar, Yu Deng, Pratibha Moogi, Prateeti Mohapatra, Naoki Abe, Chandrasekhar Narayanaswami, Tianyin Xu, Lav R. Varshney, Ruchi Mahindru, Anca Sailer, Laura Shwartz, Daby Sow, Nicholas C. M. Fuller, and Ruchir Puri. ITBench: Evaluating AI Agents across Diverse Real-World IT Automation Tasks, 2025.

Zhengbao Jiang, Frank F. Xu, Luyu Gao, Zhiqing Sun, Qian Liu, Jane Dwivedi-Yu, Yiming Yang, Jamie Callan, and Graham Neubig. Active retrieval augmented generation. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, 2023. doi: 10.18653/v1/2023.emnlp-main.495. URL https://aclanthology.org/2023.emnlp-main.495/.

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

Patrick Tser Jern Kon, Jiachen Liu, Xinyi Zhu, Qiuyi Ding, Jingjia Peng, Jiarong Xing, Yibo Huang, Yiming Qiu, Jayanth Srinivasa, Myungjin Lee, Mosharaf Chowdhury, Matei Zaharia, and Ang Chen. EXP-Bench: Can AI Conduct AI Research Experiments?, 2025.

Krishnamurthy et al. Can large language models explore in-context? In *Advances in Neural Information Processing Systems*, 2024. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/d951f73c521d069fefbb73396df01424-Abstract-Conference.html.

Tamera Lanham et al. Measuring faithfulness in chain-of-thought reasoning, 2023. URL https://arxiv.org/abs/2307.13702.

Jaeho Lee and Atharv Chowdhary. AssertBench: A benchmark for evaluating self-assertion in large language models, 2025. URL https://arxiv.org/abs/2506.11110. Presented at the NeurIPS 2025 workshop Evaluating the Evolving LLM Lifecycle under the title "AssertBench: A Benchmark for LLM Resistance to User-Induced Factual Bias".

Ge Lei and Samuel J. Cooper. Elicitation matters: How prompts and query protocols shape LLM surrogates under sparse observations, 2026. URL https://arxiv.org/abs/2605.04764.

Jacky Liang, Wenlong Huang, Fei Xia, Peng Xu, Karol Hausman, Brian Ichter, Pete Florence, and Andy Zeng. Code as policies: Language model programs for embodied control. In *IEEE International Conference on Robotics and Automation*, 2023. URL https://arxiv.org/abs/2209.07753.

Grace Liu, Yuxiao Qu, Jeff Schneider, Aarti Singh, and Aviral Kumar. CaRT: Teaching LLM agents to know when they know enough. In *Proceedings of the 43rd International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=hxGHnbzHqm.

Yupei Liu, Yuqi Jia, Runpeng Geng, Jinyuan Jia, and Neil Zhenqiang Gong. Formalizing and benchmarking prompt injection attacks and defenses. In *33rd USENIX Security Symposium*, 2024a. URL https://arxiv.org/abs/2310.12815.

Zihan Liu, Ruinan Zeng, Dongxia Wang, Gengyun Peng, Jingyi Wang, Qiang Liu, Peiyu Liu, and Wenhai Wang. Agents4PLC: Automating closed-loop PLC code generation and verification in industrial control systems using LLM-based agents, 2024b. URL https://arxiv.org/abs/2410.14209.

Chris Lu, Cong Lu, Robert Tjarko Lange, Jakob Foerster, Jeff Clune, and David Ha. The AI Scientist: Towards Fully Automated Open-Ended Scientific Discovery, 2024a. URL https://arxiv.org/abs/2408.06292.

Pengyuan Lu, Matthew Cleaveland, Oleg Sokolsky, Insup Lee, and Ivan Ruchkin. Repairing learning-enabled controllers while preserving what works, 2024b. URL https://arxiv.org/abs/2311.03477.

Deyun Lyu, Jiayang Song, Zhenya Zhang, Zhijie Wang, Tianyi Zhang, Lei Ma, and Jianjun Zhao. AutoRepair: Automated repair for AI-enabled cyber-physical systems under safety-critical conditions, 2023. URL https://arxiv.org/abs/2304.05617.

Alex Mallen et al. When not to trust language models: Investigating effectiveness of parametric and non-parametric memories. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics*, 2023. doi: 10.18653/v1/2023.acl-long.546. URL https://aclanthology.org/2023.acl-long.546/.

Potsawee Manakul, Adian Liusie, and Mark J. F. Gales. SelfCheckGPT: Zero-resource black-box hallucination detection for generative large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, 2023. doi: 10.18653/v1/2023.emnlp-main.557. URL https://aclanthology.org/2023.emnlp-main.557/.

Sina Mavali, David Pape, Jonathan Evertz, Samira Abedini, Devansh Srivastav, Thorsten Eisenhofer, Sahar Abdelnabi, and Lea Schönherr. No more, no less: Task alignment in terminal agents, 2026. URL https://arxiv.org/abs/2605.12233.

Mohammad Narimani and Seyyed Ali Emami. AgenticControl: An automated control design framework using large language models, 2025. URL https://arxiv.org/abs/2506.19160.

Ethan Perez et al. Discovering Language Model Behaviors with Model-Written Evaluations, 2022. URL https://arxiv.org/abs/2212.09251.

Mrinank Sharma et al. Towards Understanding Sycophancy in Language Models, 2023. URL https://arxiv.org/abs/2310.13548.

Freda Shi et al. Large language models can be easily distracted by irrelevant context. In *Proceedings of the 40th International Conference on Machine Learning*, 2023. URL https://proceedings.mlr.press/v202/shi23a.html.

Lei Song, Chuheng Zhang, Li Zhao, and Jiang Bian. Pre-trained large language models for industrial control, 2023. URL https://arxiv.org/abs/2308.03028.

Miles Turpin, Julian Michael, Ethan Perez, and Samuel R. Bowman. Language models don't always say what they think: Unfaithful explanations in chain-of-thought prompting. In *Advances in Neural Information Processing Systems*, 2023. URL https://arxiv.org/abs/2305.04388.

Javal Vyas, Milapji Singh Gill, Artan Markaj, Felix Gehlhoff, and Mehmet Mercangöz. From detection to action: Using LLM agents for fault-tolerant control, 2026. URL https://arxiv.org/abs/2606.28011.

Bryan Wilie, Samuel Cahyawijaya, Etsuko Ishii, Junxian He, and Pascale Fung. Belief revision: The adaptability of large language models reasoning. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 10480–10496, 2024. doi: 10.18653/v1/2024.emnlp-main.586. URL https://aclanthology.org/2024.emnlp-main.586/.

Zhenyu Wu et al. Instructing large language models to identify and ignore irrelevant conditions. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2024. doi: 10.18653/v1/2024.naacl-long.379. URL https://aclanthology.org/2024.naacl-long.379/.

Jian Xie, Kai Zhang, Jiangjie Chen, Renze Lou, and Yu Su. Adaptive chameleon or stubborn sloth: Revealing the behavior of large language models in knowledge conflicts. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/99261adc8a6356b38bcf999bba9a26dc-Abstract-Conference.html.

Haoming Xu, Weihong Xu, Zongrui Li, Mengru Wang, Yunzhi Yao, Chiyu Wu, Jin Shang, Yu Gong, and Shumin Deng. When should models change their minds? contextual belief management in large language models, 2026. URL https://arxiv.org/abs/2605.30219. Author repository reports EMNLP 2026.

Boyang Yang, Zijian Cai, Shunfu Jin, and Haoye Tian. PAFT: Preservation-aware fine-tuning for minimal-edit program repair, 2026a. URL https://arxiv.org/abs/2604.03113.

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, 2024. URL https://arxiv.org/abs/2405.15793.

Shenghao Yang, Bo Lu, Yaochen Liu, Yu Kang, Qiongfang Zhang, Chetan Bansal, Saravan Rajmohan, and Minghua Ma. DepRepair: LLM-based source-code repair for dependency breaking changes, 2026b. URL https://arxiv.org/abs/2607.17957.

Zhaocong Yuan, Adam W. Hall, Siqi Zhou, Lukas Brunke, Melissa Greeff, Jacopo Panerati, and Angela P. Schoellig. safe-control-gym: A unified benchmark suite for safe learning-based control and reinforcement learning in robotics. *IEEE Robotics and Automation Letters*, 7(4):11142–11149, 2022. doi: 10.1109/LRA.2022.3196132. URL https://arxiv.org/abs/2109.06325.

Qiusi Zhan, Zhixiang Liang, Zifan Ying, and Daniel Kang. InjecAgent: Benchmarking indirect prompt injections in tool-integrated large language model agents. In *Findings of the Association for Computational Linguistics: ACL 2024*, 2024. URL https://arxiv.org/abs/2403.02691.

Xuan Zhang et al. Ask-before-plan: Proactive language agents for real-world planning. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, 2024. doi: 10.18653/v1/2024.findings-emnlp.636. URL https://aclanthology.org/2024.findings-emnlp.636/.

Ruochen Zhao et al. Verify-and-edit: A knowledge-enhanced chain-of-thought framework. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics*, 2023. doi: 10.18653/v1/2023.acl-long.320. URL https://aclanthology.org/2023.acl-long.320/.

Zhongchao Zhou, Yuxi Lu, Yaonan Zhu, Yifan Zhao, Bin He, Liang He, Wenwen Yu, and Yusuke Iwasawa. LLMs-guided adaptive compensator: Bringing adaptivity to automatic control systems with large language models, 2025. URL https://arxiv.org/abs/2507.20509.

Wang Bill Zhu, Miaosen Chai, Shangshang Wang, Yejia Liu, Song Bian, Honghua Dong, Willie Neiswanger, and Robin Jia. Precise debugging benchmark: Is your model debugging or regenerating?, 2026. URL https://arxiv.org/abs/2604.17338.

Terry Yue Zhuo, Minh Chien Vu, Jenny Chim, Han Hu, Wenhao Yu, Ratnadira Widyasari, Imam Nur Bani Yusuf, Haolan Zhan, Junda He, Indraneil Paul, Simon Brunner, Chen Gong, Thong Hoang, Armel Randy Zebaze, Xiaoheng Hong, Wen-Ding Li, Jean Kaddour, Ming Xu, Zhihan Zhang, Prateek Yadav, Naman Jain, Alex Gu, Zhoujun Cheng, Jiawei Liu, Qian Liu, Zijian Wang, Binyuan Hui, Niklas Muennighoff, David Lo, Daniel Fried, Xiaoning Du, Harm de Vries, and Leandro Von Werra. BigCodeBench: Benchmarking code generation with diverse function calls and complex instructions. In *The Thirteenth International Conference on Learning Representations*, 2025. doi: 10.48550/arXiv.2406.15877. URL https://arxiv.org/abs/2406.15877. Oral.