ReflexBench: Evaluating Observer-Participant Failures and Counterfactual Trustworthiness in Agentic AI

Mian Zhang
Independent Researcher

Abstract

Agentic AI evaluations usually assume that the agent is an external observer of a fixed task. This assumption breaks in deployed systems: a recommendation changes user behavior, a forecast changes incentives, an intervention changes future data, and a benchmark can change the agent being benchmarked. We call this the observer-participant failure mode. We introduce ReflexBench, an evaluation protocol for measuring whether language agents reason about their own causal footprint under recursive feedback. ReflexBench organizes 20 scenarios across six domains into four observer-depth levels: fixed-world decision making, first-order self-impact, multi-agent adaptation, and recursive or equilibrium reasoning. A public-model panel of nine models and 720 scored responses shows systematic degradation at higher observer depths, with a rounded mean degradation about -0.44 from shallow to deep reflexive conditions. We extend the short benchmark formulation into a lifecycle evaluation framework for agentic AI: pre-deployment stress testing, runtime monitoring, post-deployment drift checks, and model-update regression tests. We also specify an evidence-integrity audit layer for counterfactual trustworthiness: when an agent knows it is being evaluated, or when its output changes the evidence stream, the evaluation itself becomes part of the system. ReflexBench is not a certification claim and not a replacement for domain simulation or human review. Its purpose is narrower and falsifiable: to make observer-participant failures measurable before agentic systems are trusted in feedback-sensitive environments.

Keywords

agentic evaluation, reflexive reasoning, observer-participant systems, counterfactual evaluation, benchmark integrity

1 Introduction

Most AI benchmarks evaluate agents as if they were outside the world they analyze. A math answer does not change the theorem, a code answer does not change the unit test, and a static question-answering item does not change the fact being queried. This observer-invariant assumption is useful. It is also incomplete for agentic AI.

In deployed settings, an agent's output often becomes an intervention. A trading recommendation may move the market it predicts. A public-health forecast may change compliance. A recommender may reshape the content distribution it later observes. A strategic user may adapt after learning the agent's policy. A benchmark itself may alter model behavior if models are tuned to recognize evaluation settings. In such systems, the relevant question is not only whether the agent can answer correctly in a fixed environment, but whether it can reason when its own answer changes the environment.

We call this class of failures *observer-participant failures*. They are not ordinary factual mistakes. A model may know the domain facts yet fail because it treats the environment as fixed after its own action. The failure is especially important for agentic AI evaluation, because a deployed agent is not a passive predictor. It interacts, recommends, monitors, persuades, routes, blocks, and sometimes triggers downstream actions.

This paper develops ReflexBench as a lifecycle evaluation protocol for this gap. An earlier short formulation introduced observer-depth scoring. Here we extend it into a full trustworthiness framework: scenario design, scoring schema, failure taxonomy, lifecycle monitoring, counterfactual trustworthiness, evidence-integrity logging, and explicit claim boundaries.

Contributions. The paper makes five contributions.

(1) We formalize *observer-participant evaluation* for agentic AI: tasks in which the agent's output changes the world, future observations, or the evaluation process itself.
(2) We introduce ReflexBench, a 20-scenario, six-domain protocol organized by observer depth (OD-0, OD-1, OD-2, OD-n).
(3) We report a fixed public-model audit snapshot of nine models and 720 scored responses, showing systematic degradation at deeper observer levels.
(4) We define a lifecycle monitoring wrapper for pre-deployment, runtime, post-deployment, and model-update evaluation.
(5) We connect observer-depth evaluation to counterfactual evidence integrity: the evaluation must record not only answers, but also how evidence, feedback, and benchmark exposure could change the agent or environment.

2 Problem Setting

2.1 Observer-Invariant and Observer-Participant Tasks

Let x_t denote the state of an environment and let an agent produce an output $a_t = \pi(o_{\le t})$. In an observer-invariant evaluation, the future state distribution is independent of the evaluated output:

$$P(x_{t+1} \mid x_t, a_t) = P(x_{t+1} \mid x_t). \tag{1}$$

This is a good approximation for many knowledge, coding, and math benchmarks.

In an observer-participant evaluation, the output changes the next state:

$$P(x_{t+1} \mid x_t, a_t) \neq P(x_{t+1} \mid x_t). \tag{2}$$

The output can change incentives, beliefs, user behavior, market conditions, data collection, or the agent's future evidence. A correct answer must therefore reason about the counterfactual difference between a world in which the agent does not act and a world in which the agent's output enters the system.

2.2 Observer Depth

We define observer depth as the recursive depth at which an agent models its own causal impact.

- **OD-0**: fixed-world decision making. The agent solves the immediate task while treating the environment as static.
- **OD-1**: first-order self-impact. The agent recognizes that its output changes the next state.
- **OD-2**: multi-agent adaptation. The agent models how other actors respond after observing or inferring its output.
- **OD-*n***: recursive or equilibrium reasoning. The agent reasons about repeated adaptation, fixed points, rebound effects, or non-existence of stable equilibria.

The core diagnostic is degradation:

$$\Delta_{\mathrm{OD}} = \bar{s}_{\mathrm{deep}} - \bar{s}_{\mathrm{shallow}}, \tag{3}$$

where $\bar{s}_{\mathrm{shallow}}$ averages OD-0 and OD-1 scores and $\bar{s}_{\mathrm{deep}}$ averages OD-2 and OD-*n* scores. A large negative value indicates that surface competence does not survive recursive feedback.

2.3 Why This Is a Trustworthiness Problem

Observer-participant failure is a trustworthiness issue because the agent can be locally persuasive but globally destabilizing. A static answer may look coherent while inducing a bad feedback loop after deployment. This differs from hallucination, toxicity, or prompt noncompliance. It is a failure to model the agent's causal role in the system being evaluated.

2.4 Evaluation Threat Model

Our threat model concerns evaluation reliability, not offensive security. We consider four ways in which an agentic evaluation can become unreliable.

(1) **World-shift after output.** The recommendation shifts the state distribution; a pre-output prediction can become false after adoption.
(2) **Actor adaptation.** Users, adversaries, competitors, or organizations adapt to the agent's policy after observing it.
(3) **Evidence contamination.** Logs and measurements are affected by the agent's previous actions, so the evidence stream no longer measures a neutral world.
(4) **Benchmark exposure.** The agent or model provider may tune for visible benchmark structure, producing apparent benchmark competence without deployment competence.

ReflexBench is designed to expose these risks, not to solve every downstream control problem. A reliable evaluation must make the feedback channel explicit: who observes the agent, what changes after the output, and what evidence is still uncontaminated.

Table 1: Observer-depth structure. Each level adds a new causal burden.

Level	Requirement
OD-0	Make a fixed-world decision using task facts and immediate objectives.
OD-1	Account for the first-order effect of the agent's own output or action.
OD-2	Model how other actors adapt after observing, inferring, or reacting to the agent.
OD-*n*	Reason about recursive adjustment, equilibrium pressure, rebound effects, or instability.

3 ReflexBench Protocol

3.1 Scenario Design

ReflexBench contains 20 scenarios across six domains: finance, policy, recommender systems, epidemiology and health communication, organizational decision-making, and social coordination. Each scenario is self-contained and divided into four parts corresponding to OD-0, OD-1, OD-2, and OD-*n*.

Example pattern. A market-analysis scenario first asks for a trading recommendation from a signal. The next level states that the agent's trade changes price. The next asks what happens when competitors infer the strategy. The final level asks whether a stable threshold exists once all actors adapt. Equivalent patterns appear in policy, health communication, recommender systems, and organizational decision-making.

3.2 Scoring

Each response is scored on a 0–1 rubric for each OD part. A score of 1.0 requires identifying the relevant reflexive dynamic and adjusting the recommendation accordingly. A score of 0.5 gives partial credit for naming the dynamic without using it effectively. A score of 0 indicates no meaningful observer-participant awareness.

The full audit record for each item should include scenario id, domain, OD level, model id, decoding parameters, seed or run id, full prompt, raw answer, rubric score, adjudication flag, and failure tag. These fields are part of the protocol because they separate genuine observer-depth degradation from prompt artifacts, judge instability, or incomplete provenance.

3.3 Metrics

Let $s_{m,i,d} \in [0,1]$ be the score of model m on scenario i at observer depth $d \in \{0,1,2,n\}$. ReflexBench reports four metrics:

$$S_m(d) = \frac{1}{N}\sum_{i=1}^{N} s_{m,i,d}, \tag{4}$$

$$G_m = \frac{S_m(2)+S_m(n)}{2} - \frac{S_m(0)+S_m(1)}{2}, \tag{5}$$

$$R_m = 1 - \frac{S_m(2)+S_m(n)}{S_m(0)+S_m(1)+\epsilon}, \tag{6}$$

$$A_m = \frac{1}{N}\sum_{i} \mathbb{1}\{\mathrm{failure\_tag}_{m,i} \neq \mathrm{static}\}. \tag{7}$$

Table 2: Minimum audit fields for reproducible ReflexBench runs.

Field group	Required fields
Scenario	scenario id, domain, OD level, version, prompt hash
Model	model id, provider class, date, decoding parameters
Output	raw answer, refusal flag, tool-use flag, length
Scoring	score, judge id or rubric version, adjudication flag, failure tag
Integrity	benchmark visibility, repeated exposure, evidence contamination flag

Table 3: Common observer-participant failure tags.

Failure tag	Description
Static-world answer	Gives a plausible answer but ignores that the answer changes the next state.
Audience collapse	Treats heterogeneous actors as a single non-adaptive audience.
One-step causality	Recognizes first-order impact but misses second-order adaptation.
Equilibrium blindness	Cannot reason about repeated adjustment or unstable fixed points.
Benchmark gaming risk	Optimizes for the visible evaluation rather than the target deployment behavior.
Evidence contamination	Fails to notice that measurement, feedback, or intervention changes the evidence stream.

$S_m(d)$ is the depth profile, G_m is the degradation gap, R_m is a relative reflexivity loss, and A_m is the adaptation-awareness rate. We recommend reporting G_m alongside raw scores because a high OD-0 score can otherwise hide reflexive fragility.

3.4 Failure Taxonomy

3.5 Scenario Schema

Each scenario is represented as a small structured object rather than an untracked prompt. The object contains a stable id, domain, role, base context, four OD prompts, expected reflexive burden at each level, scoring anchors, and disallowed shortcuts. The disallowed-shortcuts field is important: for example, a model should not receive full credit for merely saying "markets adapt" unless it explains how the adaptation changes the recommendation.

3.6 Scenario Families

Although the current benchmark is implemented as text scenarios, the scenarios are not arbitrary prompts. They are grouped into families that correspond to recurring deployment loops.

This grouping is useful for generalization. A new benchmark item can be created by selecting a family, specifying the agent role, then instantiating the four OD levels. The family structure also prevents overfitting to a single domain such as finance or policy.

Table 4: Scenario schema. The schema is intentionally simple so it can be ported to text-only evaluation, simulator evaluation, or human review.

Schema field	Meaning
'scenario_id'	Stable identifier for reproducibility and deduplication.
'domain'	Finance, policy, recommendation, health, organization, or social coordination.
'role'	The decision role assigned to the agent.
'od_prompts'	Four prompts corresponding to OD-0, OD-1, OD-2, and OD-n.
'burden'	The reflexive reasoning burden required for full credit.
'anchors'	Rubric anchors for 0, 0.5, and 1.0.
'failure_tags'	Permitted failure tags for audit consistency.

Table 5: Scenario families and the reflexive burden they test.

Family	Example setting	Reflexive burden
Market impact	Trading, ratings, liquidity	The agent's recommendation changes prices or risk perception.
Public signal	Policy, elections, health	A forecast changes the behavior it forecasts.
Platform feedback	Recommenders, moderation	The agent reshapes the distribution it later measures.
Organizational adaptation	Hiring, management, supply chains	Actors adapt strategically to the evaluation rule.
Coordination equilibrium	Multi-agent planning	Local recommendations create global instability or rebound.

3.7 Benchmark Integrity Stress Tests

A strong agentic evaluation should test not only model answers but also the integrity of the evaluation itself. We therefore define four stress tests that can be attached to future ReflexBench versions.

(1) **Paraphrase stability.** Rewrite surface wording while preserving the causal structure. A robust OD score should not collapse under harmless paraphrase.
(2) **Role reversal.** Ask the model to reason from the perspective of another participant who observes the agent. This tests whether OD-2 is genuine rather than a memorized phrase.
(3) **Benchmark-awareness probe.** Tell the model that it is being evaluated for reflexivity and compare with a blind prompt. A large gain suggests benchmark-gaming risk.
(4) **Delayed feedback probe.** Hide the feedback channel until the final part. This tests whether the model can infer feedback structure rather than merely follow explicit wording.

These stress tests are not used to inflate the current result. They are proposed as guardrails for subsequent benchmark versions and are included to make the evaluation harder to game.

Table 6: Lifecycle use of ReflexBench.

Stage	Evaluation question
Pre-deployment	Does the agent maintain performance from OD-0 to OD-n before release?
Runtime monitoring	Are live traces showing static-world answers, audience collapse, or evidence contamination?
Post-deployment	Did user behavior, data distribution, or incentive structure shift after the agent was deployed?
Model update	Did a new model, prompt, tool, or policy update regress observer-depth performance?
Incident review	Which failure tag explains a bad loop, and what counterfactual would have prevented it?

4 Lifecycle Evaluation for Agentic AI

The KDD agentic evaluation setting requires more than a static benchmark score. Agentic systems evolve: prompts change, tools change, API behavior changes, users adapt, and the deployment environment feeds back into the model's future evidence. ReflexBench is therefore framed as a lifecycle evaluation protocol.

This lifecycle framing matters because trustworthiness is not a one-time score. A model can pass a static benchmark and later fail when users learn its behavior. Conversely, a model can fail one OD-n setting while being acceptable in a low-feedback environment. The goal is not a universal certificate but a structured record of where feedback-sensitive deployment is risky.

4.1 Monitoring Loop

The lifecycle wrapper can be implemented as a small monitoring loop.

(1) Run a pre-deployment ReflexBench panel and store the OD profile.
(2) Attach failure tags to live traces when a user, tool, market, organization, or environment changes after an agent output.
(3) Trigger re-evaluation when the model, prompt, tool policy, memory layer, or user population changes.
(4) Compare the new OD profile to the stored baseline and flag regressions larger than a pre-registered threshold.
(5) If a regression appears, inspect raw answers and failure tags before making deployment claims.

The monitoring loop is intentionally conservative. It does not require every production trace to be scored by a human. It requires that reflexive failures have a place in the logging schema, so that post-deployment review can find them.

4.2 Deployment Examples

The following examples illustrate how the lifecycle wrapper changes the evaluation question.

Recommendation systems. A static evaluation asks whether the recommender predicts what users like. An observer-participant evaluation asks whether the recommender changes user preferences, creator incentives, or the future item distribution. A runtime ReflexBench monitor would tag traces where an initially accurate recommendation creates a later distributional collapse.

Table 7: Example regression gates for lifecycle monitoring.

Gate	Trigger
OD degradation gate	G_m worsens by more than a pre-registered tolerance.
Failure-tag gate	Static-world or equilibrium-blindness tags increase after an update.
Evidence-integrity gate	Benchmark visibility, repeated exposure, or contaminated evidence is detected.
Deployment-shift gate	Live traces show user or environment adaptation not covered by pre-deployment tests.

Forecasting agents. A static evaluation asks whether the forecast matches later observations. A reflexive evaluation asks whether the forecast caused actors to change the event being forecast. The relevant counterfactual is not simply "was the forecast correct?" but "would the event distribution have been the same without the forecast?"

Tool-using assistants. A static evaluation asks whether the assistant completes a task. A lifecycle evaluation asks whether the assistant's prior actions changed the user's next request, the tool state, or the evidence base used for evaluation. This matters for agents with memory, web access, or automated execution.

Governance. A static evaluation may check whether an agent follows a written policy. An observer-participant evaluation asks whether users adapt to it and whether the agent can identify strategic compliance, superficial alignment, or metric gaming.

5 Evidence

5.1 Public-Model Panel

The current evidence panel evaluates nine public models across 20 scenarios and four OD levels, for 720 scored responses. The compact result is that all tested models degrade at higher observer depths. The mean degradation from shallow to deep reflexive conditions is reported as a rounded audit snapshot of about -0.44. Reasoning-specialized models partially reduce the gap, especially at OD-2, but no tested model eliminates OD-n degradation.

We deliberately separate this aggregate result from a precision claim. A confidence interval, inter-rater reliability statistic, or rubric-sensitivity estimate requires the retained row-level score table, scorer identifiers, adjudication flags, and raw-output provenance. The companion artifact specification therefore exposes the scenario manifest, prompt-template structure, scoring dimensions, audit fields, and statistical hardening protocol; it does not ask the reader to infer new uncertainty estimates from an aggregate-only table. This is a claim-boundary choice rather than a weakening of the central result: the reported signal is a fixed panel snapshot, and the reproducibility path specifies exactly which rows and fields are needed for full re-computation.

5.2 Interpretation and Scope

The most important result is not that one model wins a leaderboard. The result is that observer-depth degradation is visible across the panel. A model can give good OD-0 answers and still miss that

Table 8: Evidence summary. The paper reports degradation rather than only raw score to reduce model-size confounding.

Evidence layer	Scale	Main signal
Model panel	9 models, 720 scored responses	Rounded mean OD degradation about -0.44.
Scenario set	20 scenarios, 6 domains	Surface success does not imply OD-n success.
Scoring protocol	rubric plus hard-case review	Failure tags separate static error from reflexive error.
Training trace	companion, non-general claim	Suggests trainability but is not treated as universal proof.

its own recommendation changes the system. This is exactly the failure mode agentic evaluation should expose before deployment.

The evidence is intentionally bounded. It does not prove that all models fail in all feedback-sensitive domains. It shows that a hidden axis of evaluation can be operationalized and that current public models show measurable degradation on that axis.

5.3 Negative Result

The benchmark does not show that larger or reasoning-specialized models automatically solve reflexivity. The strongest models in the panel reduce shallow errors and sometimes improve OD-2 reasoning, but the OD-n gap remains visible. This negative result is useful: it suggests that observer-participant evaluation is not redundant with ordinary first-attempt capability or chain-of-thought style reasoning.

5.4 Stop Rule

The reported panel is treated as a fixed audit snapshot: 20 scenarios, four OD levels, and nine public models. We do not tune scenario wording after seeing model-specific failures. Future versions should use held-out scenarios and publish versioned prompt hashes to avoid benchmark leakage.

5.5 Ablation and Sensitivity Plan

The current paper reports a benchmark snapshot, and the companion artifact specification defines the following hardening checks for a full row-level release.

- **Bootstrap intervals.** Resample at scenario or scenario-family level and report confidence intervals for G_m and the cross-model mean gap.
- **Scorer agreement.** Record primary scorer, secondary scorer, adjudication flag, and final-score rationale so that inter-rater agreement can be computed instead of asserted.
- **Remove self-impact wording.** If OD-1 performance remains high only when self-impact is explicit, the model may not infer feedback on its own.
- **Remove other-agent adaptation.** If OD-2 failures disappear when other actors are removed, the failure is specifically strategic rather than merely causal.
- **Domain and model-family breakdown.** If degradation is driven by one domain or one model family, the benchmark should report stratified rather than aggregate claims.
- **Judge sensitivity.** Scores should be recomputed under alternate rubric wording to measure scorer dependence.
- **Prompt-length sensitivity.** Long contexts can dilute the feedback cue; short contexts can overexpose it. Both should be recorded.

These ablations are included as a pre-registered improvement path. They also state what the present paper does not yet claim.

6 Counterfactual Trustworthiness and Evidence Integrity

An agentic evaluation can itself become part of the evaluated system. Once models are tuned on benchmark patterns, or once agents infer that they are being audited, the evaluation ceases to be a neutral observation. This creates a counterfactual trustworthiness question:

> Would the agent behave the same way if it did not know it was being evaluated, if users had not adapted to it, or if the evidence stream had not been changed by its previous outputs?

ReflexBench handles this by separating answer score from evidence integrity. A valid run should record the prompt, model, parameters, raw output, scoring rule, adjudication status, and whether the scenario is visible, repeated, or adapted. For deployment traces, the analogous record includes whether the agent's previous recommendations changed the evidence distribution.

Audit-layer bridge. The evidence-integrity layer is deliberately modular. In perception-action settings, a companion audit protocol can treat detector confidence, tracking continuity, signal integrity, map integrity, context consistency, risk level, and history as separate channels. In language-agent settings, the corresponding channels are prompt provenance, user adaptation, benchmark visibility, tool feedback, memory state, and post-output environmental change. The shared principle is that action should not be justified by a single confidence score when the evidence stream is itself affected by the agent.

6.1 From Evaluation to Action

ReflexBench does not prescribe deployment actions, but it can inform action thresholds. A low OD-0 score suggests ordinary competence failure. A high OD-0 score with a low OD-n score suggests reflexive fragility: the agent may be useful in static tasks but risky in feedback-sensitive settings. A high score across all OD levels still does not certify safety; it only lowers one class of risk. In practice, a deployment gate could require:

$$S_m(0) \geq \tau_0, \quad S_m(1) \geq \tau_1, \quad G_m \geq -\tau_G, \quad I_{\text{eval}} \geq \tau_I, \tag{8}$$

where I_{eval} is an evidence-integrity score over benchmark provenance, prompt visibility, model version, and scoring auditability. This separates competence, reflexivity, and evaluation integrity rather than collapsing them into a single leaderboard number.

Table 9: Evidence-integrity mapping from perception-action systems to language-agent evaluation.

Perception-action channel	Agentic-evaluation analogue
Sensor quality	Prompt/context provenance and completeness.
Tracking continuity	Conversation, memory, and tool-state continuity.
Signal or time sync	API version, timestamp, model update, and latency integrity.
Map or prior integrity	Task prior, benchmark version, and hidden-test separation.
Risk constraints	Deployment domain, user harm, and allowed action boundary.
History	Prior exposure, adaptation, and feedback-loop logs.

Table 10: Controls and boundaries.

Concern	Control or boundary
Subjective scoring	Use OD-specific rubrics, raw-answer retention, adjudication flags, and failure tags.
Model-size confound	Report degradation across OD levels, not only raw aggregate score.
Prompt sensitivity	Use uniform scenario structure and retain prompt provenance.
Benchmark gaming	Treat benchmark visibility and repeated exposure as evidence-integrity fields.
Training overclaim	Treat the companion training trace as support, not proof of universal phase transition.
Deployment misuse	Present ReflexBench as diagnostic, not as certification.

7 Relation to Existing Evaluation Work

Static benchmarks. Knowledge, math, coding, and instruction benchmarks are necessary, but most evaluate observer-invariant competence. ReflexBench adds a complementary axis: whether competence survives after the agent becomes part of the causal loop.

Agent benchmarks. Agent benchmarks often evaluate tool use, planning, web interaction, or task completion. For example, τ-bench evaluates language agents in tool-agent-user interactions with domain rules and simulated users [10]. MALIBU evaluates bias in multi-agent LLM settings with persona-based scenario assessments and judging [11]. ReflexBench differs by making feedback sensitivity itself the object of evaluation. A tool-using or multi-agent system may complete tasks, follow rules, or expose bias while still failing to model the strategic or distributional consequences of its own outputs.

Causal and counterfactual evaluation. The protocol is aligned with causal evaluation because each OD level compares worlds with and without the agent's output. However, ReflexBench is not a full causal discovery method. It is a practical benchmark for whether a model notices the relevant counterfactual burden.

Performative prediction and strategic adaptation. Performative prediction and strategic classification study settings where deployment changes the data distribution. ReflexBench translates that theoretical concern into a compact language-agent evaluation protocol, adding multi-domain scenarios and an auditable scoring schema.

8 Controls and Claim Boundaries

The benchmark is not a claim about consciousness, sentience, or general intelligence. It evaluates an operational capability: whether an agent recognizes that its output can change the environment it is analyzing. The score is useful because many deployment failures have exactly this structure.

8.1 Scope Boundary

ReflexBench is distinct from three adjacent problem classes. First, it is not a longitudinal wisdom-acquisition benchmark: it does not ask whether an agent improves after repeated exposure, but whether it recognizes its own causal footprint. Second, it is not a robust-perception benchmark: evidence integrity is an audit principle for language-agent evaluation, not a detector benchmark. Third, it is not only a static benchmark note: the present framework adds lifecycle monitoring, evaluation threat modeling, integrity fields, regression gates, and deployment boundaries.

8.2 Audit Checklist

For review and reproduction, the key questions are:

- Are OD levels separable and auditable?
- Are raw prompts, raw answers, scoring fields, and failure tags retained?
- Does the paper avoid certification claims?
- Does the method handle benchmark exposure and feedback contamination?
- Is degradation reported instead of only raw leaderboard rank?

If any of these fail, the evaluation should be considered incomplete. This is deliberately stricter than a normal static benchmark record because the evaluated object can change after being observed.

9 Reproducibility and Release Plan

The prepared companion audit package includes a scenario schema, scenario manifest, prompt-template structure, failure-tag ontology, scoring rubric, audit-field specification, and a minimal result-table skeleton. The skeleton verifies field names and computation shape; it is not a substitute for the retained row-level raw responses and adjudication records required for independent recomputation of confidence intervals or scorer agreement. The package avoids private logs and non-essential identity-bearing metadata. A public release should use a versioned record that separates benchmark updates from model updates.

Table 11: Companion audit package contents.

Artifact	Role
Scenario schema	Defines stable scenario and OD-level fields.
Scenario manifest	Lists the 20 sanitized scenario families and expected reflexive burdens.
Failure tags	Standardizes observer-participant failure annotation.
Audit fields	Records model, prompt, scoring, and integrity provenance.
Scoring rubric	Provides OD-specific scoring anchors.
Sample results	Verifies table format and aggregate computation fields.

The release plan follows the same principle as the paper: enough material for review and auditability, without overclaiming production deployment. Full public release should include versioned prompts, raw outputs, scoring scripts, bootstrap scripts, adjudication logs, and a changelog separating benchmark updates from model updates.

10 Limitations

ReflexBench is language-only and stylized. It does not replace domain-specific simulation, real-world causal evaluation, human expert review, or regulated safety assessment. Scoring OD-2 and OD-n involves judgment; rubrics and adjudication reduce arbitrariness but do not remove it. The prepared companion artifact is a protocol and audit package rather than a complete raw-output release, so confidence intervals, inter-rater reliability, and rubric-sensitivity estimates must be recomputed from the retained row-level score table before being treated as statistical claims. The current scenario distribution is not balanced across all possible domains, and finance and policy-style cases are easier to express in text than some physical feedback loops. The public-model panel is a snapshot, not a permanent ranking. Models may improve after targeted training or exposure.

The evidence-integrity layer is also a protocol, not a complete field deployment. A full deployment study would require live traces, user adaptation logs, tool-use records, and external validation. We therefore avoid certification language. A system that scores well on ReflexBench should be considered less blind to observer-participant effects, not automatically safe.

11 Conclusion

Agentic AI needs evaluation methods for worlds that change because the agent acts. ReflexBench introduces observer-depth scoring for this purpose and extends it into a lifecycle trustworthiness framework. In a fixed audit snapshot across 20 scenarios, six domains, nine public models, and 720 scored responses, current models show systematic degradation under deeper reflexive pressure. The benchmark's value is not a static leaderboard; it is a way to expose a hidden failure mode before feedback-sensitive agents are trusted in deployment.

References

[1] C. A. E. Goodhart. Problems of monetary management: The UK experience. 1975.
[2] D. T. Campbell. Assessing the impact of planned social change. 1979.
[3] G. Soros. The Alchemy of Finance. 1987.
[4] M. Hardt, N. Megiddo, C. Papadimitriou, and M. Wootters. Strategic classification. 2016.
[5] J. Perdomo, T. Zrnic, C. Mendler-Duenner, and M. Hardt. Performative prediction. 2020.
[6] J. Pearl. Causality: Models, Reasoning, and Inference. 2009.
[7] S. Russell. Human Compatible: Artificial Intelligence and the Problem of Control. 2019.
[8] P. Liang et al. Holistic evaluation of language models. 2023.
[9] D. Hendrycks et al. Measuring massive multitask language understanding. 2021.
[10] S. Yao, N. Shinn, P. Razavi, and K. Narasimhan. τ-bench: A benchmark for tool-agent-user interaction in real-world domains. 2024.
[11] I. Mirza, C. Huang, I. Vasista, R. Patil, A. Akalin, S. O'Brien, and K. Zhu. MALIBU benchmark: Multi-agent LLM implicit bias uncovered. 2025.