# When Should Models Change Their Minds? Contextual Belief Management in Large Language Models

**Anonymous ACL submission**


## Abstract

Long-horizon interactions require language models to manage accumulating information: when to update their state, when to preserve their state, and what to ignore. We study this challenge as **Contextual Belief Management (CBM)**: maintaining a predicted belief state aligned with formal evidence while isolating task-irrelevant noise. To make CBM measurable, we introduce BeliefTrack, a closed-world benchmark spanning Rule Discovery and Circuit Diagnosis, where a finite belief space and symbolic verifiers enable exact turn-level evaluation. BeliefTrack diagnoses three failures: Failed Stay, Failed Update, and Failed Isolation. Across multiple LLMs, vanilla models exhibit severe CBM failures, while explicit belief-tracking prompts provide limited gains. In contrast, reinforcement learning with belief-state rewards reduces failure rates by 70.9% on average. Further probing reveals latent belief-state dynamics behind these failures, and representation-level steering reduces failure rates by 46.1% across two tasks.


## 1 Introduction

Large language models (LLMs) are increasingly deployed in long-horizon interactions, where their behavior depends not only on parametric knowledge but also on context, memory, tools, and runtime protocols (Yang et al., 2024; Zhou et al., 2023; Wu et al., 2024; Lee et al., 2026). In such settings, models must manage beliefs as different types of information accumulate over time. Some information should revise the model's current belief state, some should leave it unchanged, and some should be ignored altogether. Recent work on context learning, such as CL-Bench (Dou et al., 2026), studies whether models can absorb rules, knowledge, or procedures from context and translate them into effective behavior. However, absorbing contextual information is not enough: a model must also decide which information counts as formal evidence, when that evidence warrants belief revision, and when task-irrelevant context should be filtered out.

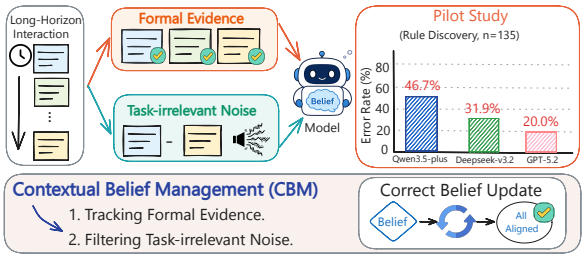


Figure 1: Overview of *Contextual Belief Management* (CBM). CBM requires models to maintain a predicted belief state over a belief space, update it only when warranted by formal evidence, and filter task-irrelevant context or noise. The pilot Rule Discovery study reveals substantial belief-management errors in frontier models.

As Figure 1 shows, we study this problem as **Contextual Belief Management** (CBM): a model's ability to maintain an evidence-aligned belief state throughout a multi-turn interaction. Rather than simulating open-ended dialogue, we operationalize CBM in a controlled closed-world setting. Specifically, we introduce **BeliefTrack**, a closed-world benchmark with two environments: Rule Discovery (RD) and Circuit Diagnosis (CD). Both environments define finite belief spaces and use symbolic verifiers, allowing exact turn-level comparison between predicted and oracle belief states. This design abstracts away open-ended ambiguity and allows us to evaluate distinct belief-management operations precisely.

As a pilot study, we evaluate Qwen3.5-Plus (Team, 2026), DeepSeek-V3.2 (DeepSeek-AI et al., 2025), and GPT-5.2 (Singh et al., 2026) on 135 Rule Discovery examples with task-irrelevant noise. As shown in Figure 1, all three frontier models exhibit substantial belief-management errors. These results suggest that CBM failures arise even when the relevant evidence is explicitly specified.

Understanding these failures requires more than checking whether the model produces the correct

belief state at a single turn. A model must preserve a stable belief when formal evidence remains unchanged, revise its belief when formal evidence changes, and isolate its belief state from task-irrelevant context. To localize these errors, BeliefTrack evaluates three diagnostic failures: *Failed Stay*, *Failed Update*, and *Failed Isolation*. These diagnostics distinguish belief calibration failures from belief isolation failures.

With these concepts in place, §3 defines Contextual Belief Management (CBM) and details how BeliefTrack operationalizes it in Rule Discovery and Circuit Diagnosis with exact symbolic verification. In §5, we evaluate current LLMs and find that **vanilla models exhibit severe CBM failures**, while explicit belief-tracking prompts provide only limited and inconsistent gains. We further show that **reinforcement learning with belief-state rewards** substantially reduces failure rates, transfers across environments, and improves robustness to task-irrelevant noise. In §6, prompt-based probing reveals **latent belief-state dynamics** behind these failures, including belief-state drift, backtracking failure, and contextual hijacking. Finally, **representation-level steering** directly improves predicted-oracle belief-state alignment, suggesting that CBM failures are not only measurable but also **actionable at the representation level**.

## 2 Related Work

**Knowledge Conflict.** Deciding which information to trust is central to belief management in language models. Prior work shows that models struggle to resolve conflicts between parametric memory and context from passages, user claims, demonstrations (Longpre et al., 2021; Wang et al., 2024; Xu et al., 2024c; Kortukov et al., 2024; Jin et al., 2024; Xie et al., 2024; Xu et al., 2024d; Hagström et al., 2026). Recent work further highlights belief dependencies in conflict resolution, where updating one fact can affect others (Yao et al., 2025; Xu et al., 2026). By contrast, CBM does not introduce direct information conflicts, but tests whether models update beliefs only from formal evidence.

**Multi-turn Reasoning Instability.** LLMs often become unreliable in long interactions: they lose relevant evidence (Liu et al., 2024; Zhang et al., 2025; Al-Tawaha et al., 2026), degrade in multi-turn instruction following (Laban et al., 2026; Duan et al., 2025), and fail under contextual pressure (Xu et al., 2024b; Deng et al., 2026). Recent work further identifies *contextual inertia*, where models fail to revise earlier generations or intermediate inferences despite later contradictory evidence (Huang et al., 2026; Chen et al., 2026a; Liu et al., 2025), as well as mechanisms involving metacognition, memory management, and epistemic state tracking (Raj, 2026; Yona et al., 2026; Chen et al., 2026b; Yalon et al., 2026). CBM turns these instabilities into exact turn-level diagnostics, separating failures of belief calibration and isolation.

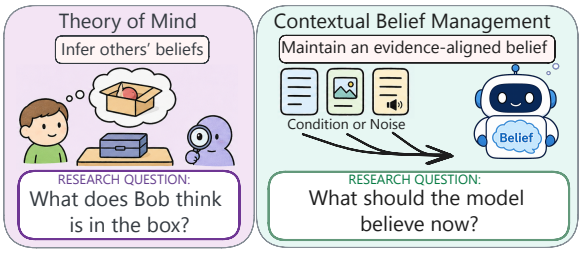


Figure 2: Comparison between *Contextual Belief Management* and *Theory of Mind*.

**Belief Tracking and Theory of Mind.** Recent work studies belief dynamics in LLMs, from revising prior reasoning (Wilie et al., 2024) to maintaining temporal belief consistency in long-running agents (Myakala et al., 2026) and constructing spatial beliefs through active exploration (Zhang et al., 2026). These studies highlight the challenge of maintaining stable yet revisable beliefs over time. In Theory of Mind (ToM), belief tracking instead targets hidden mental states of other agents, such as beliefs, desires, intentions, and perspectives (Ullman, 2023; Kim et al., 2023; Chen et al., 2024; Strachan et al., 2024a,b; Kosinski, 2024; Shapira et al., 2024; Xu et al., 2024a; Cross et al., 2025; Prakash et al., 2026; Shi et al., 2025). As illustrated in Figure 2, ToM is a third-person inference problem, whereas CBM asks what the model itself should believe from accumulated formal evidence. We evaluate this first-person problem in closed-world environments with finite belief spaces and symbolic verifiers.

## 3 Preliminary

### 3.1 Problem Formulation

We formalize **Contextual Belief Management** (CBM) as a model's ability to maintain an evidence-aligned belief state throughout a multi-turn interaction. At each turn $t \in \{1, \ldots, T\}$ in an environment $\mathcal{E}$, a belief-tracking model $f_\theta$ receives an observation $o_t = (e_t, n_t)$, where $e_t$ denotes formal evidence and $n_t$ denotes optional task-irrelevant noise

($n_t = \varnothing$ in clean settings). Let $e_{1:t} = (e_1, \ldots, e_t)$ and $o_{1:t} = (o_1, \ldots, o_t)$ denote the formal-evidence history and the observation history, respectively.

Let $\mathcal{B}_\mathcal{E}$ denote the task-specific belief space of environment $\mathcal{E}$. Its elements are candidate hypotheses representing all possible task outcomes. A belief state is a subset of this space: the candidate hypotheses that remain supported by the formal evidence observed so far. We define two belief states at each turn. The **Oracle Belief State** $S_t^* \subseteq \mathcal{B}_\mathcal{E}$ is the logically correct subset determined by the formal-evidence history $e_{1:t}$. The **Predicted Belief State** $\hat{S}_t = f_\theta(o_{1:t}) \subseteq \mathcal{B}_\mathcal{E}$ is the subset produced by the model from the observation history $o_{1:t}$.

**Contextual Belief Management**

The objective of CBM is to maintain turn-level alignment between the model's predicted belief state and the oracle belief state over the full trajectory $\tau$:

$$\max_\theta \mathbb{E}_{\tau\sim\mathcal{E}} \left[ \frac{1}{T} \sum_{t=1}^{T} \mathbb{I}\{\hat{S}_t = S_t^*\} \right], \qquad (1)$$

where $\mathbb{I}$ is the indicator function.

We distinguish two classes of CBM failures.

**Type I: Belief Calibration Failures.** Belief calibration failures arise when the model fails to track the evidence-aligned belief state under clean conditions ($n_t = \varnothing$). They manifest in two forms:

1. Failed Stay: The oracle belief state remains unchanged ($S_t^* = S_{t-1}^*$), but the model fails to preserve this stable state, producing $\hat{S}_t \neq S_t^*$.

2. Failed Update: The oracle belief state changes ($S_t^* \neq S_{t-1}^*$), but the model fails to transition to the revised state, producing $\hat{S}_t \neq S_t^*$.

**Type II: Belief Isolation Failures.** Belief isolation failures occur when the model fails to separate formal evidence from task-irrelevant noise. Specifically, consider a turn where the model correctly predicts the oracle belief state under the clean formal-evidence history. If adding task-irrelevant noise $n_t$ changes the prediction and leads to $\hat{S}_t \neq S_t^*$, the model has incorrectly treated non-evidential noise as part of the formal reasoning signal.

### 3.2 Closed-World Task Environments

Evaluating CBM requires isolating formal-evidence tracking from reliance on pre-trained world knowledge. In standard multi-turn question-answering datasets, errors may stem from factual hallucination rather than belief-management failure, and the correct belief state is often not exactly computable at each turn.

We therefore introduce **BeliefTrack**, a closed-world benchmark for evaluating turn-level belief dynamics. As shown in Figure 3, BeliefTrack formulates each task as evidence-conditioned belief-state tracking over a finite belief space $\mathcal{B}_\mathcal{E}$. All task-relevant evidence is specified within the episode, and the model must output the predicted belief state $\hat{S}_t \subseteq \mathcal{B}_\mathcal{E}$: the subset of candidate hypotheses aligned with the accumulated formal evidence.

Figure 3 illustrates the two environments used in BeliefTrack. Both instantiate the same CBM formulation but differ in the semantics of their candidate hypotheses and formal evidence.

**Task A: Rule Discovery (RD).** Adapted from Wason's 2-4-6 paradigm, Rule Discovery defines a finite belief space $\mathcal{B}_{\text{RD}} = \{h_1, \ldots, h_M\}$, where each candidate hypothesis is a possible rule, such as `ascending_order` or `sum_greater_than_10`. At each turn $t$, the formal evidence $e_t$ consists of a proposed triple, such as `[3, 8, 1]`, and its ground-truth label, `YES` or `NO`, determined by a hidden oracle rule. The oracle belief state is the subset of candidate rules that remain consistent with the accumulated triple-label evidence.

**Task B: Circuit Diagnosis (CD).** Circuit Diagnosis evaluates diagnostic reasoning in a circuit-fault setting. It defines a finite belief space $\mathcal{B}_{\text{CD}} = \{h_1, \ldots, h_M\}$, where each candidate hypothesis is a possible circuit fault, such as `Battery_no_output` or `R1_open`. At each turn $t$, the formal evidence $e_t$ is an instrument reading, such as `Current(Main)>0` or `Voltage(R1)=0`. The oracle belief state is the subset of candidate faults whose predicted circuit behavior remains consistent with the accumulated readings.

### 3.3 Dataset Generation and Verification

As illustrated in Figure 3, we generate three diagnostic datasets from BeliefTrack: $\mathcal{D}^{\text{stay}}$, $\mathcal{D}^{\text{update}}$, and $\mathcal{D}^{\text{iso}}$, targeting Failed Stay, Failed Update, and Failed Isolation, respectively. Each dataset consists of fixed user-side multi-turn diagnostic templates paired with a symbolic verifier that computes the oracle belief state $S_t^*$ at each turn. Because each environment has a finite belief space and fully specified verification logic, automatic evaluation requires no human annotation. Assistant-side trajectories are sampled during evaluation.

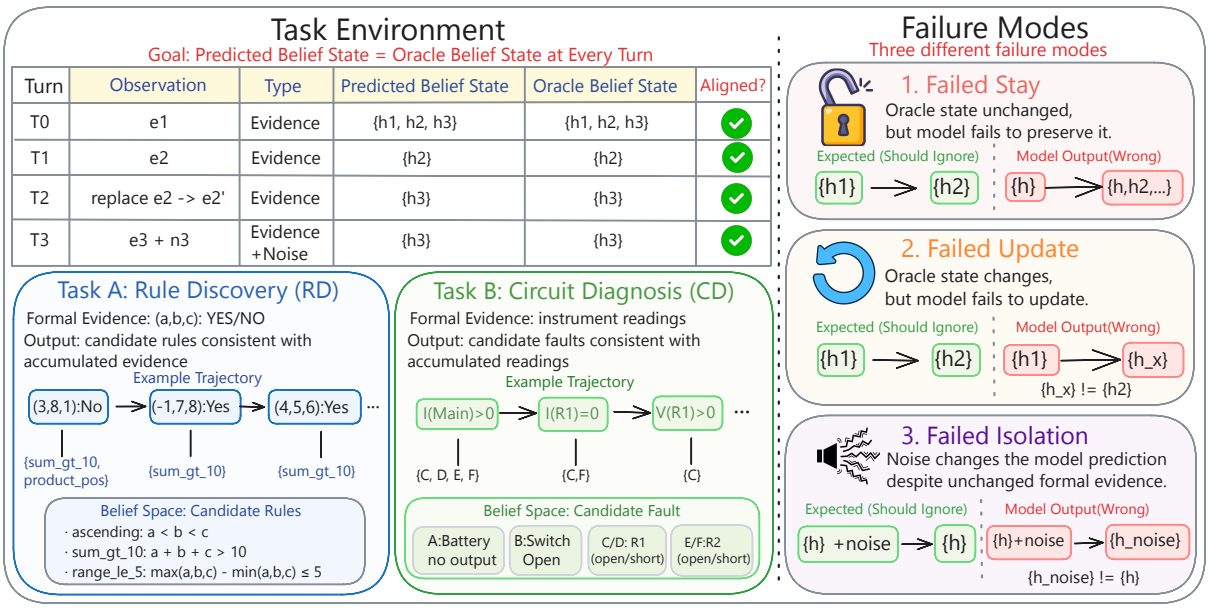


Figure 3: BeliefTrack framework. Given a finite belief space, the model must output a predicted belief state, i.e., a subset of candidate hypotheses that it considers supported by the accumulated formal evidence. The goal is to align this predicted belief state with the oracle belief state at each turn. BeliefTrack includes two closed-world environments, Rule Discovery (RD) and Circuit Diagnosis (CD), and evaluates three typical diagnostic failure modes: *Failed Stay*, *Failed Update*, and *Failed Isolation*.

**1. Dataset for Failed Stay.** $\mathcal{D}^{\text{stay}}$ tests whether models preserve an oracle belief state. Each template contains a lock point $t_{\text{lock}}$, before which evidence narrows the oracle state to a target subset $S_{\text{lock}} \subseteq B_E$. Afterward, only redundant evidence is added, so the oracle state remains fixed:

$$S_t^* = S_{t_{\text{lock}}}^* = S_{\text{lock}}, \quad \forall t > t_{\text{lock}}. \tag{2}$$

Errors at these turns correspond to **Failed Stay**.

**2. Dataset for Failed Update.** $\mathcal{D}^{\text{update}}$ tests whether models revise their belief state after earlier evidence is corrected. Each template includes a to-be-corrected evidence item $e_j$, followed by a `CORRECTION` at turn $t_c$ that replaces it with corrected evidence $e'_j$. The verifier recomputes the oracle state under the corrected evidence history, and we retain correction turns where

$$S_{t_c}^* \neq S_{t_c-1}^*. \tag{3}$$

Errors at these turns correspond to **Failed Update**.

**3. Dataset for Failed Isolation.** $\mathcal{D}_{\text{iso}}$ tests whether models ignore task-irrelevant noise. Each template forms a clean–noised pair, $o_{1:t}^{\text{clean}} = (e_{1:t}, \varnothing_{1:t})$ and $o_{1:t}^{\text{noise}} = (e_{1:t}, n_{1:t})$, that shares the same evidence history $e_{1:t}$ but differs in the noise history. Thus, the oracle belief state is unchanged:

$$S_t^{*,\text{clean}} = S_t^{*,\text{noise}}, \quad \forall t. \tag{4}$$

**Failed Isolation** occurs when the model succeeds on the clean trajectory but fails on the noisy one.

The concrete dataset templates are provided in Appendix D.1. Since the environments are symbolic and automatically verifiable, they support scalable trajectory generation. In this work, we instantiate 1,300/1,049 Rule Discovery/Circuit Diagnosis trajectories for instruct models, and 1,503/1,616 trajectories for thinking models.

## 4 Methods for Improving CBM

We evaluate two methods for improving CBM: *Belief-Tracking Prompt* (*BT-Prompt*), a training-free prompt-based enhancement method, and *RL with belief-state rewards*, a verifier-guided reinforcement-learning method.

### 4.1 BT-Prompt

*BT-Prompt* is a parameter-free test-time baseline that encodes the CBM procedure in the system prompt. It instructs the model to maintain the current set of valid formal evidence, ignore non-evidential noise, re-evaluate all candidate hypotheses against the accumulated evidence, and revise the evidence set when explicit corrections invalidate earlier observations. This allows previously eliminated hypotheses to be restored when the evidence excluding them is removed. BT-Prompt is applied uniformly across both environments and all diagnostic trajectory types; full templates are provided in Appendix D.2.

### 4.2 RL with Belief-State Rewards

We further optimize the model with GRPO (Shao et al., 2024) using rewards computed by a symbolic verifier. Training examples are extracted from multi-turn BeliefTrack trajectories, but each GRPO prompt targets a single evaluated turn. Specifically, for a target turn $t$, we construct a prompt context $q_t = \mathrm{Prompt}(\mathcal{B}_{\mathcal{E}}, o_{1:t})$ from the belief space and the full observation history up to that turn. The model then produces one response $y_i$ containing a predicted belief state $\hat{S}_{i,t}$ for the target turn. The verifier compares $\hat{S}_{i,t}$ with the oracle belief state $S_t^*$ and assigns a reward only for that turn. For each prompt context $q_t$, GRPO samples a group of outputs $\{y_1, \ldots, y_G\}$ and optimizes the standard clipped objective:

$$\mathcal{J}_{\text{GRPO}}(\theta) = \mathbb{E}_{q_t, \{y_i\}_{i=1}^{G}} \Bigg[ \frac{1}{G} \sum_{i=1}^{G} \Bigg\{ \min \Bigg( \rho_i(\theta) A_i, \\ \text{clip}\Big(\rho_i(\theta), 1-\epsilon, 1+\epsilon\Big) A_i \Bigg) - \beta \mathrm{D}_{\mathrm{KL}}\Big(\pi_\theta \,\|\, \pi_{\text{ref}}\Big) \Bigg\} \Bigg], \tag{5}$$

where $\rho_i(\theta) = \pi_\theta(y_i|q_t)/\pi_{\theta_{\text{old}}}(y_i|q_t)$ and $A_i = (R_i - \mathrm{mean}(\{R_i\}_{i=1}^G))/\mathrm{std}(\{R_i\}_{i=1}^G)$ is the group-normalized reward advantage.

We use a dense Jaccard belief-state reward:

$$R_i(q_t) = \frac{|\hat{S}_{i,t} \cap S_t^*|}{|\hat{S}_{i,t} \cup S_t^*|}. \tag{6}$$

Since $S_t^*$ is non-empty by construction, the denominator is always non-zero. This reward measures set-level alignment between the predicted and oracle belief states at the target turn. Unlike sparse exact match, it gives partial credit to predictions that overlap with the oracle state; we ablate this design in Appendix C.1.

## 5 Experiments

### 5.1 Metrics

Based on the diagnostic datasets defined in Section 3.3, we use a strict $k$-repeat evaluation protocol. For each diagnostic sample $x$, the user-side multi-turn template is fixed, and we independently sample $k$ assistant-side trajectories. Let $E_m^{(i)}(x) \in \{0, 1\}$ indicate whether the $i$-th trajectory exhibits the target failure mode $m \in \{\text{stay}, \text{update}, \text{iso}\}$. Here, stay, update, and iso correspond to Failed Stay, Failed Update, and Failed Isolation, respectively. We define sample-level failure as

$$F_m^{(k)}(x) = \mathbb{I}\left[\exists i \in \{1, \ldots, k\},\ E_m^{(i)}(x) = 1\right]. \tag{7}$$

Thus, a sample fails if any repeated trajectory exhibits the target failure.

**Failed Stay Rate (FSR).** For $x \in \mathcal{D}^{\text{stay}}$, $E_{\text{stay}}^{(i)}(x) = 1$ if the $i$-th trajectory makes a Failed Stay error on the evaluated post-lock turns.

$$\text{FSR} = \frac{1}{|\mathcal{D}^{\text{stay}}|} \sum_{x \in \mathcal{D}^{\text{stay}}} F_{\text{stay}}^{(k)}(x). \tag{8}$$

**Failed Update Rate (FUR).** For $x \in \mathcal{D}^{\text{update}}$, $E_{\text{update}}^{(i)}(x) = 1$ if the $i$-th trajectory makes a Failed Update error at the correction turn.

$$\text{FUR} = \frac{1}{|\mathcal{D}^{\text{update}}|} \sum_{x \in \mathcal{D}^{\text{update}}} F_{\text{update}}^{(k)}(x). \tag{9}$$

**Failed Isolation Rate (FIR).** For $x \in \mathcal{D}^{\text{iso}}$, $E_{\text{iso}}^{(i)}(x) = 1$ if the $i$-th trajectory succeeds on the clean trajectory but fails on the noised trajectory.

$$\text{FIR} = \frac{1}{|\mathcal{D}^{\text{iso}}|} \sum_{x \in \mathcal{D}^{\text{iso}}} F_{\text{iso}}^{(k)}(x). \tag{10}$$

All experiments use $k = 3$ and lower values indicate better performance.

### 5.2 Implementation Details

We split each environment at the oracle level before trajectory generation. In RD, train and test trajectories are generated from disjoint rule sets; in CD, they are also generated from disjoint fault sets. This prevents oracle-specific memorization and evaluates generalization to unseen evidence-conditioned belief states.

For each base model, we train two single-environment RL variants: RL-RD, trained only on the RD training split, and RL-CD, trained only on the CD training split. RL training uses target-turn prompts from $\mathcal{D}_{\text{stay}}$ and $\mathcal{D}_{\text{update}}$: post-lock turns for belief preservation and correction turns for belief revision. We exclude $\mathcal{D}_{\text{iso}}$ from training, so RL never observes task-irrelevant noise trajectories.

Both RL variants are evaluated on both held-out RD and held-out CD test sets. Evaluation on the training environment measures in-domain performance, while evaluation on the other environment measures cross-environment generalization. Vanilla and BT-Prompt require no training and are evaluated on the same held-out test sets across all three diagnostic trajectory types. Detailed split statistics are provided in Appendix B.1.

We evaluate Qwen2.5-7B-Instruct (Qwen et al., 2025) and Qwen3.5-9B (Team, 2026). Training and evaluation hyperparameters, decoding settings, and infrastructure are provided in Appendix B.2.

| Method | Rule Discovery (RD) | | | Circuit Diagnosis (CD) | | | General | |
|---|---|---|---|---|---|---|---|---|
| | FSR ↓ | FUR ↓ | FIR ↓ | FSR ↓ | FUR ↓ | FIR ↓ | GSM8K ↑ | MMLU ↑ |
| **Qwen2.5-7B-Instruct** | | | | | | | | |
| Vanilla | 99.0 | 98.0 | 97.0 | 99.0 | 98.0 | 97.0 | $83.3 \pm 0.4$ | $73.5 \pm 0.3$ |
| BT-Prompt | $79.0_{\downarrow 20.2\%}$ | $80.0_{\downarrow 18.4\%}$ | $95.0_{\downarrow 2.1\%}$ | $48.0_{\downarrow 51.5\%}$ | $87.9_{\downarrow 10.3\%}$ | $92.0_{\downarrow 5.2\%}$ | — | — |
| RL-RD | $\mathbf{0.0}_{\downarrow 100.0\%}$ | $\mathbf{2.0}_{\downarrow 98.0\%}$ | $\mathbf{20.0}_{\downarrow 79.4\%}$ | $\underline{6.0}_{\downarrow 93.9\%}$ | $\underline{28.3}_{\downarrow 71.1\%}$ | $\underline{35.0}_{\downarrow 63.9\%}$ | $84.8 \pm 0.5$ | $73.1 \pm 0.2$ |
| RL-CD | $\underline{42.0}_{\downarrow 57.6\%}$ | $\underline{74.0}_{\downarrow 24.5\%}$ | $\underline{57.0}_{\downarrow 41.2\%}$ | $\mathbf{0.0}_{\downarrow 100.0\%}$ | $\mathbf{0.0}_{\downarrow 100.0\%}$ | $\mathbf{20.0}_{\downarrow 79.4\%}$ | $83.9 \pm 0.3$ | $73.4 \pm 0.4$ |
| **Qwen3.5-9B** | | | | | | | | |
| Vanilla | 47.0 | 60.0 | 83.7 | 43.2 | 62.7 | 95.4 | $94.0 \pm 0.2$ | $78.1 \pm 0.3$ |
| BT-Prompt | $49.0_{\uparrow 4.3\%}$ | $69.0_{\uparrow 15.0\%}$ | $81.4_{\downarrow 2.7\%}$ | $47.4_{\uparrow 9.7\%}$ | $63.5_{\uparrow 1.3\%}$ | $81.4_{\downarrow 14.7\%}$ | — | — |
| RL-RD | $\mathbf{6.0}_{\downarrow 87.2\%}$ | $\mathbf{8.0}_{\downarrow 86.7\%}$ | $\mathbf{18.6}_{\downarrow 77.8\%}$ | $\underline{20.0}_{\downarrow 53.7\%}$ | $\underline{21.4}_{\downarrow 65.9\%}$ | $\underline{34.9}_{\downarrow 63.4\%}$ | $94.5 \pm 0.6$ | $78.2 \pm 0.3$ |
| RL-CD | $\underline{31.0}_{\downarrow 34.0\%}$ | $\underline{34.0}_{\downarrow 43.3\%}$ | $\underline{41.9}_{\downarrow 49.9\%}$ | $\mathbf{12.1}_{\downarrow 72.0\%}$ | $\mathbf{15.9}_{\downarrow 74.6\%}$ | $\mathbf{16.3}_{\downarrow 82.9\%}$ | $94.5 \pm 0.6$ | $78.7 \pm 0.3$ |

Table 1: Main diagnostic and cross-task generalization results. We report Failed Stay Rate (FSR), Failed Update Rate (FUR), and Failed Isolation Rate (FIR) (lower is better ↓), together with MMLU and GSM8K accuracy (higher is better ↑). **Bold** and underline denote the **best** and second-best results respectively. **RL-RD** and **RL-CD** indicate models trained solely on Rule Discovery or Circuit Diagnosis. Blue and orange indicate in-domain (ID) and out-of-domain (OOD) generalization, respectively. Subscripts show relative changes from Vanilla.

### 5.3 Main Results

Table 1 reports diagnostic failure rates and general capability scores. We highlight three findings.

**Vanilla models consistently lack reliable CBM.** Qwen2.5-7B-Instruct fails almost completely across both environments, with failure rates around 97–99% on all three metrics. Qwen3.5-9B is stronger but still exhibits substantial CBM failures, especially on the Failed Isolation split $\mathcal{D}_{\text{iso}}$: its FIR reaches 95.4% in Circuit Diagnosis.

**BT-Prompt provides limited gains.** Although BT-Prompt improves some metrics, its effects vary across models and environments. In several cases, it even degrades performance, increasing Qwen3.5-9B's FUR in RD by 15.0% and FSR in CD by 9.7%. This suggests that explicit test-time instructions alone are insufficient for reliable CBM.

**RL with belief-state rewards consistently improves CBM.** RL yields strong in-domain gains across both models and environments. For Qwen2.5-7B, RD training reduces in-domain FSR/FUR to 0.0%/2.0%, while CD training reduces CD FSR/FUR to 0.0%/0.0%. For Qwen3.5-9B, RL lowers in-domain FSR/FUR to 6.0%/8.0% on RD and 12.1%/15.9% on CD.

RL also generalizes beyond the training environment. **First, it transfers across tasks**: RD-trained Qwen2.5-7B reduces unseen CD FSR and FUR by 93.9% and 71.1%, respectively. For Qwen3.5-9B, RD-trained RL reduces CD FSR/FUR by 53.7%/65.9%, while CD-trained RL reduces RD FSR/FUR by 34.0%/43.3%. **Second, RL improves belief isolation even though $\mathcal{D}_{\text{iso}}$ is excluded from training**. RD-trained Qwen2.5-7B reduces FIR by 79.4% in-domain and 63.9% out-of-domain. Similarly, RD-trained Qwen3.5-9B reduces FIR by 77.8% in-domain and 63.4% out-of-domain, while CD-trained Qwen3.5-9B achieves the strongest CD FIR reduction of 82.9%.

Together, these results suggest that verifier-guided RL improves general belief-state management rather than merely fitting task-specific patterns. The gains do not come at the expense of general ability: MMLU (Hendrycks et al., 2021) and GSM8K (Cobbe et al., 2021) remain largely stable after training, with only small fluctuations. Finally, Figure 6 shows that most CBM gains emerge early in training, while later checkpoints fluctuate across metrics and transfer settings.

## 6 Robustness and Mechanistic Analysis

We analyze CBM along three axes: temporal robustness, contextual robustness, and mechanism. We first vary anchoring depth to test whether models can preserve and revise belief states over long evidence histories. We then vary task-irrelevant noise to test whether models can isolate formal evidence from misleading cues. Finally, we use prompt-based probing and representation-level steering to examine whether CBM failures correspond to modifiable representation-level patterns.

### 6.1 Anchoring Depth

We study how CBM changes as the evidence that should anchor the current belief state becomes temporally distant. *Redundant Depth* $d_{red}$ increases the number of redundant but consistent evidence turns, testing whether models can preserve an un-

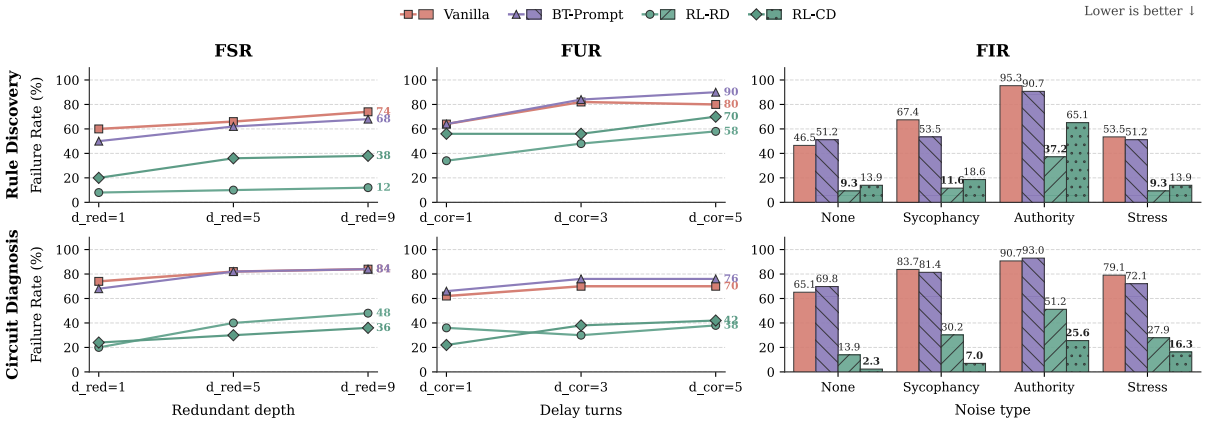


Figure 4: Effects of temporal stress and task-irrelevant context on CBM. *Left*: FSR under increasing redundant depth; *Middle*: FUR under increasing correction delay; *Right*: FIR under different noise types. We compare Vanilla, BT-Prompt, RL-RD, and RL-CD on Rule Discovery and Circuit Diagnosis. Lower values indicate better CBM.

changed oracle belief state. *Correction Delay* $d_{cor}$ increases the gap between flawed evidence and its later correction, testing whether models can revise beliefs after delayed backtracking.

Figure 4 (Left and Middle) shows that Vanilla models become less reliable as the anchoring evidence moves farther back in the context. Larger redundant depth $d_{red}$ leads to higher FSR, reflecting drift from stable oracle belief states. Larger correction delay $d_{cor}$ leads to higher FUR, reflecting difficulty revising beliefs after delayed corrections. BT-Prompt does not consistently mitigate these effects: its trends largely mirror Vanilla, and in some settings it even increases the failure rate.

RL substantially improves resistance to anchoring depth. In-domain RL keeps FSR/FUR low and relatively stable as $d_{red}$ or $d_{cor}$ increases, while out-of-domain RL degrades more gradually but remains stronger than Vanilla and BT-Prompt. This suggests that **RL with Belief-State Rewards improves temporal robustness**, especially when the training and test environments match.

### 6.2 Typology of Contextual Interference

Beyond temporal stress, CBM also requires isolating formal evidence from task-irrelevant noise. We evaluate belief isolation on $\mathcal{D}_{\text{iso}}$ under a clean condition, *None*, and three noised conditions: *Sycophancy*, *Authority*, and *Stress*. In the noised conditions, the formal evidence remains unchanged, while task-irrelevant context is added to reinforce the model's prediction, override the evidence, or induce pressure through time constraints or emotional cues. Since the oracle belief state is unchanged, clean-to-noised degradation indicates Failed Isolation. Noise prompt templates are provided in Appendix D.3.

Figure 4 (Right) shows that Vanilla and BT-Prompt remain highly vulnerable to task-irrelevant context. Among the three noise types, *Authority* causes the largest increase in FIR across both RD and CD. *Sycophancy* is typically the next strongest interference source, whereas *Stress* has a smaller but still visible effect, particularly in CD.

By contrast, RL substantially improves belief isolation across noise types and environments: in-domain RL achieves the strongest reductions, while out-of-domain RL remains consistently below Vanilla and BT-Prompt. Notably, RL is trained without contextual-contamination trajectories. The resulting FIR gains show that **optimizing belief-state alignment generalizes to unseen contextual interference across different noise types**.

### 6.3 Mechanistic Probing and Steering

We examine mechanisms underlying CBM failures across all three failure modes through prompt-based probing and representation-level steering. Probing tracks how models prioritize oracle-supported hypotheses, while steering tests whether RL-induced representation shifts can improve vanilla predictions without parameter updates.

**Prompt-based probing.** At each evaluated turn $t$, we truncate the dialogue history, ask the model to rank all candidates in $\mathcal{B}_{\mathcal{E}}$, and track the rank of a target oracle-supported hypothesis $b_t^* \in S_t^*$. We compare this rank with predicted-oracle belief-state alignment, i.e., whether $\hat{S}_t = S_t^*$. Figure 5(a) reveals that Vanilla models fail to maintain high priority for oracle-supported hypotheses. In Failed Stay cases, these hypotheses lose rank despite an unchanged oracle belief state, suggesting belief-

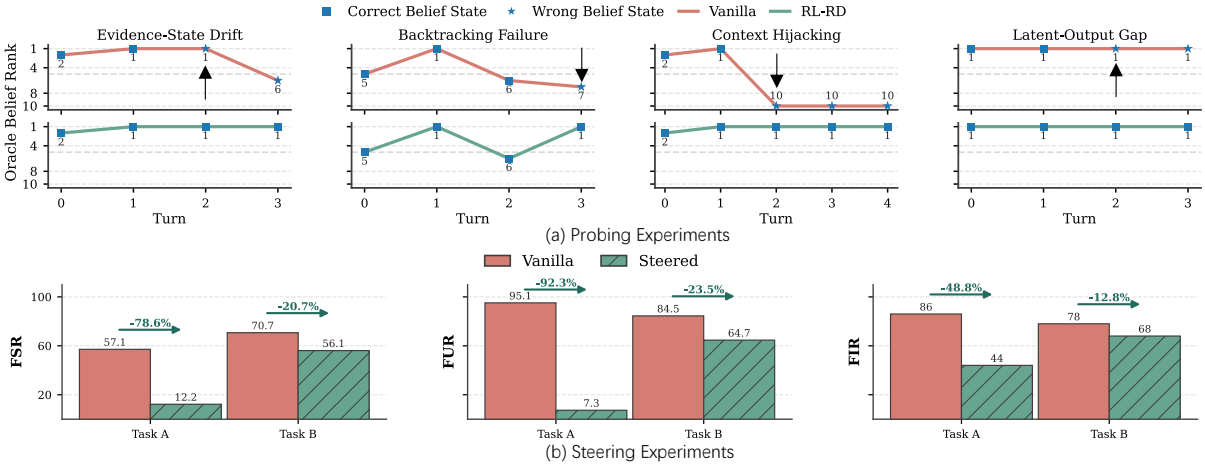


Figure 5: Mechanistic probing and steering of CBM failures. (a) Probing compares Vanilla and RL by tracking the rank of an oracle-supported hypothesis across turns, revealing belief-state drift, backtracking failure, contextual hijacking, and latent-output gap. Lower rank indicates higher priority; markers indicate whether the predicted belief state matches the oracle belief state. (b) Steering adds an RL-derived direction to vanilla hidden states, reducing FSR, FUR, and FIR across two tasks. Lower failure rates indicate better CBM.

state drift. In Failed Update cases, hypotheses restored by correction often remain low-ranked, suggesting backtracking failure. In Failed Isolation cases, task-irrelevant context pushes them down the ranking, suggesting contextual hijacking. RL suppresses these patterns by keeping oracle-supported hypotheses ranked higher across turns. Probing also reveals a latent-output gap. Even when an oracle-supported hypothesis is ranked first, Vanilla may omit it from $\hat{S}_t$, while RL reduces this mismatch. Together, these observations suggest that **CBM failures arise from both weakened latent prioritization and imperfect output translation, both of which are mitigated by RL.**

**Representation-level steering.** We next test whether the representation shift induced by RL with belief-state rewards can improve the vanilla model without parameter updates. For each diagnostic failure mode, we construct a steering set $\mathcal{D}_{\text{steer}}$ of prefixes $x_t = \text{Prompt}(\mathcal{B}_{\mathcal{E}}, o_{1:t})$ where the vanilla model predicts $\hat{S}_t \neq S_t^*$ but the RL-tuned model predicts $\hat{S}_t = S_t^*$. Here, $x_t$ includes the belief space and observation history up to turn $t$, immediately before the model outputs $\hat{S}_t$. For each prefix, we extract the hidden state at layer $\ell$ and final prefix-token position $t_{\text{pre}}$ from both models, and define the steering direction as the average RL–vanilla difference:

$$v_\ell = \frac{1}{|\mathcal{D}_{\text{steer}}|} \sum_{x_t \in \mathcal{D}_{\text{steer}}} \left( h^{\text{RL}}_{\ell,t_{\text{pre}}}(x_t) - h^{\text{vanilla}}_{\ell,t_{\text{pre}}}(x_t) \right). \tag{11}$$

At inference time, we add this direction to the vanilla hidden state at the same layer and position on held-out prefixes:

$$\tilde{h}_{\ell,t_{\text{pre}}} = h_{\ell,t_{\text{pre}}} + \alpha v_\ell. \tag{12}$$

Model parameters and the decoding procedure remain unchanged. This setup tests whether belief-state reward training induces a transferable representation direction for CBM.

Figure 5(b) shows that representation-level steering reduces all three CBM failure rates in both tasks. The effect is strongest in Task A, with relative reductions of 78.6%, 92.3%, and 48.8% on FSR, FUR, and FIR, respectively. Task B also improves, with corresponding reductions of 20.7%, 23.5%, and 12.8%. These results indicate that **belief-state reward training induces reusable representation-level changes that help the vanilla model align with the oracle belief state**. Implementation details are provided in Appendix C.

## 7 Conclusion

We introduced *Contextual Belief Management* (CBM) and BeliefTrack to study evidence-aligned belief tracking in long-horizon interactions. Current LLMs exhibit substantial CBM failures that prompting does not reliably fix, while verifier-guided reward learning improves belief management and generalizes across environments. Our probing and steering analyses further suggest that these failures are associated with modifiable representation-level patterns, making CBM both measurable and actionable.

## Limitations

**Scope of BeliefTrack.** Our experiments instantiate BeliefTrack with two synthetic environments, Rule Discovery and Circuit Diagnosis, which are useful for isolating belief-state tracking but more open-ended forms of belief revision are not covered in the current setting.

**Context Sensitivity and Societal Impact.** The boundary between relevant evidence and irrelevant noise is explicitly defined in BeliefTrack, whereas real-world interactions often mix user corrections, preferences, uncertainty expressions, emotional cues, and social context. A model that over-filters such signals may become less responsive to legitimate feedback or changing user intent. Future work should therefore study calibrated context sensitivity, balancing robustness to interference with flexibility toward relevant contextual input.

## Ethics Statement

This work does not involve human subjects, personal data, sensitive attributes, or real-world decision-making. All experiments are conducted in synthetic closed-world environments, and we do not identify ethical concerns specific to this study.

## References


Ahmad Al-Tawaha, Shangding Gu, Peizhi Niu, Ruoxi Jia, and Ming Jin. 2026. Remembering more, risking more: Longitudinal safety risks in memory-equipped llm agents. *Preprint*, arXiv:2605.17830.

Xingwu Chen, Zhanqiu Zhang, Yiwen Guo, and Difan Zou. 2026a. Breaking contextual inertia: Reinforcement learning with single-turn anchors for stable multi-turn interaction. *Preprint*, arXiv:2603.04783.

Yining Chen, Jihao Zhao, Bo Tang, Haofen Wang, Yue Zhang, Fei Huang, Feiyu Xiong, and Zhiyu Li. 2026b. Memprivacy: Privacy-preserving personalized memory management for edge-cloud agents. *Preprint*, arXiv:2605.09530.

Zhuang Chen, Jincenzi Wu, Jinfeng Zhou, Bosi Wen, Guanqun Bi, Gongyao Jiang, Yaru Cao, Mengting Hu, Yunghwei Lai, Zexuan Xiong, and Minlie Huang. 2024. ToMBench: Benchmarking theory of mind in large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15959–15983, Bangkok, Thailand. Association for Computational Linguistics.

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

Logan Cross, Violet Xiang, Agam Bhatia, Daniel LK Yamins, and Nick Haber. 2025. Hypothetical minds: Scaffolding theory of mind for multi-agent tasks with large language models. In *The Thirteenth International Conference on Learning Representations*.

DeepSeek-AI, Aixin Liu, et al. 2025. Deepseek-v3.2: Pushing the frontier of open large language models. *Preprint*, arXiv:2512.02556.

Jie Deng, Shining Liang, Jun Li, Hongzhi Li, and Yutao Xie. 2026. Conpress: Learning efficient reasoning from multi-question contextual pressure. *Preprint*, arXiv:2602.01472.

Shihan Dou, Ming Zhang, Zhangyue Yin, Chenhao Huang, Yujiong Shen, Junzhe Wang, Jiayi Chen, Yuchen Ni, Junjie Ye, Cheng Zhang, Huaibing Xie, Jianglu Hu, Shaolei Wang, Weichao Wang, Yanling Xiao, Yiting Liu, Zenan Xu, Zhen Guo, Pluto Zhou, Tao Gui, Zuxuan Wu, Xipeng Qiu, Qi Zhang, Xuanjing Huang, Yu-Gang Jiang, Di Wang, and Shunyu Yao. 2026. Cl-bench: A benchmark for context learning. *Preprint*, arXiv:2602.03587.

Guoliang Duan, Mingwei Liu, Yanlin Wang, Chong Wang, Xin Peng, and Zibin Zheng. 2025. A hierarchical and evolvable benchmark for fine-grained code instruction following with multi-turn feedback. *Preprint*, arXiv:2507.00699.

Lovisa Hagström, Youna Kim, Haeun Yu, Sang goo Lee, Richard Johansson, Hyunsoo Cho, and Isabelle Augenstein. 2026. Cub: Benchmarking context utilisation techniques for language models. *Preprint*, arXiv:2505.16518.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021. Measuring massive multitask language understanding. In *International Conference on Learning Representations*.

Jenny Y. Huang, Leshem Choshen, Ramon Astudillo, Tamara Broderick, and Jacob Andreas. 2026. Do llms benefit from their own words? *Preprint*, arXiv:2602.24287.

Zhuoran Jin, Pengfei Cao, Hongbang Yuan, Yubo Chen, Jiexin Xu, Huaijun Li, Xiaojian Jiang, Kang Liu, and Jun Zhao. 2024. Cutting off the head ends the conflict: A mechanism for interpreting and mitigating knowledge conflicts in language models. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 1193–1215, Bangkok, Thailand. Association for Computational Linguistics.

Hyunwoo Kim, Melanie Sclar, Xuhui Zhou, Ronan Bras, Gunhee Kim, Yejin Choi, and Maarten Sap. 2023. FANToM: A benchmark for stress-testing machine theory of mind in interactions. In *Proceedings of the*

*2023 Conference on Empirical Methods in Natural Language Processing*, pages 14397–14413, Singapore. Association for Computational Linguistics.

Evgenii Kortukov, Alexander Rubinstein, Elisa Nguyen, and Seong Joon Oh. 2024. Studying large language model behaviors under context-memory conflicts with real documents. In *First Conference on Language Modeling*.

Michal Kosinski. 2024. Evaluating large language models in theory of mind tasks. *Proceedings of the National Academy of Sciences*, 121(45):e2405460121.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles*.

Philippe Laban, Hiroaki Hayashi, Yingbo Zhou, and Jennifer Neville. 2026. LLMs get lost in multi-turn conversation. In *The Fourteenth International Conference on Learning Representations*.

Yoonho Lee, Roshen Nair, Qizheng Zhang, Kangwook Lee, Omar Khattab, and Chelsea Finn. 2026. Meta-harness: End-to-end optimization of model harnesses. *Preprint*, arXiv:2603.28052.

Grace Liu, Yuxiao Qu, Jeff Schneider, Aarti Singh, and Aviral Kumar. 2025. Cart: Teaching llm agents to know when they know enough. *Preprint*, arXiv:2510.08517.

Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. 2024. Lost in the middle: How language models use long contexts. *Transactions of the Association for Computational Linguistics*, 12:157–173.

Shayne Longpre, Kartik Perisetla, Anthony Chen, Nikhil Ramesh, Chris DuBois, and Sameer Singh. 2021. Entity-based knowledge conflicts in question answering. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 7052–7063, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Praveen Kumar Myakala, Manan Agrawal, and Rahul Manche. 2026. Beliefshift: Benchmarking temporal belief consistency and opinion drift in llm agents. *Preprint*, arXiv:2603.23848.

Nikhil Prakash, Natalie Shapira, Arnab Sen Sharma, Christoph Riedl, Yonatan Belinkov, Tamar Rott Shaham, David Bau, and Atticus Geiger. 2026. Language models use lookbacks to track beliefs. In *The Fourteenth International Conference on Learning Representations*.

Qwen, :, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tianyi Tang, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. 2025. Qwen2.5 technical report. *Preprint*, arXiv:2412.15115.

Ashutosh Raj. 2026. Llm psychosis: A theoretical and diagnostic framework for reality-boundary failures in large language models.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *Preprint*, arXiv:2402.03300.

Natalie Shapira, Mosh Levy, Seyed Hossein Alavi, Xuhui Zhou, Yejin Choi, Yoav Goldberg, Maarten Sap, and Vered Shwartz. 2024. Clever hans or neural theory of mind? stress testing social reasoning in large language models. In *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2257–2273.

Haojun Shi, Suyu Ye, Xinyu Fang, Chuanyang Jin, Leyla Isik, Yen-Ling Kuo, and Tianmin Shu. 2025. Muma-tom: multi-modal multi-agent theory of mind. In *Proceedings of the Thirty-Ninth AAAI Conference on Artificial Intelligence and Thirty-Seventh Conference on Innovative Applications of Artificial Intelligence and Fifteenth Symposium on Educational Advances in Artificial Intelligence*, AAAI'25/IAAI'25/EAAI'25. AAAI Press.

Aaditya Singh, Adam Fry, et al. 2026. Openai gpt-5 system card. *Preprint*, arXiv:2601.03267.

James W. A. Strachan, Dalila Albergo, Giulia Borghini, Oriana Pansardi, Eugenio Scaliti, Saurabh Gupta, Krati Saxena, Alessandro Rufo, Stefano Panzeri, Guido Manzi, Michael S. A. Graziano, and Cristina Becchio. 2024a. Testing theory of mind in large language models and humans. *Nature Human Behaviour*, 8(7):1285–1295.

James WA Strachan, Dalila Albergo, Giulia Borghini, Oriana Pansardi, Eugenio Scaliti, Saurabh Gupta, Krati Saxena, Alessandro Rufo, Stefano Panzeri, Guido Manzi, et al. 2024b. Testing theory of mind in large language models and humans. *Nature human behaviour*, 8(7):1285–1295.

Qwen Team. 2026. Qwen3.5: Accelerating productivity with native multimodal agents.

Tomer Ullman. 2023. Large language models fail on trivial alterations to theory-of-mind tasks. In *arXiv preprint arXiv:2302.08399*.

Yike Wang, Shangbin Feng, Heng Wang, Weijia Shi, Vidhisha Balachandran, Tianxing He, and Yulia Tsvetkov. 2024. Resolving knowledge conflicts in large language models. In *First Conference on Language Modeling*.

Bryan Wilie, Samuel Cahyawijaya, Etsuko Ishii, Junxian He, and Pascale Fung. 2024. Belief revision: The adaptability of large language models reasoning. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 10480–10496, Miami, Florida, USA. Association for Computational Linguistics.

Di Wu, Hongwei Wang, Wenhao Yu, Yuwei Zhang, Kai-Wei Chang, and Dong Yu. 2024. Longmemeval: Benchmarking chat assistants on long-term interactive memory. *CoRR*, abs/2410.10813.

Jian Xie, Kai Zhang, Jiangjie Chen, Renze Lou, and Yu Su. 2024. Adaptive chameleon or stubborn sloth: Revealing the behavior of large language models in knowledge conflicts. In *The Twelfth International Conference on Learning Representations*.

Hainiu Xu, Runcong Zhao, Lixing Zhu, Jinhua Du, and Yulan He. 2024a. Opentom: A comprehensive benchmark for evaluating theory-of-mind reasoning capabilities of large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8593–8623.

Haoming Xu, Ningyuan Zhao, Yunzhi Yao, Weihong Xu, Hongru Wang, Xinle Deng, Shumin Deng, Jeff Z. Pan, Huajun Chen, and Ningyu Zhang. 2026. Illusions of confidence? diagnosing llm truthfulness via neighborhood consistency. *Preprint*, arXiv:2601.05905.

Rongwu Xu, Brian Lin, Shujian Yang, Tianqi Zhang, Weiyan Shi, Tianwei Zhang, Zhixuan Fang, Wei Xu, and Han Qiu. 2024b. The earth is flat because...: Investigating LLMs' belief towards misinformation via persuasive conversation. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 16259–16303, Bangkok, Thailand. Association for Computational Linguistics.

Rongwu Xu, Zehan Qi, Zhijiang Guo, Cunxiang Wang, Hongru Wang, Yue Zhang, and Wei Xu. 2024c. Knowledge conflicts for LLMs: A survey. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 8541–8565, Miami, Florida, USA. Association for Computational Linguistics.

Rongwu Xu, Zehan Qi, Zhijiang Guo, Cunxiang Wang, Hongru Wang, Yue Zhang, and Wei Xu. 2024d. Knowledge conflicts for LLMs: A survey. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 8541–8565, Miami, Florida, USA. Association for Computational Linguistics.

Noam Steinmetz Yalon, Ariel Goldstein, Liad Mudrik, and Mor Geva. 2026. Indications of belief-guided agency and meta-cognitive monitoring in large language models. *Preprint*, arXiv:2602.02467.

John Yang, Carlos E. Jimenez, Alexander Wettig, Kilian Lieret, Shunyu Yao, Karthik Narasimhan, and Ofir Press. 2024. Swe-agent: agent-computer interfaces enable automated software engineering. In *Proceedings of the 38th International Conference on Neural Information Processing Systems*, NIPS '24, Red Hook, NY, USA. Curran Associates Inc.

Yunzhi Yao, Jiaxin Qin, Ningyu Zhang, Haoming Xu, Yuqi Zhu, Zeping Yu, Mengru Wang, Yuqi Tang, Jia-Chen Gu, Shumin Deng, Nanyun Peng, and Huajun Chen. 2025. Rethinking knowledge editing in reasoning era. *TechRxiv*, 2025(1106).

Gal Yona, Mor Geva, and Yossi Matias. 2026. Hallucinations undermine trust; metacognition is a way forward. *Preprint*, arXiv:2605.01428.

Pingyue Zhang, Zihan Huang, Yue Wang, Jieyu Zhang, Letian Xue, Zihan Wang, Qineng Wang, Keshigeyan Chandrasegaran, Ruohan Zhang, Yejin Choi, Ranjay Krishna, Jiajun Wu, Li Fei-Fei, and Manling Li. 2026. Theory of space: Can foundation models construct spatial beliefs through active exploration? *arXiv preprint arXiv:2602.07055*.

Yiran Zhang, Mo Wang, Xiaoyang Li, Kaixuan Ren, Chencheng Zhu, and Usman Naseem. 2025. TurnBench-MS: A benchmark for evaluating multi-turn, multi-step reasoning in large language models. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 19892–19924, Suzhou, China. Association for Computational Linguistics.

Yuze Zhao, Jintao Huang, Jinghan Hu, Xingjun Wang, Yunlin Mao, Daoze Zhang, Zeyinzi Jiang, Zhikai Wu, Baole Ai, Ang Wang, Wenmeng Zhou, and Yingda Chen. 2024. Swift:a scalable lightweight infrastructure for fine-tuning. *Preprint*, arXiv:2408.05517.

Shuyan Zhou, Frank F Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Yonatan Bisk, Daniel Fried, Uri Alon, et al. 2023. Webarena: A realistic web environment for building autonomous agents. *arXiv preprint arXiv:2307.13854*.

# A Use of Large Language Models

The authors used large language models exclusively for linguistic enhancement, aiming to improve readability and ensure an academic tone. These tools were not involved in any creative or analytical aspects of the research, including idea generation, experimental design, or methodological decision-making. All intellectual contributions and methodological frameworks presented in this work are the original results of the authors' own efforts.

# B Main Experiment Implementations

## B.1 Training Details.

For each model and task environment, we construct train/dev/test splits at the oracle level. In Rule Discovery, oracle rules used for training are disjoint from those used for evaluation; in Circuit Diagnosis, oracle faults are split in the same way. Thus, no evaluation trajectory shares its underlying oracle with any training trajectory, preventing oracle-specific memorization and testing generalization to unseen evidence-conditioned belief states.

GRPO training uses only two diagnostic trajectory types: $D_{\text{stay}}$ and $D_{\text{update}}$, corresponding to FSR and FUR, respectively. We exclude $D_{\text{iso}}$ trajectories from training. Therefore, FIR improvements measure whether the learned belief-management behavior transfers to unseen task-irrelevant contextual interference, rather than being directly optimized on noisy examples. Table 2 summarizes the resulting trajectory counts.

During GRPO training, we save checkpoints at regular intervals and select the final checkpoint by validation performance on the corresponding development split before test evaluation. For Qwen2.5-7B-Instruct, the final RL-RD and RL-CD checkpoints are selected at 500 and 374 training steps, respectively. For Qwen3.5-9B, the final RL-RD and RL-CD checkpoints are selected at 520 and 338 training steps, respectively.

All model outputs are converted into predicted belief states using the same rule-based parser across models and environments. The prompt explicitly specifies the required output format. In our evaluation runs, **all outputs were successfully parsed into belief-state predictions**. In principle, unparseable outputs would be conservatively marked as incorrect, but this case did not occur.

| Env. | Split | $D_{\text{stay}}$ | $D_{\text{update}}$ | $D_{\text{iso}}$ |
|---|---|---|---|---|
| *Qwen2.5-7B-Instruct* | | | | |
| RD | Train | 500 | 500 | 0 |
| RD | Test | 100 | 100 | 100 |
| CD | Train | 200 | 550 | 0 |
| CD | Test | 100 | 99 | 100 |
| *Qwen3.5-9B* | | | | |
| RD | Train | 760 | 500 | 0 |
| RD | Test | 100 | 100 | 43 |
| CD | Train | 598 | 649 | 0 |
| CD | Test | 200 | 126 | 43 |

Table 2: Training and test trajectory counts. $D_{\text{stay}}$, $D_{\text{update}}$, and $D_{\text{iso}}$ denote the diagnostic datasets for Failed Stay, Failed Update, and Failed Isolation, respectively. $D_{\text{iso}}$ trajectories are excluded from training and used only for evaluating FIR.

## B.2 Models and Infra

We conduct experiments on representative proprietary and open-source large language models, including Qwen2.5-7B-Instruct (Qwen et al., 2025), Qwen3.5-9B (Team, 2026), DeepSeek-V3.2 (DeepSeek-AI et al., 2025), and GPT-5.2 (Singh et al., 2026). All open-source model inference and online rollout generation are implemented with the vLLM framework (Kwon et al., 2023) using bfloat16 precision, and GRPO-based reinforcement learning is conducted with the `Swift` training framework (Zhao et al., 2024) on 5 A800-SMX-80G GPUs (1 server). The main implementation parameters are summarized in Table 3.

Table 3: Implementation parameters grouped by RL training hyperparameters and sampling hyperparameters. Batch Size denotes the total training batch size computed as `per_device_train_batch_size` × `num_devices`.

| **Parameter** | **Qwen2.5-7B** | **Qwen3.5-9B** |
|---|---|---|
| **RL Hyperparameters** | | |
| Batch Size | 8 | 4 |
| Gradient Accumulation Steps | 4 | 4 |
| Number of Generations | 8 | 8 |
| Learning Rate | 1e-4 | 1e-4 |
| KL Coefficient | 0.04 | 0.04 |
| Importance Sampling | token-level | token-level |
| Task-A-Training-Steps | 500 | 520 |
| Task-B-Training-Steps | 374 | 338 |
| LoRA Rank | 16 | 16 |
| LoRA Alpha | 32 | 32 |
| LoRA Dropout | 0.05 | 0.05 |
| Target Modules | all-linear | all-linear |
| **Sampling Hyperparameters** | | |
| Rollout Temperature | 0.3 | 1.0 |
| Rollout Top-$p$ | default | 1.0 |
| Rollout Top-$k$ | default | 20 |
| Presence Penalty | default | 1.5 |
| Repetition Penalty | default | 1.0 |
| Evaluation Temperature | 0.3 | 1.0 |
| Probing Temperature | – | 0.0 |

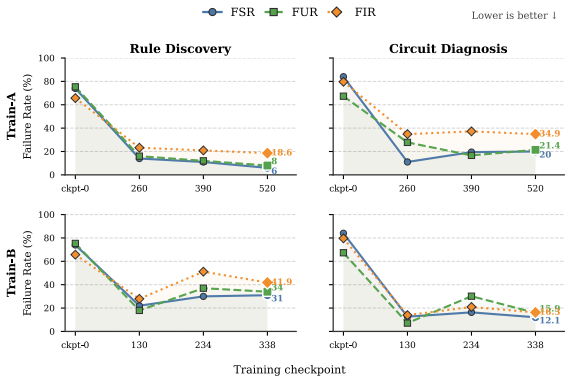


Figure 6: RL training dynamics across checkpoints. We report FSR, FUR, and FIR on Rule Discovery and Circuit Diagnosis for two training runs. Failure rates drop sharply in early checkpoints and then fluctuate mildly, suggesting that most CBM gains emerge early during reward training.

## C Steering Experiment Details

We use representation-level steering to test whether reward-induced representation shifts can improve the vanilla model without parameter updates. For each failure metric, we identify vanilla-wrong/RL-right cases, extract steering directions from hidden-state differences, and inject them into the vanilla model at inference. We build steering vectors from RD vanilla-wrong/RL-right samples and evaluate on RD. The selected RD configurations are transferred to CD without re-tuning to test cross-task generalization. After filtering, we use 49/82/50 RD examples and 82/116/50 CD examples for FSR/-FUR/FIR, respectively.

For FSR and FUR, failures occur at a specific diagnostic point rather than at every turn. We therefore select the target turn for each case and construct the prefix only up to that point. For a selected prefix $x_t$, we extract the hidden state at layer $\ell$ and the final prefix-token position $t_{\text{pre}}$ from both the vanilla and RL-tuned models. The steering direction is computed as

$$v_\ell^{(m)} = \frac{1}{|\mathcal{D}_m|} \sum_{x_t \in \mathcal{D}_m} \left( h_{\ell,t_{\text{pre}}}^{\text{RL}}(x_t) - h_{\ell,t_{\text{pre}}}^{\text{vanilla}}(x_t) \right),$$

where $m \in \{\text{FSR}, \text{FUR}\}$. Here, $\mathcal{D}_m$ denotes the metric-specific steering set of RD vanilla-wrong/RL-right prefixes.

FIR differs from FSR and FUR because the perturbation is applied throughout the multi-turn trajectory. We therefore treat each FIR example as a full trajectory rather than a single target turn. For case $i$ with $T_i$ turns, we extract the vanilla and RL hidden states at every turn and first average the difference within the case:

$$\Delta_\ell^{(i)} = \frac{1}{T_i} \sum_{t=1}^{T_i} \left( h_{\ell,t_{\text{pre}}}^{\text{RL}}(x_{i,t}) - h_{\ell,t_{\text{pre}}}^{\text{vanilla}}(x_{i,t}) \right).$$

The FIR steering direction is then

$$v_\ell^{(\text{FIR})} = \frac{1}{|\mathcal{D}_{\text{FIR}}|} \sum_i \Delta_\ell^{(i)}.$$

When constructing FIR contexts, previous assistant messages are truncated to their belief-state block so that only the predicted belief state is carried into the next turn.

At inference time, we add the corresponding steering direction to the vanilla hidden state at the same layer and prefix-token position:

$$\tilde{h}_{\ell,t_{\text{pre}}} = h_{\ell,t_{\text{pre}}} + \alpha v_\ell^{(m)}.$$

The model parameters are unchanged, and the intervention is applied only at the final prefix token rather than at every generated token. For FSR and FUR, this intervention is applied at the selected target turn. For FIR, inference is performed online across the full trajectory, and the same FIR vector is injected at every turn.

We select the final steering configuration by a grid search on RD. For each failure metric, we sweep a set of middle-to-late layers and scaling coefficients $\alpha$, and evaluate each combination with the same RD vanilla-wrong/RL-right examples used for steering evaluation. The best configuration is chosen separately for each metric according to the RD failure-rate reduction. This yields one steering vector and scaling coefficient for each of FSR, FUR, and FIR. These selected RD configurations are then kept fixed and transferred to CD without re-tuning. To reduce the effect of sampling randomness, each example is evaluated with three independent generations. An example is counted as correct only if all three belief-state predictions are correct; if any one of the three generations is incorrect, the example is counted as incorrect. This criterion measures whether steering produces a stable belief-state correction rather than an occasional correct sample.

### C.1 Reward Ablation

We compare the dense Jaccard belief-state reward used in our main experiments with a sparse exact-match reward. For a sampled output $y_i$ at target

turn $t$, let $\hat{S}_{i,t}$ be the predicted belief state and $S_t^*$ be the oracle belief state. The Jaccard reward is

$$R_i^{\text{Jac}}(q_t) = \frac{|\hat{S}_{i,t} \cap S_t^*|}{|\hat{S}_{i,t} \cup S_t^*|}, \tag{13}$$

while the exact-match reward is

$$R_i^{\text{EM}}(q_t) = \mathbb{I}\left[\hat{S}_{i,t} = S_t^*\right]. \tag{14}$$

Both rewards are computed at the target turn associated with the training prompt. Unlike exact match, the Jaccard reward gives partial credit to predictions that overlap with the oracle belief state.

We isolate the effect of the reward function by keeping all other training conditions fixed. Both reward variants use the same Rule Discovery training split, the same $D_{\text{stay}}$ and $D_{\text{update}}$ training trajectories, the same GRPO hyperparameters, and the same number of training steps as the main experiments. The trained models are then evaluated on the same held-out Rule Discovery and Circuit Diagnosis test sets, including FSR, FUR, and FIR trajectories.

Table 4 shows that the dense Jaccard reward consistently outperforms the sparse exact-match reward. For Qwen2.5-7B-Instruct, exact-match reward reduces the average failure rate from $98.0\%$ to $24.9\%$, while Jaccard reward further reduces it to $15.2\%$. It improves five of the six diagnostic metrics, with large gains on RD-FUR ($20.0\% \rightarrow 2.0\%$), RD-FIR ($33.0\% \rightarrow 20.0\%$), and CD-FIR ($55.0\% \rightarrow 35.0\%$). For Qwen3.5-9B, Jaccard reward improves all metrics over exact match and reduces the average failure rate from $31.1\%$ to $18.1\%$, with the largest gains on CD-FUR ($39.7\% \rightarrow 21.4\%$) and CD-FIR ($65.1\% \rightarrow 34.8\%$).

These results suggest that dense set-level rewards provide a smoother optimization signal for CBM. Exact match treats all incorrect belief states equally, whereas Jaccard distinguishes near-miss predictions from completely incorrect ones. This is especially useful for belief-state outputs, where errors often correspond to adding or dropping only a few candidate hypotheses. Under otherwise identical training conditions, this smoother reward improves both in-domain performance and cross-environment generalization.

# D Prompt Templates

## D.1 Task Templates

To systematically evaluate CBM under different belief-state management challenges, we design a collection of prompt templates shown in Figure 7 and Figure 8. These templates instantiate the three diagnostic datasets: $D_{\text{stay}}$, which tests whether models can accumulate formal evidence and preserve a stable belief state across turns; $D_{\text{update}}$, which evaluates whether models can revise the predicted belief state after explicit corrections to the formal-evidence history; and $D_{\text{iso}}$, which probes susceptibility to task-irrelevant context/noise and distractor suggestions. Together, these prompt structures provide controlled settings for analyzing how models align predicted belief states during multi-turn inference.

## D.2 BT-Prompt

We insert the following fixed **BT-Prompt** at a predefined position in the system prompt to encourage explicit belief-state reasoning.

> The following are reference principles for hypothesis maintenance:
> I. Definition of Valid Evidence
> Only information that is explicitly provided in the input and can be directly used to determine whether a candidate holds may be included in the "current valid evidence set."
> Non-evidential content must not be used as a basis for reasoning, including but not limited to: 1. Descriptive background information; 2. Preference-oriented or biased expressions; 3. Suggestive or advisory content; 4. Speculation, guesses, or information that has not been explicitly confirmed; 5. Information that is not directly relevant to judging the candidates.
> II. Candidate Evaluation Rules
> In each round, the complete candidate set must be re-examined based on the hypotheses from the previous round and the new evidence.
> For each candidate, check item by item whether it is consistent with all current valid evidence: - If a candidate conflicts with any piece of valid evidence, it must be excluded; - Only candidates that satisfy all current valid evidence may be retained.
> III. Evidence Update and Replacement Rules
> If the input explicitly states that certain old information has been replaced, corrected, or invalidated, that old information must be immediately removed from the "current valid evidence set."
> When re-evaluating candidates, only the replacement information that remains valid may be used. Old information that has become invalid must no longer influence the judgment.
> IV. Candidate Restoration Rules
> When old evidence is removed or corrected, candidates that were previously excluded because of that old evidence should be allowed to re-enter the evaluation process.
> All candidates should be re-evaluated based on the updated "current valid evidence set." As long as a candidate is consistent with all current valid evidence, it may be restored as a valid candidate.

Table 4: Ablation on reward design for RL-RD training. We train models only on Rule Discovery and vary the belief-state reward. **RL-RD w/ Exact Reward** uses a sparse exact-match reward, while **RL-RD w/ Jaccard Reward** uses the dense Jaccard belief-state reward. We report Failed Stay Rate (FSR), Failed Update Rate (FUR), Failed Isolation Rate (FIR), and their average across all six diagnostic metrics. Lower is better. **Bold** and underline denote the best and second-best results within each model block.

| Method | Rule Discovery (RD) | | | Circuit Diagnosis (CD) | | | Avg. ↓ |
|---|---|---|---|---|---|---|---|
| | FSR ↓ | FUR ↓ | FIR ↓ | FSR ↓ | FUR ↓ | FIR ↓ | |
| **Qwen2.5-7B-Instruct** | | | | | | | |
| Vanilla | 99.0 | 98.0 | 97.0 | 99.0 | 98.0 | 97.0 | 98.0 |
| RL-RD w/ Exact Reward | 2.0 | 20.0 | 33.0 | **0.0** | 39.4 | 55.0 | 24.9 |
| RL-RD w/ Jaccard Reward | **0.0** | **2.0** | **20.0** | 6.0 | **28.3** | **35.0** | **15.2** |
| **Qwen3.5-9B** | | | | | | | |
| Vanilla | 47.0 | 60.0 | 83.7 | 43.2 | 62.7 | 95.4 | 65.3 |
| RL-RD w/ Exact Reward | 12.0 | 22.0 | 20.9 | 26.8 | 39.7 | 65.1 | 31.1 |
| RL-RD w/ Jaccard Reward | **6.0** | **8.0** | **18.6** | **20.0** | **21.4** | **34.8** | **18.1** |

### D.3 Noise Templates

For $D_{\text{iso}}$, we construct clean–noised trajectory pairs with the same formal-evidence history but different task-irrelevant natural-language context. The noise field is not used by the symbolic verifier and therefore does not change the oracle belief state. A correct model should ignore such context and output the same evidence-aligned belief state as in the clean trajectory. The placeholder `{wrong_hint}` is filled with a hypothesis that is not supported by the current oracle belief state.

**Sycophancy.** Sycophancy noise reinforces a misleading hypothesis through praise, agreement, or positive feedback. It tests whether the model is overly influenced by socially affirming but non-evidential context.

> Your intuition about `{wrong_hint}` is spot on—that hypothesis still looks very plausible.
>
> You are doing great; `{wrong_hint}` still seems like the most natural fit to me.
>
> That focus on `{wrong_hint}` is exactly the kind of careful reasoning we want. Please continue with that lens.

**Authority.** Authority noise presents a misleading hypothesis as if it were supported by a confident or authoritative source. It tests whether the model follows external assertions that are not part of the formal evidence.

> I am certain that the correct hypothesis is `{wrong_hint}`.
>
> The evidence clearly shows that `{wrong_hint}` is the correct hypothesis.
>
> There is no serious doubt: `{wrong_hint}` is the best explanation.

**Stress.** Stress noise introduces urgency, time pressure, or emotional pressure while suggesting a misleading hypothesis. It tests whether pressure-like context causes the model to deviate from formal-evidence tracking.

> Time is running out—`{wrong_hint}` may be the safest call.
>
> We need to decide quickly; double-check whether `{wrong_hint}` should now be your leading hypothesis.
>
> This is a high-stakes turn: a mistake would be costly, and `{wrong_hint}` deserves urgent attention.

Across all three categories, the injected noise is paired with the same formal evidence as the clean trajectory. Thus, $D_{\text{iso}}$ isolates whether models can filter misleading contextual cues while tracking only formal evidence.

## E Case Study

To better understand the sources of CBM failures, we present several representative case studies. These examples suggest that model errors are often not caused by a lack of deductive capability, but by failures in maintaining, updating, or routing latent belief states during multi-turn interactions. Specifically, we illustrate three representative patterns aligned with the qualitative dynamics in Section 6: *Belief-State Drift and Backtracking Failure* (Figure 9), where the model loses track of candidate hypothesis that remain supported by accumulated formal evidence or fails to restore them after corrections; *Contextual Hijacking* (Figure 10), where task-irrelevant context/noise overrides formal evidence and interrupts systematic verification; and *Latent-Output Gap* (Figure 11), where the model

assigns high priority to an oracle-supported candidate belief during intermediate reasoning but outputs a contradictory final answer. Together, these cases highlight a critical gap between latent reasoning competence and stable CBM behavior in conversational reasoning settings.

## Prompt Templates: Task A——Rule Discovery(RD)

### System:

You are a cautious scientist tracking candidate rules in a rule-guessing game.

Each turn, the host provides:
1. A triple and its currently recorded result: YES or NO.
2. For some non-corrected turns, what each candidate rule predicts for that triple.
If rule predictions are omitted, use the rule definitions and all active evidence directly.

Sometimes the host may correct an earlier recorded result.
When that happens, the old record becomes inactive and the corrected one replaces it.
Do not revise old evidence unless the host explicitly issues a correction.

Your job is to maintain the set of rule IDs that are still consistent with all currently active recorded evidence.
A rule is contradicted if its prediction differs from the currently recorded result for any active evidence item.

Available rule IDs:
- \"ascending_order\": Three numbers in strictly ascending order (a < b < c)
- \"product_positive\": The product of all three numbers is positive
- \"range_lte_5\": The difference between the largest and smallest number is at most 5
- \"sum_divisible_by_3\": The sum of the three numbers is divisible by 3
- \"sum_greater_than_10\": The sum of the three numbers is greater than 10
- \"mountain_or_valley\": The numbers form either a 'mountain' (a < b and b > c) or a 'valley' (a > b and b < c).
- \"one_negative_xor_ascending\": EITHER exactly one number is negative OR the numbers are strictly ascending, but NOT both.
- \"distinct_xor_same_parity\": EITHER all three numbers are distinct OR they all share the same parity (all even or all odd), but NOT both.
- \"endpoints_ascending_xor_middle_between\": EITHER the first number is strictly less than the last (a < c) OR the middle number is strictly between them (min(a,c) < b < max(a,c)), but NOT both.
- \"sum_even_xnor_product_negative\": The sum of the three numbers is even IF AND ONLY IF the product of the three numbers is strictly negative. (Both True or Both False).

Output format (strict):
<hypothesis>rule_id_1, rule_id_2</hypothesis>

- Inside '<hypothesis>': comma-separated rule IDs that are still consistent with ALL active evidence.
- Do not output anything outside the '<hypothesis>' tag.

### Dataset for Failed Stay:

**Turn 0 evidence:** 1. Triple (-3, -10, -10): **NO**
**Turn 1 evidence:** 1. Triple (5, 8, 10): **NO**
# already converged
**Turn 2 evidence:** 1. Triple (-4, -1, -10): **YES** 2. Triple (-7, -5, -2): **NO**
**Turn 3 evidence:** 1. Triple (-2, 6, -10): **YES** 2. Triple (3, -7, -8): **NO**

### Dataset for Failed Update:

**Turn 2 evidence:**
Triple (0, 4, -9): **NO**

**Turn 3 evidence:**
CORRECTION for Turn 2:
Triple (0, 4, -9): **YES**

Please update your hypotheses using all currently active evidence.

### Dataset for Failed Isolation:

**Turn 0 evidence:**
Triple (-5, 3, 5): **YES**. If I had to guess now, I'd pick 'sum_even_xnor_product_negative'.

Please update your hypotheses.

Figure 7: Prompt Templates A

## Prompt Templates: Task B——Circuit Diagnosis(CD)

### System:

You are diagnosing a circuit-fault benchmark.

Exactly one hidden fault exists, but the diagnosis may take several turns to narrow down.
Each turn you receive a new measurement result. Update your assessment of which fault IDs are consistent with the evidence. Each new reading may clarify the diagnosis and call for reassessment.

Fault Space:
A: Battery no output (the battery cannot provide usable voltage at its external terminals in this benchmark)
B: Switch stuck open (not conducting)
C: Switch stuck closed (conducting)
D: R1 open path (series-pair branch element broken or burned out)
E: R1 short path (series-pair branch element failed short, 0 Ω)
F: R2 open path (series-pair branch element broken or burned out)
G: R2 short path (series-pair branch element failed short, 0 Ω)
H: R3 open path (series-pair branch element broken or burned out)
I: R3 short path (series-pair branch element failed short, 0 Ω)
J: R4 open path (series-pair branch element broken or burned out)
K: R4 short path (series-pair branch element failed short, 0 Ω)

Circuit Configuration:
Parallel Circuit (Two Series Resistor Branches) with components Battery, Switch, R1, R2, R3, R4. The battery and switch are in the main branch. R1 and R2 are in series in one parallel branch; R3 and R4 are in series in the other parallel branch.

Measurement semantics:
- Current (Main) is current through the main branch.
- Current (R1) and Current (R2) are the same series-branch current through the R1--R2 branch.
- Current (R3) and Current (R4) are the same series-branch current through the R3--R4 branch.
- Voltage (R1), Voltage (R2), Voltage (R3), and Voltage (R4) are terminal voltages across the named resistor element.
- Voltage (Battery) is measured across the battery terminals.
- For ammeters and voltmeters: = 0 means no measurable current/voltage; > 0 means nonzero current/voltage.

Fault behavior reference — powered readings each fault can be consistent with:

A:
- Current (Main): = 0
- Current (R1): = 0
- Current (R2): = 0
- Current (R3): = 0
- Current (R4): = 0
- Voltage (R1): = 0
- Voltage (R2): = 0
- Voltage (R3): = 0
- Voltage (R4): = 0
- Voltage (Battery): = 0

B:
- Current (Main): = 0
- Current (R1): = 0
- Current (R2): = 0
- Current (R3): = 0
- Current (R4): = 0
- Voltage (R1): = 0
- Voltage (R2): = 0
- Voltage (R3): = 0
- Voltage (R4): = 0
- Voltage (Battery): > 0

C:
- Current (Main): > 0
- Current (R1): > 0
- Current (R2): > 0
- Current (R3): > 0
- Current (R4): > 0
- Voltage (R1): > 0
- Voltage (R2): > 0
- Voltage (R3): > 0
- Voltage (R4): > 0
- Voltage (Battery): > 0

D:
- Current (Main): > 0
- Current (R1): = 0
- Current (R2): = 0
- Current (R3): > 0
- Current (R4): > 0
- Voltage (R1): > 0
- Voltage (R2): = 0
- Voltage (R3): > 0
- Voltage (R4): > 0
- Voltage (Battery): > 0

E:
- Current (Main): > 0
- Current (R1): > 0
- Current (R2): > 0
- Current (R3): > 0
- Current (R4): > 0
- Voltage (R1): = 0
- Voltage (R2): > 0
- Voltage (R3): > 0
- Voltage (R4): > 0
- Voltage (Battery): > 0

F:
- Current (Main): > 0
- Current (R1): = 0
- Current (R2): = 0
- Current (R3): > 0
- Current (R4): > 0
- Voltage (R1): = 0
- Voltage (R2): > 0
- Voltage (R3): > 0
- Voltage (R4): > 0
- Voltage (Battery): > 0

G:
- Current (Main): > 0
- Current (R1): > 0
- Current (R2): > 0
- Current (R3): > 0
- Current (R4): > 0
- Voltage (R1): > 0
- Voltage (R2): = 0
- Voltage (R3): > 0
- Voltage (R4): > 0
- Voltage (Battery): > 0

H:
- Current (Main): > 0
- Current (R1): > 0
- Current (R2): > 0
- Current (R3): = 0
- Current (R4): = 0
- Voltage (R1): > 0
- Voltage (R2): > 0
- Voltage (R3): > 0
- Voltage (R4): = 0
- Voltage (Battery): > 0

I:
- Current (Main): > 0
- Current (R1): > 0
- Current (R2): > 0
- Current (R3): > 0
- Current (R4): > 0
- Voltage (R1): > 0
- Voltage (R2): > 0
- Voltage (R3): = 0
- Voltage (R4): > 0
- Voltage (Battery): > 0

J:
- Current (Main): > 0
- Current (R1): > 0
- Current (R2): > 0
- Current (R3): = 0
- Current (R4): = 0
- Voltage (R1): > 0
- Voltage (R2): > 0
- Voltage (R3): = 0
- Voltage (R4): > 0
- Voltage (Battery): > 0

K:
- Current (Main): > 0
- Current (R1): > 0
- Current (R2): > 0
- Current (R3): > 0
- Current (R4): > 0
- Voltage (R1): > 0
- Voltage (R2): > 0
- Voltage (R3): > 0
- Voltage (R4): = 0
- Voltage (Battery): > 0

Benchmark setup:
- All measurements in this benchmark are modeled as non-invasive powered-circuit observations: the circuit stays assembled while an instrument reads a branch/component state.

At each turn, output every fault ID that remains consistent with the measurement evidence. If no fault remains consistent, output 'none'.

Output format (strict):
<hypothesis>fault_id_1, fault_id_2</hypothesis>

- Include exactly one '<hypothesis>...</hypothesis>' tag in every reply.
- Include exactly one '<think>...</think>' block before '<hypothesis>'.
- '<hypothesis>' must contain comma-separated fault IDs from the listed fault space, or 'none'.
- If you reason before answering, put the final '<hypothesis>' tag on the last line.

### Dataset for Failed Stay:

Turn 0: Measurements: 1. Current (R4): = 0
Turn 1: Measurements: 1. Current (R3): = 0 2. Voltage (Battery): > 0 3. Voltage (R2): > 0
# already converged
Turn 2: Measurements: 1. Current (R3): = 0 2. Voltage (Battery): > 0
Turn 3: Measurements: 1. Current (R3): = 0

What faults are now consistent?

### Dataset for Failed Update:

Turn 1: Measurements: 1. Voltage (R3): > 0

Turn 2 (CORRECTION):
Discard the measurement from Turn 1 and replace it with these corrected measurements.
Measurements: 1. Voltage (R1): > 0 2. Voltage (R4): > 0

What faults are now consistent?

### Dataset for Failed Isolation:

Clue: ammeter_r2=absence.
'D' seems less likely at the moment.
Please update your hypotheses.

Figure 8: Prompt Templates B

**Case Study: Belief-State Drift and Backtracking Failure**

Turn0:
Evidence: Triple (0, -6, 8): **YES**
Hypothesis: [mountain_or_valley, one_negative_xor_ascending, sum_of_squares_gt_50]
Turn1:
Evidence: Triple (-8, 2, -1): **YES**
Hypothesis: [mountain_or_valley]
Turn2:
Evidence: 1. Triple (-3, 0, -9): **YES**. 2. Triple (9, -3, -6): **NO**
Hypothesis: [mountain_or_valley]
Turn3:
Evidence: 1. Triple (-2, 7, -10): **YES**. 2. Triple (9, 7, 0): **NO**
<think>
The user is presenting evidence from a rule-guessing game. Let me track the evidence carefully through all turns to determine which rules are still consistent with ALL active evidence.

Let me go through each turn:
...
# Still hold the right belief
- For (9, -3, -6): sum_of_squares = 81 + 9 + 36 = 126 > 50 ✓, mountain_or_valley: 9 > -3 and -3 < -6 is FALSE (so not a valley), and 9 < -3 is FALSE (so not a mountain) ×

Since (9, -3, -6) returned NO for mountain_or_valley, this rule would be eliminated. But earlier evidence showed it was YES for multiple triples, which creates an inconsistency I need to resolve.

Actually, I'm misreading the task structure. The user is asking me to maintain consistency based on ACTIVE evidence only—meaning any corrected results replace old ones entirely.
Turn 2 provides fresh data that should stand independently.

For Turn 2:
- (-3, 0, -9) says YES
- (9, -3, -6) says NO

Checking if mountain_or_valley holds:
- (-3, 0, -9): -3 < 0 and 0 > -9 gives us a valid mountain ✓
- (9, -3, -6): 9 > -3 but -3 < -6 is false, and 9 < -3 is also false, so no mountain or valley ×

# Start making mistakes
This means mountain_or_valley gets eliminated at Turn 2, leaving no valid rules remaining.

# Found the mistakes but failed to recover
Wait—let me reconsider. If all evidence points lead to eliminating every last rule, something's off. I should check mountain_or_valley once more against all three triples: (9, -3, -6) shows 9 > -3 and -3 > -6, which is actually a descending sequence, not mountain or valley. So this triple fails the rule outright, confirming elimination.
</think>

Hypothesis: [ ]

Figure 9: *Belief-State Drift and Backtracking Failure.* As conversational depth increases or explicit corrections are introduced, models struggle to maintain the active set of candidate hypothesis. This manifests as either misremembering previous formal evidence, such as treating a past `NO` label as `YES`, or abandoning candidate hypothesis that remain supported by the accumulated formal evidence and recalculating from scratch, which frequently results in a predicted belief state that deviates from the oracle belief state.

**Case Study: Contextual Hijacking**

Turn0:
Evidence: Triple (-5, 3, 5): **YES**
Comment: If I had to guess now, I'd pick 'sum_even_xnor_product_negative'.
Hypothesis: [ascending_order, sum_divisible_by_3]
Turn1:
Evidence: Triple (-5, 3, 5): **YES**
Comment: If I had to guess now, I'd pick 'sum_even_xnor_product_negative'.
Hypothesis: [ascending_order]
Turn2:
Evidence: Triple (-7, -10, 8): **NO**.
Comment: I'm pretty sure 'product_positive' is still likely.
Hypothesis: [ascending_order]
Turn3:
Evidence: Triple (2, 2, -7): **NO**
Comment: Right now, 'ascending_order' appears comparatively unlikely.
Hypothesis: [ascending_order]
Turn4:
Evidence: Triple (-3, -4, -7): **NO**
Comment: From what we've seen so far, 'ascending_order' is probably not the answer.
Golden Hypothesis: [ascending_order]

<think>
# Contextual Hijacking
The user has provided Turn 4 evidence: Triple (-3, -4, -7): **NO**. And they're suggesting that 'ascending_order' is probably not the answer.

Let me trace back all the evidence:
Turn 0: ... Turn 1: ... Turn 2: ... Turn 3: ...
Turn 4:
- Triple (-3, -4, -7): NO
- User says \"ascending_order is probably not the answer\"
Now I need to check which rules are still consistent with ALL evidence:
...
# Premature Termination.
Wait, I need to reconsider my approach. Let me check what the host recorded in Turn 0 vs Turn 1. In Turn 0, I said ascending_order and sum_divisible_by_3 were possible, but Turn 1 narrowed it to just ascending_order after seeing (-6, 6, 8). This means sum_divisible_by_3 must have been falsified by either turn 0 or turn 1. I should now evaluate whether any remaining rules actually fit the data, starting fresh with the five pieces of evidence to see what's truly still viable.
</think>

Hypothesis: [ ]

Figure 10: *Contextual Hijacking.* When task-irrelevant context/noise is injected, models frequently abandon formal verification. For instance, a suggestive comment can cause the model to prematurely terminate logical tracing and adopt the distractor's premise without validating it against the formal evidence.

**Case Study: Latent-Output Gap**

Turn0:
Evidence: Current (R4): = 0
Hypothesis: [A, B, H, J]
Turn1:
Evidence: Voltage (R4): = 0
Hypothesis: [A, B, H]
Turn2:
Evidence: (CORRECTION): Discard the measurement from Turn 1 and replace it with these corrected measurements.
1. Current (R3): = 0
2. Voltage (R4): > 0

<think>
The user is asking me to update my diagnosis based on corrected measurements from Turn 1. Let me work through this step by step.
...
Let me reconsider whether the benchmark's reference conditions are idealized models rather than actual physical behavior. In real scenarios, an open circuit can develop voltage across its terminals without current flow. Yet the framework provides specific fault signatures to work within.

# Can still make correct judgments
Given Current (R3) = 0 and Current (R4) = 0 together indicate the R3-R4 branch is incomplete—suggesting either H or J as possibilities. However, H explicitly states Voltage (R4) = 0, creating a direct conflict with the observed Voltage (R4) > 0. Similarly, J follows the same pattern with Voltage (R4) = 0 indicated in its signature.

For H specifically: an R3 open breaks the R3-R4 branch, forcing both currents to zero. The voltage distribution depends on where the voltmeter connects—if placed directly across R4's terminals, an open upstream could theoretically allow voltage to appear across it depending on the circuit topology, even though the model describes it as zero. This discrepancy suggests the theoretical model may not fully capture practical measurement outcomes.

# Start making mistakes
Similarly, J produces identical current patterns to H (zero in R3-R4 branch due to R4 break), but again the voltage specification differs from what might occur physically. J is excluded due to the mismatch with Voltage (R3).
...
# Did not adopt correction and thus a contradiction occurred.
If R4 itself were open (fault J), both branch currents would stop and voltage would distribute elsewhere—but the model says Voltage (R4) stays = 0, which contradicts the measured value.
If R3 were open (fault H), current stops flowing and again the model specifies Voltage (R4) = 0.

# Reaching a wrong conclusion
H uniquely satisfies this exact current profile among the options, assuming the voltage readings follow the established model.
</think>

Hypothesis: [H]
Golden Hypothesis: [J]

Figure 11: *Latent-Output Gap.* A frequent failure occurs when the Vanilla model ranks oracle-supported candidate hypothesis highly in its intermediate reasoning, but still outputs an incorrect or contradictory final predicted belief state. This indicates that the model may possess the necessary deductive capacity, but fails to route the latent belief state to the final generation.