INERTIABENCH: WHEN LLMS ACKNOWLEDGE CORRECTIONS BUT STILL GIVE OLD ANSWERS

Anonymous authors
Paper under double-blind review

ABSTRACT

Large language models (LLMs) have demonstrated impressive capabilities in multi-turn conversations, yet how they handle user corrections remains poorly understood. To investigate this, we perform experiments where a user revises the premises, goal, or constraints after the model answers and later asks the full revised question. We observe an intriguing phenomenon: *models acknowledge a revised premise, goal, or constraint but still give the old answer*, a failure we term **thinking inertia**, which accuracy-based benchmarks conflate with other errors. In this paper, we introduce InertiaBench, a diagnostic benchmark that examines whether models acknowledge a correction, apply it in later answers, and recover when reminded. Concretely, we pair problems from extensive datasets with LLM-generated corrections and standalone revised questions, cross-check both reference answers with distinct LLMs, and manually modify a stratified sample. Finally, we gather 5,996 instances spanning 4 domains and 3 correction types, and responses are classified as the old answer, the correct revised answer, or another error. To evaluate the InertiaBench performance of popular LLMs, we conduct comprehensive experiments on 17 models, illustrating that each model gives the old answer to 18–21% of revised questions, usually after acknowledging the correction, and that 43–77% of these failures persist after a reminder, more often for reasoning models (68%) than standard LLMs (54%). Our code and data are available at https://anonymous.4open.science/r/InertiaBench-4165.

1 INTRODUCTION

Large language models (LLMs) have become capable assistants that follow user instructions over multiple turns (Ouyang et al., 2022; Zheng et al., 2023; Bai et al., 2024). In real-world scenarios, tasks often evolve: users may revise a premise, change the goal, or add a constraint after an initial answer, and the model must carry each correction into its later answers (Tack et al., 2026). However, this remains challenging: LLMs perform 39% worse on average when information is spread across turns (Laban et al., 2025) and may keep following earlier reasoning despite explicit corrections (Chen et al., 2026). Yet these studies mainly report performance drops, leaving a basic question open: *do LLMs actually apply the corrections they acknowledge?*

In the investigation, we analyze this question via controlled multi-turn experiments. Concretely, rather than posing a revised question in isolation as shown in Figure 1(a), we embed the correction in a dialogue as in Figure 1(b): a model first answers an original problem, the user then corrects the task, and the model replies to the correction. After an intervening exchange, the user asks the revised question, restating all corrected conditions. This design is deliberately favorable to the model: the latest request fully specifies the revised task, so answering the original task cannot be explained by missing information. We then examine two responses in each dialogue: the model's reply to the correction and its final answer.

Surprisingly, even under this favorable design, we observe a recurring failure: the model accurately acknowledges the correction in its reply, yet still gives the old answer in its final response. As illustrated in Figure 1(b), the model confirms the new goal and even states the corrected answer 0, but answers 2 again when the revised question is asked. This pattern recurs across model families, including recent reasoning models. Because the model has registered the correction, the failure lies

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

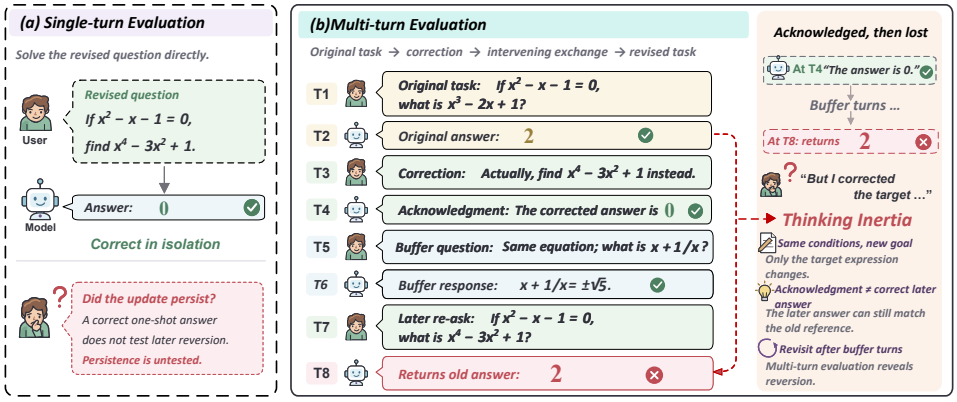


Figure 1: **Thinking Inertia in LLM Conversations.**

not in noticing the change but in applying it. We term this mismatch between acknowledgment and application *thinking inertia*.

Final-answer accuracy alone cannot isolate this failure: it counts an old answer as just another error, ignores the model's reply to the correction, and says nothing about what happens after a reminder. Therefore, we introduce **InertiaBench**, a diagnostic benchmark that tracks each correction through acknowledgment, application, and recovery. The *Corrupt-and-Correct* protocol follows the dialogue design above and evaluates each dialogue in two layers, as shown in Figure 2. Layer 1 scores how accurately the model restates the correction and compares its final answer with the reference answers of both the original and the revised task. For selected dialogues that end with the old answer, Layer 2 adds a follow-up reminder and measures how often the model keeps the old answer or recovers the correct revised answer.

As shown in Figure 3, we build InertiaBench from problems in 6 public reasoning datasets. For each problem, an LLM generates a correction that revises a premise, changes the goal, or adds a constraint, together with a standalone revised question that restates the corrected task and a logically independent intervening question. Each correction must change the reference answer, so every instance pairs an original answer with a distinct revised answer. Two LLMs from different model families then cross-check the reference answers, and we keep only candidates whose revised answer both models confirm. Finally, 3 annotators independently audit a stratified sample, checking that each correction is valid, that the revised question incorporates it correctly, and that the intervening question is independent of the task; any instance rejected by at least two annotators is replaced.

The resulting benchmark contains 5,996 instances, with approximately 500 in each of 12 settings formed by crossing 4 task domains with the 3 correction types. The domains are formal logic, algorithmic/numeric reasoning, constraint-based reasoning, and world knowledge. Because each instance has distinct original and revised answers, every final answer falls into one of 3 outcomes: the old answer, the correct revised answer, or another error. Thinking inertia can thus be distinguished from ordinary mistakes, and linking each outcome to the acknowledgment in the same dialogue shows whether an old answer follows an accurate acknowledgment.

We evaluate standard LLMs together with large reasoning models, as reasoning-oriented training might be expected to make a model more responsive to corrections. We conduct extensive experiments with 17 models of distinct architectures, and results show that all models give the old answer to 18% to 21% of revised questions, where most of the old answers follow an accurate acknowledgment. After a reminder, 43% to 77% of the tested failures persist, depending on the model, and reasoning models keep the old answer more often than standard LLMs, at 68% versus 54% on average. Moreover, in a cross-model study, a request to reason from scratch leaves more old answers than a reminder that restates the correction. These results suggest that reasoning-oriented training alone cannot ensure that a correction is applied. Our contributions are as follows:

- We investigate how LLMs handle user corrections through controlled multi-turn experiments. We find that models accurately acknowledge a correction yet still give the old answer in their final response, a failure we term thinking inertia, as illustrated in Figure 1.

- We design the Corrupt-and-Correct protocol, which scores how accurately a model restates each correction, compares its final answer with both reference answers, and measures how often the old answer survives a reminder, as described in Section 3.1.
- We build and release InertiaBench, which contains 5,996 instances across 4 task domains and 3 correction types. The reference answers of every instance are cross-checked by 2 LLMs, and human annotators audit a stratified sample, as described in Section 3.2.
- We conduct extensive experiments on 17 models from 7 model families. The results show that thinking inertia is widespread, persists after reminders more often in reasoning models, and depends on how the reminder is phrased, as reported in Section 4.

2 Related Work

Reasoning has become a central target of LLM post-training, through process supervision (Lightman et al., 2024), reinforcement learning (Shao et al., 2024; DeepSeek-AI, 2025a; OpenAI, 2024), and inference-time computation (Snell et al., 2025). Another line revises answers after generation. Madaan et al. (2023) refined outputs with self-generated feedback, Kumar et al. (2025) trained self-correction with multi-turn reinforcement learning, and Huang et al. (2024) found that intrinsic self-correction can even reduce reasoning accuracy. Nonetheless, these works improve the solution to a fixed task, ignoring whether a model applies a user correction that changes the task itself.

Multi-turn benchmarks cover dialogue and instruction following (Zheng et al., 2023; Bai et al., 2024; Deshpande et al., 2025) and long-term memory with knowledge updates (Maharana et al., 2024; Wu et al., 2025). Laban et al. (2025) and Tack et al. (2026) found that LLMs degrade when a task is revealed or revised across turns, and that recapping earlier information closes only part of the gap. RefuteBench (Yan et al., 2024; 2025), Meeseeks (Wang et al., 2026), and Inverse IFEval (Zhang et al., 2025) evaluate responses to refutations, feedback, and revised instructions. Nevertheless, none of them separates an old answer after an accurate acknowledgment from other errors, as Table 1 shows.

Chen et al. (2026) described contextual inertia, where models keep following earlier reasoning after explicit corrections, and reduced it with reinforcement learning on single-turn anchors. Knowledge-conflict studies examined how models weigh context against parametric knowledge (Longpre et al., 2021; Xie et al., 2024; Mallen et al., 2023). Sycophancy concerns agreement with users at the expense of truthfulness (Sharma et al., 2024), whereas InertiaBench targets valid revisions that a model should apply. Generated explanations also need not reflect the computation behind an answer (Lanham et al., 2023; Turpin et al., 2023). However, none of these works measures the gap between acknowledging a correction and applying it within the same dialogue.

Table 1: **Evaluation components relevant to thinking inertia.** ✓: explicitly included; △: related analysis with a different measure or setup; —: not separately evaluated.

Benchmark / protocol	Updates	Recogn.	Old ans.	Recovery	Joint
RefuteBench (Yan et al., 2024)	✓	✓	—	△	△
LongMemEval (Wu et al., 2025)	✓	—	—	—	—
MultiChallenge (Deshpande et al., 2025)	✓	—	—	—	—
RefuteBench 2.0 (Yan et al., 2025)	✓	—	—	△	—
Meeseeks (Wang et al., 2026)	—	—	—	✓	—
Evolving intent (Tack et al., 2026)	✓	✓	—	△	△
MT-Refine (RLSTA) (Chen et al., 2026)	✓	—	△	—	—
InertiaBench (this work)	✓	✓	✓	✓	✓

3 InertiaBench

3.1 Corrupt-and-Correct Protocol

As observed in Section 1, a model can accurately acknowledge a correction and still give the old answer. Inspired by this observation, we present InertiaBench, which measures how a correction

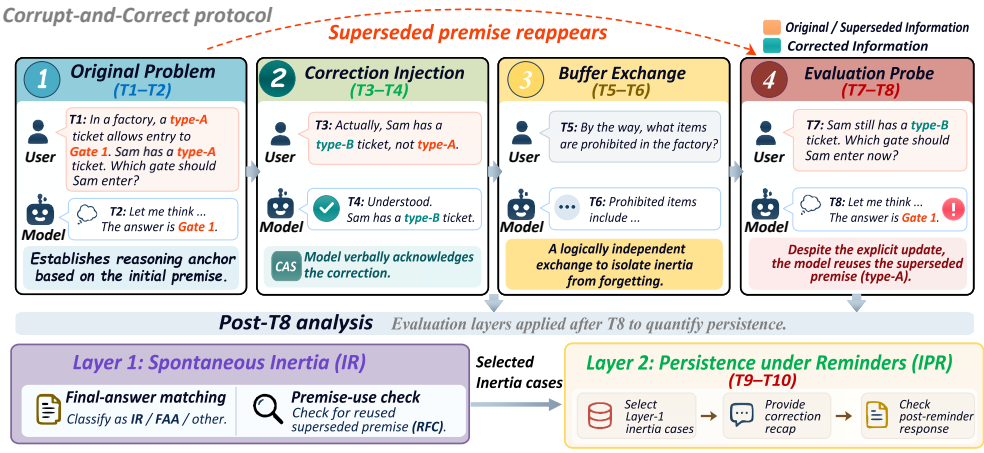


Figure 2: **The Corrupt-and-Correct protocol.** Each dialogue has 4 exchanges: the original problem at T_1 and T_2, the correction at T_3 and T_4, a buffer exchange at T_5 and T_6, and the revised question at T_7 and T_8. Layer 1 scores the acknowledgment at T_4 and classifies the answer at T_8 as the old answer, the correct revised answer, or another error. Layer 2 adds a reminder at T_9 to selected old-answer failures and scores the answer at T_{10}.

is carried from acknowledgment into later answers. Each instance pairs an original problem with a correction and a standalone revised question, and 2 reference answers serve as the evaluation targets: a_{orig} for the original problem and a_{corr} for the revised question. Construction requires $a_{\text{orig}} \neq a_{\text{corr}}$, which allows the evaluation to separate the old answer from the correct revised answer. The original problem remains valid on its own, and the correction only changes what counts as a correct later answer.

Dialogue structure. As shown in Figure 2, each dialogue consists of 4 user and model exchanges, indexed by 8 alternating turns from T_1 to T_8. In the *original-problem exchange*, turns T_1 and T_2, the model solves the original problem and places its answer in the conversation history. In the *correction exchange*, turns T_3 and T_4, the user corrects the task, and the model's reply shows how accurately it acknowledges the correction. The *buffer exchange*, turns T_5 and T_6, asks a topically related question that is logically independent of the task, which separates the correction from the final question while keeping the earlier turns in context. In the *evaluation exchange*, turns T_7 and T_8, the user asks the standalone revised question and the model answers it. Since T_7 restates every corrected condition, the model sees the current task in its latest request and does not need to recover the correction from earlier turns.

Evaluation layers. Layer 1 scores the reply at T_4 and the final answer at T_8, which show how accurately the model acknowledges the correction and whether its later answer follows the revised task. Layer 2 continues selected Layer-1 dialogues whose final answers match a_{orig}: it adds a reminder at T_9 and scores the new answer at T_{10}. It thus measures how often the old answer persists when the model receives another chance to apply the same correction. Section 4.3 specifies the selection and the reminder conditions.

3.2 Dataset Construction and Review

Source tasks. To examine shared and task-specific patterns of thinking inertia, we build instances from 6 source datasets in 4 task groups. **Formal logic (FL)** draws on FOLIO (Han et al., 2024), a first-order logic reasoning dataset, and ReClor (Yu et al., 2020), a logical reading comprehension dataset. **Algorithmic/numeric reasoning (AN)** adopts competition mathematics problems from MATH-500 (Hendrycks et al., 2021). **Constraint-based reasoning (CB)** draws on ZebraLogic (Lin et al., 2025), a logic-grid puzzle benchmark, and LogiQA (Liu et al., 2020), a logical reasoning reading comprehension dataset. **World knowledge (WK)** adopts graduate-level science questions from GPQA Diamond (Rein et al., 2023). These groups describe source coverage rather than mutually exclusive reasoning abilities, and Table 4 in the appendix details each source.

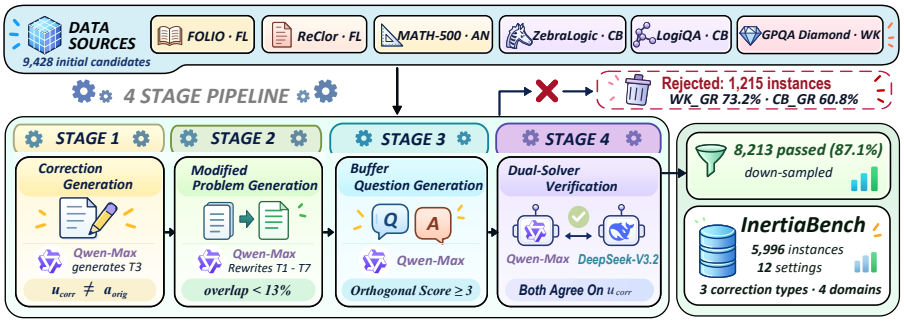


Figure 3: **InertiaBench construction pipeline.** Source problems pass through 4 stages: correction generation, revised-question rewriting, buffer-question generation, and answer verification by 2 LLMs.

Correction types. To vary how a task changes, we target 3 aspects of its specification: the information supplied, the result requested, and the conditions a solution must satisfy. *State refinement* (SR) replaces a premise, fact, or numerical value and keeps the requested goal fixed. *Goal retargeting* (GR) changes the requested result, directing the model to a different question about the same task. *Constraint evolution* (CE) adds a requirement that the revised answer must satisfy together with the earlier conditions that remain applicable. Crossing the 3 correction types with the 4 task groups yields $4 \times 3 = 12$ combinations, which we call *cells*, and FL_SR, for example, denotes state refinement in formal logic. Appendix C gives a worked example of each type. An answer-changing revision must alter the answer content, not merely its wording or option label. The revised reference must follow from the updated task together with the conditions that remain applicable. For multiple-choice items, references are interpreted relative to the answer options in the corresponding task version (Appendix B).

Generation and verification. As shown in Figure 3, construction proceeds in 4 stages. First, Qwen-Max (Alibaba Cloud, 2026) generates a correction of the target type together with a candidate revised answer, and checks that this answer differs from a_{orig}. Second, it rewrites the original problem into a standalone revised question T_7 that incorporates the correction, which also makes the revised task available as a single-turn problem. Third, it generates a buffer question T_5 under instructions to keep it logically independent of the task. Finally, Qwen-Max and DeepSeek-V3.2 (DeepSeek-AI, 2025b) check the candidate reference answers, and we keep a candidate only when both models agree on the revised answer. Because Qwen-Max also generates the candidates, this stage combines the generator's own check with a second model from another family. We then sample from the accepted candidates with a target of 500 instances per cell, yielding 5,996 instances in total. Candidate filtering, retained construction records, and valid model evaluations are different counting units. Appendix J.3 reports the retained construction snapshots and log events, whereas Appendix I reports per-model valid Layer-1 evaluation counts.

Human audit. We asked 3 researchers to independently review a stratified sample of 1,200 instances, with 100 drawn from each cell, without access to the evaluated models' responses. They checked 3 criteria: the validity of each correction, its correct incorporation into T_7, and the logical independence of T_5 from the task. Mean pairwise agreement on these criteria was 87.6%, 93.2%, and 90.8%, respectively. These values measure agreement between annotators, not instance acceptance rates. We removed every instance that at least 2 annotators rejected on any criterion and replaced it with a candidate from the accepted pool. This audit concerns the constructed tasks, whereas model responses are scored with the metrics below.

3.3 Evaluation Metrics

We score the 3 observable outcomes of the protocol: the acknowledgment at T_4, the answer to the revised question at T_8, and the answer after a reminder at T_{10}. All rates are computed per model, and a diagnostic of the task information in the visible T_8 response complements them.

Final-answer outcomes at T_8. For a model evaluated on N valid dialogues, let $o_i \in \{0, 1\}$ indicate whether its final answer in $T_{8,i}$ matches $a_{\text{orig},i}$, and let $u_i \in \{0, 1\}$ indicate whether it matches $a_{\text{corr},i}$.

Following the LLM-as-a-judge approach (Zheng et al., 2023), we employ DeepSeek-V3.2 (DeepSeek-AI, 2024) to extract the final answer and compare it with both references while ignoring intermediate reasoning steps, as detailed in Appendix D. The *inertia rate* (IR) and *final-answer accuracy* (FAA) are

$$\mathrm{IR} = \frac{1}{N}\sum_{i=1}^{N} o_i, \qquad \mathrm{FAA} = \frac{1}{N}\sum_{i=1}^{N} u_i, \qquad \mathrm{IR}, \mathrm{FAA} \in [0,1]. \tag{1}$$

IR counts final answers that match the old answer, whereas FAA counts correct answers to the revised question. Both rates rely on the benchmark references rather than on the model's own answer at T_2, and attributing an old answer to the earlier dialogue would require a matched comparison without that history. The *both-fail* rate (BF) counts answers that match neither reference, $\mathrm{BF} = N^{-1}\sum_{i=1}^{N} \mathbf{1}[o_i = 0 \wedge u_i = 0]$, where $\mathbf{1}[\cdot]$ is the indicator function. Since the 2 references differ, $\mathrm{IR} + \mathrm{FAA} + \mathrm{BF} = 1$, and a double match would signal a reference or scoring conflict. Reporting all 3 outcomes separates a successful revision from merely replacing the old answer with another error, and Table 2 reports them as percentages.

Acknowledgment at T_4 and subsequent failure. The *correction acknowledgment score* (CAS) rates how accurately the model's reply at T_4 represents the correction on 5 ordered levels. Levels 1 to 3 cover no mention of the correction, a verbal confirmation without restating it, and an incomplete or inaccurate restatement. Levels 4 and 5 cover a complete restatement and an accurate restatement of both the correction and its implications. Hence $\mathrm{CAS} \geq 4$ marks a substantive acknowledgment rather than a generic confirmation. CAS characterizes the correction as expressed in the visible response at T_4; it does not by itself establish that the correction is applied in the final answer at T_8. Linking CAS to the later answer therefore distinguishes substantive acknowledgment from successful task revision. To identify old answers that follow such an acknowledgment, we define the *acknowledge-but-not-update* rate (AnU) on a common set of jointly labeled dialogues:

$$\mathrm{AnU} = \frac{1}{N}\sum_{i=1}^{N} \mathbf{1}[\mathrm{CAS}_i \geq 4] o_i, \qquad 0 \leq \mathrm{AnU} \leq \mathrm{IR}. \tag{2}$$

Its denominator includes all N dialogues, hence, AnU measures the joint occurrence of an accurate acknowledgment and an old answer, rather than a failure rate conditioned on acknowledgment. Unlike a comparison of mean CAS with aggregate IR, AnU links the 2 events within the same dialogue.

Persistence after a reminder at T_{10}. For a given model and reminder condition, let $\mathcal{S}$ be the nonempty set of selected Layer-1 old-answer failures evaluated in Layer 2, with $o_i = 1$ for every $i \in \mathcal{S}$. Let o_i^{rem} indicate whether the final answer in $T_{10,i}$ still matches $a_{\mathrm{orig},i}$. The *inertia persistence rate* (IPR) is

$$\mathrm{IPR} = \frac{1}{|\mathcal{S}|}\sum_{i \in \mathcal{S}} o_i^{\mathrm{rem}}, \qquad \mathrm{IPR} \in [0,1]. \tag{3}$$

IPR measures how often the old answer persists on the selected failures, and the reminder comparisons in Section 4.3 also call this outcome *still-IR*. A response counts as a correct recovery only when it matches a_{corr}, and responses that match neither reference remain incorrect, hence $1 - \mathrm{IPR}$ is not the recovery rate. Since the selected failure sets differ across models, cross-model IPR comparisons describe persistence on those sets.

4 Experimental Setup and Results

Our experiments use the Corrupt-and-Correct protocol (Figure 2) to examine whether models apply acknowledged corrections and how old-answer failures respond to reminders. After describing the experimental setup (Section 4.1), we report final-answer outcomes and their relationship to acknowledgment, followed by a breakdown across domains and correction types (Section 4.2). We then examine persistence after reminders and compare reminder formulations (Section 4.3).

4.1 Experimental Setup

We evaluate 17 configurations on InertiaBench, comprising six categorized as large reasoning models (LRMs) and eleven comparison LLMs (Table 2). The evaluation covers 12 cells formed by crossing

Table 2: **Model-level answer outcomes on InertiaBench (%).** IR/FAA describe Layer 1; IPR* measures post-reminder persistence. Darker/lighter purple marks best/second-best displayed model values (ties included), with the best in bold. *IPR uses model-specific selected failures under information recap; its summaries are unweighted means of the displayed model rates.

Model	Family	Type	IR↓	FAA↑	IPR*↓
● QwQ-Plus	Alibaba	LRM	21.3%	34.5%	75%
● QwQ-32B	Alibaba	LRM	20.8%	34.5%	77%
● Qwen-Plus	Alibaba	LLM	20.0%	35.9%	62%
● Qwen-Max	Alibaba	LLM	18.6%	**39.9%**	46%
● Qwen2.5-72B	Alibaba	LLM	18.4%	38.8%	44%
● Qwen2.5-7B	Alibaba	LLM	**17.6%**	35.8%	57%
◆ Gemini-2.5-Pro	Google	LLM	21.3%	33.1%	52%
◆ Gemini-2.5-Flash	Google	LLM	20.1%	33.8%	57%
■ Claude-Opus-4.6	Anthropic	LLM	21.2%	35.3%	69%
■ Claude-4.6-Sonnet	Anthropic	LLM	19.5%	34.6%	63%
★ DeepSeek-R1	DeepSeek	LRM	19.1%	35.6%	61%
★ DeepSeek-V3	DeepSeek	LLM	18.8%	39.1%	**43%**
● GLM-4.7	Zhipu AI	LLM	19.7%	39.1%	55%
● GLM-5	Zhipu AI	LRM	18.7%	34.3%	72%
▲ GPT-4.1	OpenAI	LLM	18.5%	37.2%	47%
▲ o4-mini	OpenAI	LRM	18.1%	32.2%	59%
⊙ MiniMax-M2.5	MiniMax	LRM	18.5%	33.9%	65%

LRM average			LLM average			Overall		
IR↓	FAA↑	IPR*↓	IR↓	FAA↑	IPR*↓	IR↓	FAA↑	IPR*↓
19.4%	34.2%	68.2%	19.4%	36.7%	54.1%	**19.4%**	**35.8%**	**59.1%**

four task domains with three correction types (Section 3.2). Each dialogue follows the protocol in Section 3.1: the update is introduced at T_3 and incorporated into T_7; T_4 records acknowledgment, T_8 answers the revised task, and T_9–T_{10} provide a reminder and response on selected failures. We use the evaluation metrics defined in Section 3.3. To complement answer matching, the Reasoning Fidelity Check (RFC) uses three cross-family LLM judges to label original and updated task information in the visible T_8 response (Appendix D). RFC labels characterize task information in the visible response and are separate from CAS and final-answer matching. They complement these measures without establishing the model's internal reasoning process. Valid Layer-1 evaluation counts are reported in Appendix I.

4.2 Correction Outcomes across Models and Task Types

Table 3: Layer-1 cell IR ranges across the three correction types within each task group.

Domain	Source	Cell IR (%)
Formal Logic (FL)	FOLIO, ReClor	24.9–38.3
Algorithmic/Numeric (AN)	MATH-500	2.6–4.2
Constraint-Based (CB)	ZebraLogic, LogiQA	12.8–24.8
World Knowledge (WK)	GPQA Diamond	24.2–28.9

We first assess whether old-reference answers still occur when the revised task is explicitly restated at T_7. As shown in Table 2, the inertia rate (IR) ranges from 17.6% for Qwen2.5-7B to 21.3% for QwQ-Plus and Gemini-2.5-Pro, while final-answer accuracy (FAA) ranges from 32.2% to 39.9%. Both groups have a reported mean IR of 19.4%. Thus, every evaluated configuration sometimes produces the pre-update reference despite receiving the revised conditions in its latest request. Old-answer failures also occur in both Gemini-2.5 variants, with IRs of 21.3% for Pro and 20.1% for Flash. The within-family Qwen2.5 comparison further shows that higher task accuracy does not necessarily coincide with fewer old-answer failures: the 72B model achieves higher FAA than the 7B model (38.8% vs. 35.8%), but also has a higher IR (18.4% vs. 17.6%). Higher revised-task accuracy therefore does not imply a lower old-reference match rate.

Acknowledgment does not ensure updating. Across 15 model groups with joint CAS and answer labels, substantive acknowledgment (CAS $\geq$ 4) averages 93.0%, while IR and AnU are 19.40% and 17.70%, respectively (Figure 5(a)). All three values are equal-weight means of per-model rates computed on the same records within each model. AnU identifies records in which substantive

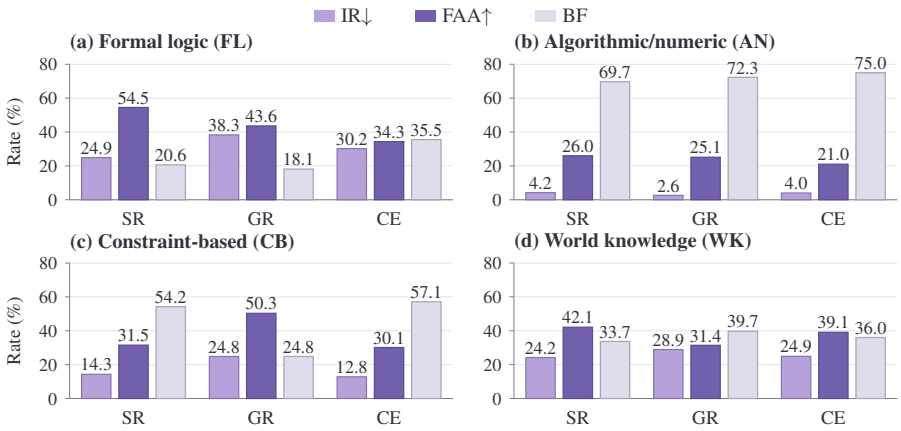


Figure 4: **Cell-level Layer-1 answer outcomes.** IR, FAA, and BF denote old-reference matches, revised-task accuracy, and matches to neither reference. Panels compare SR, GR, and CE on a shared 0–80% scale; labels show reported percentages.

acknowledgment at T_4 is followed by an old-reference answer at T_8, showing that old-answer failures also occur after the update has been accurately restated. Per-model counts and label coverage are reported in Appendix F.

Domain and correction-type differences. Complementing the model-level analysis, we summarize Layer-1 IR across the evaluated models by domain (Table 3) and compare the three update types within each domain (Figure 4). Across SR, GR, and CE, IR ranges from 24.9–38.3% in formal logic (FL), 12.8–24.8% in constraint-based reasoning (CB), and 24.2–28.9% in world knowledge (WK), compared with only 2.6–4.2% in mathematical tasks (AN). GR has the highest aggregate IR in FL, CB, and WK. For example, FL_GR reaches 38.3%, compared with 24.9% for SR and 30.2% for CE: in these formal-logic tasks, changing the requested conclusion is more often followed by an old-reference answer than replacing a premise or adding a constraint. This ordering does not extend to AN, where GR has the lowest IR (2.6%), indicating that the relationship between update type and old-answer matching differs across domains. The accuracy results add a further distinction. Within CB, GR has both the highest IR (24.8%) and the highest FAA (50.3%), alongside a lower BF rate than SR or CE. A higher old-answer rate therefore need not coincide with lower revised-task accuracy. Conversely, AN combines low IR with BF rates of 69.7–75.0% and FAA of only 21.0–26.0%; most responses match neither reference, rather than successfully solve the revised task. These contrasting profiles show why avoiding the old answer and correctly applying the update must be assessed separately. By crossing domains with update types and distinguishing old-reference matches, correct revised answers, and other errors, InertiaBench reveals how the frequency and composition of failures vary with the task and its revision—differences that an overall accuracy or inertia rate alone would conceal. Additional comparisons with history-free single-turn conditions are reported in Appendix K.

4.3 Thinking Inertia after Reminders

We first examine whether Layer-1 old-reference failures persist when the correction is explicitly repeated. For selected T_8 old-reference failures from the nine non-mathematical cells, we repeat the correction T_3 at T_9 and evaluate the response at T_{10}. The resulting IPR describes persistence on these selected failure sets, rather than a failure rate over all benchmark instances. IPR remains 43–77% across the 17 configurations (Table 2), with means of 68.2% for the LRM group and 54.1% for the comparison LLM group. Their similar initial IR is accompanied by different persistence on these model-specific selected failure sets. These results quantify continued old-reference matching, not correct recovery: a response may stop matching a_{orig} while still failing to match a_{corr}.

To examine how persistence varies with the reminder request, a separate four-model study compares three formulations at T_9: repeating the update (A), pointing out that the previous answer is incorrect (B), and requesting a new derivation without repeating the update (C). These requests continue the existing conversation. C does not remove the earlier context. The comparison covers FL_SR, FL_GR, CB_GR, and WK_SR. The reported pooled IPR values are 59.1%, 51.8%, and 87.8% for A, B, and

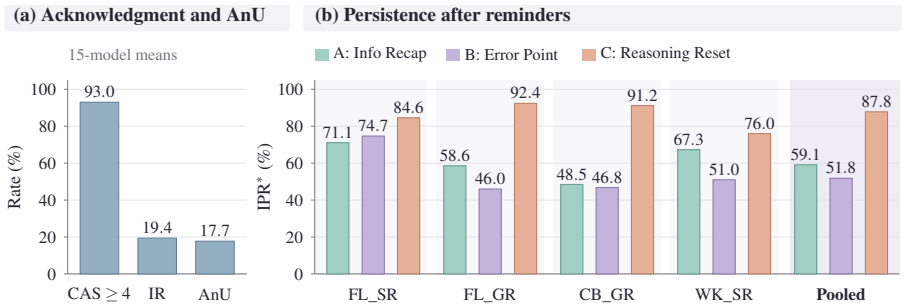


Figure 5: **Acknowledgment and post-reminder persistence (%).** (a) Equal-weight means over 15 model groups on records with joint CAS/answer labels; AnU counts CAS ≥ 4 and old-answer matches in the same record. (b) IPR* is post-reminder *still-IR* on selected failures, by cell and pooled. Bars are ordered A, B, C within each group.

C, respectively (Figure 5(b)). C exceeds A by 28.7 percentage points in the pooled results and has the highest IPR in all four cells. B has a lower pooled IPR than A, but this ordering does not hold in every cell.

5 Discussion and Analysis

InertiaBench makes the gap between acknowledgment and application a measurable dimension of model reliability in multi-turn dialogues. Its joint analysis identifies old-reference answers following substantive acknowledgment in the same dialogue (Section 4.2). Across SR, GR, and CE, correct adaptation requires selective updating of the task specification while retaining still-valid conditions, rather than indiscriminate agreement or erasure of prior context.

The model-level comparisons show why this correction robustness should be evaluated separately from task accuracy. Qwen2.5-72B attains higher FAA than 7B without a lower IR (Table 2). Because FAA, IR, and BF partition the outcomes, an accuracy gain can coexist with increased old-answer matching when other errors decrease. The LRM mean IPR also exceeds the comparison LLM mean on their model-specific selected failure sets. This observations motivates assessing both update application and recovery, rather than inferring correction robustness from scale or reasoning capability.

The reminder comparisons further distinguish re-derivation from explicit correction recap. Both retain the conversation history, but in the four-model study, re-derivation without recap has higher IPR than repeating the correction (Section 4.3). Recovery is judged by correctness under the revised specification, not by departure from the old answer alone. The reminder results also motivate a testable hypothesis: the effect of restarting depends on whether the update is explicitly repeated at the reminder turn. A follow-up comparison of restart requests with and without recap would help examine this hypothesis. InertiaBench thereby evaluates correction across a response trajectory—acknowledgment, application, and recovery—rather than at an isolated statement of acknowledgment.

6 Conclusion

We introduced InertiaBench to evaluate whether language models acknowledge task updates, apply them in subsequent answers, and recover from failures after reminders. Across 17 model configurations, models continue to produce old-reference answers despite explicitly restated task conditions. The joint analysis identifies such answers after substantive acknowledgment in the same dialogue, capturing the mismatch we term thinking inertia. Reasoning models exhibit higher mean post-reminder persistence than comparison LLMs on their model-specific selected failure sets. In the four-model study, old-answer persistence is higher when models are asked to reason from scratch without restating the correction than when the correction is repeated. These findings establish correction robustness as an important evaluation dimension beyond task accuracy. InertiaBench provides a framework for measuring this capability, distinguishing acknowledgment from application and correct recovery from merely abandoning an outdated answer.

AI USE STATEMENT

We used Qwen-Max to generate candidate task corrections and reference answers, buffer questions, and standalone revised questions. We also used LLMs for automated reference-answer checks and response evaluation, including final-answer matching and the Reasoning Fidelity Check (RFC). The construction procedures and prompt templates are documented in Appendix J, and the evaluation procedures are described in Appendix D. Human annotators independently audited a stratified sample of 1,200 dataset instances (Appendix E).

Beyond the benchmark pipeline, we used LLM-based tools to assist with literature review, manuscript drafting and organization, language editing, and formatting. These tools also provided feedback on research methodology and the interpretation of results, as well as suggestions for improving figures and textual presentation. As supplementary checks, we used them to help cross-check reported experimental statistics and assess whether manuscript statements were consistent with the corresponding experimental data or project code. These checks were intended to help identify potential errors, inconsistencies, and omissions that might otherwise be overlooked.

we take full responsibility for all final research and editorial decisions and for the accuracy, originality, and integrity of this work, including AI-assisted content.

REFERENCES

Alibaba Cloud. Alibaba Cloud Model Studio: Text generation. Official documentation, 2026. URL https://www.alibabacloud.com/help/en/model-studio/text-generation-model. Last updated September 24, 2026. Accessed September 26, 2026.

Ge Bai, Jie Liu, Xingyuan Bu, Yancheng He, Jiaheng Liu, Zhanhui Zhou, Zhuoran Lin, Wenbo Su, Tiezheng Ge, Bo Zheng, and Wanli Ouyang. Mt-bench-101: A fine-grained benchmark for evaluating large language models in multi-turn dialogues. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2024, Bangkok, Thailand, August 11-16, 2024*, pp. 7421–7454. Association for Computational Linguistics, 2024. doi: 10.18653/V1/2024.ACL-LONG.401. URL https://doi.org/10.18653/v1/2024.acl-long.401.

Xingwu Chen, Zhanqiu Zhang, Steven Y. Guo, and Difan Zou. Breaking contextual inertia: Reinforcement learning with single-turn anchors for stable multi-turn interaction. In Maria Liakata, Viviane P. Moreira, Jiajun Zhang, and David Jurgens (eds.), *Findings of the Association for Computational Linguistics, ACL 2026, San Diego, California, United States, July 2-7, 2026*, pp. 6302–6318. Association for Computational Linguistics, 2026. doi: 10.18653/V1/2026.FINDINGS-ACL.313. URL https://doi.org/10.18653/v1/2026.findings-acl.313.

DeepSeek-AI. Deepseek-v3 technical report. *CoRR*, abs/2412.19437, 2024. doi: 10.48550/ARXIV.2412.19437. URL https://doi.org/10.48550/arXiv.2412.19437.

DeepSeek-AI. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *CoRR*, abs/2501.12948, 2025a. doi: 10.48550/ARXIV.2501.12948. URL https://doi.org/10.48550/arXiv.2501.12948.

DeepSeek-AI. Deepseek-v3.2: Pushing the frontier of open large language models. *CoRR*, abs/2512.02556, 2025b. doi: 10.48550/ARXIV.2512.02556. URL https://doi.org/10.48550/arXiv.2512.02556.

Kaustubh Deshpande, Ved Sirdeshmukh, Johannes Baptist Mols, Lifeng Jin, Ed-Yeremai Hernandez-Cardona, Dean Lee, Jeremy Kritz, Willow E. Primack, Summer Yue, and Chen Xing. Multichallenge: A realistic multi-turn conversation evaluation benchmark challenging to frontier llms. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Findings of the Association for Computational Linguistics, ACL 2025, Vienna, Austria, July 27 - August 1, 2025*, volume ACL 2025 of *Findings of ACL*, pp. 18632–18702. Association for Computational Linguistics, 2025. doi: 10.18653/V1/2025.FINDINGS-ACL.958. URL https://doi.org/10.18653/v1/2025.findings-acl.958.

Simeng Han, Hailey Schoelkopf, Yilun Zhao, Zhenting Qi, Martin Riddell, Wenfei Zhou, James Coady, David Peng, Yujie Qiao, Luke Benson, Lucy Sun, Alexander Wardle-Solano, Hannah Szabó, Ekaterina Zubova, Matthew Burtell, Jonathan Fan, Yixin Liu, Brian Wong, Malcolm Sailor, Ansong Ni, Linyong Nan, Jungo Kasai, Tao Yu, Rui Zhang, Alexander R. Fabbri, Wojciech Kryscinski, Semih Yavuz, Ye Liu, Xi Victoria Lin, Shafiq Joty, Yingbo Zhou, Caiming Xiong, Rex Ying, Arman Cohan, and Dragomir Radev. FOLIO: natural language reasoning with first-order logic. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (eds.), *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing, EMNLP 2024, Miami, FL, USA, November 12-16, 2024*, pp. 22017–22031. Association for Computational Linguistics, 2024. doi: 10.18653/V1/2024.EMNLP-MAIN.1229. URL https://doi.org/10.18653/v1/2024.emnlp-main.1229.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the MATH dataset. In Joaquin Vanschoren and Sai-Kit Yeung (eds.), *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks 1, NeurIPS Datasets and Benchmarks 2021, December 2021, virtual*, 2021. URL https://datasets-benchmarks-proceedings.neurips.cc/paper/2021/hash/be83ab3ecd0db773eb2dc1b0a17836a1-Abstract-round2.html.

Jie Huang, Xinyun Chen, Swaroop Mishra, Huaixiu Steven Zheng, Adams Wei Yu, Xinying Song, and Denny Zhou. Large language models cannot self-correct reasoning yet. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024. URL https://openreview.net/forum?id=IkmD3fKBPQ.

Aviral Kumar, Vincent Zhuang, Rishabh Agarwal, Yi Su, John D. Co-Reyes, Avi Singh, Kate Baumli, Shariq Iqbal, Colton Bishop, Rebecca Roelofs, Lei M. Zhang, Kay McKinney, Disha Shrivastava, Cosmin Paduraru, George Tucker, Doina Precup, Feryal M. P. Behbahani, and Aleksandra Faust. Training language models to self-correct via reinforcement learning. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025. URL https://openreview.net/forum?id=CjwERcAU7w.

Philippe Laban, Hiroaki Hayashi, Yingbo Zhou, and Jennifer Neville. Llms get lost in multi-turn conversation. *CoRR*, abs/2505.06120, 2025. doi: 10.48550/ARXIV.2505.06120. URL https://doi.org/10.48550/arXiv.2505.06120.

Tamera Lanham, Anna Chen, Ansh Radhakrishnan, Benoit Steiner, Carson Denison, Danny Hernandez, Dustin Li, Esin Durmus, Evan Hubinger, Jackson Kernion, Kamile Lukosiute, Karina Nguyen, Newton Cheng, Nicholas Joseph, Nicholas Schiefer, Oliver Rausch, Robin Larson, Sam McCandlish, Sandipan Kundu, Saurav Kadavath, Shannon Yang, Thomas Henighan, Timothy Maxwell, Timothy Telleen-Lawton, Tristan Hume, Zac Hatfield-Dodds, Jared Kaplan, Jan Brauner, Samuel R. Bowman, and Ethan Perez. Measuring faithfulness in chain-of-thought reasoning. *CoRR*, abs/2307.13702, 2023. doi: 10.48550/ARXIV.2307.13702. URL https://doi.org/10.48550/arXiv.2307.13702.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024. URL https://openreview.net/forum?id=v8L0pN6EOi.

Bill Yuchen Lin, Ronan Le Bras, Kyle Richardson, Ashish Sabharwal, Radha Poovendran, Peter Clark, and Yejin Choi. Zebralogic: On the scaling limits of llms for logical reasoning. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), *Forty-second International Conference on Machine Learning, ICML 2025, Vancouver, BC, Canada, July 13-19, 2025*, volume 267 of *Proceedings of Machine Learning Research*. PMLR / OpenReview.net, 2025. URL https://proceedings.mlr.press/v267/lin25i.html.

Jian Liu, Leyang Cui, Hanmeng Liu, Dandan Huang, Yile Wang, and Yue Zhang. Logiqa: A challenge dataset for machine reading comprehension with logical reasoning. In Christian Bessiere

(ed.), *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI 2020*, pp. 3622–3628. ijcai.org, 2020. doi: 10.24963/IJCAI.2020/501. URL https://doi.org/10.24963/ijcai.2020/501.

Shayne Longpre, Kartik Perisetla, Anthony Chen, Nikhil Ramesh, Chris DuBois, and Sameer Singh. Entity-based knowledge conflicts in question answering. In Marie-Francine Moens, Xuanjing Huang, Lucia Specia, and Scott Wen-tau Yih (eds.), *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 7-11 November, 2021*, pp. 7052–7063. Association for Computational Linguistics, 2021. doi: 10.18653/V1/2021.EMNLP-MAIN.565. URL https://doi.org/10.18653/v1/2021.emnlp-main.565.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. Self-refine: Iterative refinement with self-feedback. In Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine (eds.), *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023. URL http://papers.nips.cc/paper_files/paper/2023/hash/91edff07232fb1b55a505a9e9f6c0ff3-Abstract-Conference.html.

Adyasha Maharana, Dong-Ho Lee, Sergey Tulyakov, Mohit Bansal, Francesco Barbieri, and Yuwei Fang. Evaluating very long-term conversational memory of LLM agents. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2024, Bangkok, Thailand, August 11-16, 2024*, pp. 13851–13870. Association for Computational Linguistics, 2024. doi: 10.18653/V1/2024.ACL-LONG.747. URL https://doi.org/10.18653/v1/2024.acl-long.747.

Alex Mallen, Akari Asai, Victor Zhong, Rajarshi Das, Daniel Khashabi, and Hannaneh Hajishirzi. When not to trust language models: Investigating effectiveness of parametric and non-parametric memories. In Anna Rogers, Jordan L. Boyd-Graber, and Naoaki Okazaki (eds.), *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2023, Toronto, Canada, July 9-14, 2023*, pp. 9802–9822. Association for Computational Linguistics, 2023. doi: 10.18653/V1/2023.ACL-LONG.546. URL https://doi.org/10.18653/v1/2023.acl-long.546.

OpenAI. Learning to reason with LLMs. OpenAI website, September 2024. URL https://openai.com/index/learning-to-reason-with-llms/. Published September 12, 2024. Accessed September 26, 2026.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul F. Christiano, Jan Leike, and Ryan Lowe. Training language models to follow instructions with human feedback. In Sanmi Koyejo, S. Mohamed, A. Agarwal, Danielle Belgrave, K. Cho, and A. Oh (eds.), *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, 2022. URL http://papers.nips.cc/paper_files/paper/2022/hash/b1efde53be364a73914f58805a001731-Abstract-Conference.html.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R. Bowman. GPQA: A graduate-level google-proof q&a benchmark. *CoRR*, abs/2311.12022, 2023. doi: 10.48550/ARXIV.2311.12022. URL https://doi.org/10.48550/arXiv.2311.12022.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *CoRR*, abs/2402.03300, 2024. doi: 10.48550/ARXIV.2402.03300. URL https://doi.org/10.48550/arXiv.2402.03300.

Mrinank Sharma, Meg Tong, Tomasz Korbak, David Duvenaud, Amanda Askell, Samuel R. Bowman, Esin Durmus, Zac Hatfield-Dodds, Scott R. Johnston, Shauna Kravec, Timothy Maxwell, Sam McCandlish, Kamal Ndousse, Oliver Rausch, Nicholas Schiefer, Da Yan, Miranda Zhang, and Ethan Perez. Towards understanding sycophancy in language models. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024. URL https://openreview.net/forum?id=tvhaxkMKAn.

Charlie Victor Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling LLM test-time compute optimally can be more effective than scaling parameters for reasoning. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025. URL https://openreview.net/forum?id=4FWAwZtd2n.

Jihoon Tack, Philippe Laban, and Jennifer Neville. Llms get lost in evolving user intent. *CoRR*, abs/2607.20734, 2026. doi: 10.48550/ARXIV.2607.20734. URL https://doi.org/10.48550/arXiv.2607.20734.

Miles Turpin, Julian Michael, Ethan Perez, and Samuel R. Bowman. Language models don't always say what they think: Unfaithful explanations in chain-of-thought prompting. In Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine (eds.), *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023. URL http://papers.nips.cc/paper_files/paper/2023/hash/ed3fea9033a80fea1376299fa7863f4a-Abstract-Conference.html.

Jiaming Wang, Yunke Zhao, Peng Ding, Jun Kuang, Yibin Shen, Zhe Tang, Yilin Jin, Zongyu Wang, Xiaoyu Li, and Xuezhi Cao. Meeseeks: A feedback-driven, iterative self-correction benchmark evaluating llms' instruction following capability. In Maria Liakata, Viviane P. Moreira, Jiajun Zhang, and David Jurgens (eds.), *Findings of the Association for Computational Linguistics, ACL 2026, San Diego, California, United States, July 2-7, 2026*, pp. 14745–14773. Association for Computational Linguistics, 2026. doi: 10.18653/V1/2026.FINDINGS-ACL.725. URL https://doi.org/10.18653/v1/2026.findings-acl.725.

Di Wu, Hongwei Wang, Wenhao Yu, Yuwei Zhang, Kai-Wei Chang, and Dong Yu. Longmemeval: Benchmarking chat assistants on long-term interactive memory. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025. URL https://openreview.net/forum?id=pZiyCaVuti.

Jian Xie, Kai Zhang, Jiangjie Chen, Renze Lou, and Yu Su. Adaptive chameleon or stubborn sloth: Revealing the behavior of large language models in knowledge conflicts. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024. URL https://openreview.net/forum?id=auKAUJZMO6.

Jianhao Yan, Yun Luo, and Yue Zhang. Refutebench: Evaluating refuting instruction-following for large language models. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Findings of the Association for Computational Linguistics, ACL 2024, Bangkok, Thailand and virtual meeting, August 11-16, 2024*, volume ACL 2024 of *Findings of ACL*, pp. 13775–13791. Association for Computational Linguistics, 2024. doi: 10.18653/V1/2024.FINDINGS-ACL.818. URL https://doi.org/10.18653/v1/2024.findings-acl.818.

Jianhao Yan, Yun Luo, and Yue Zhang. Refutebench 2.0 - agentic benchmark for dynamic evaluation of LLM responses to refutation instruction. *CoRR*, abs/2502.18308, 2025. doi: 10.48550/ARXIV.2502.18308. URL https://doi.org/10.48550/arXiv.2502.18308.

Weihao Yu, Zihang Jiang, Yanfei Dong, and Jiashi Feng. Reclor: A reading comprehension dataset requiring logical reasoning. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020. URL https://openreview.net/forum?id=HJgJtT4tvB.

Aohan Zeng, Bin Xu, Bowen Wang, Chenhui Zhang, Da Yin, Diego Rojas, Guanyu Feng, Hanlin Zhao, Hanyu Lai, Hao Yu, Hongning Wang, Jiadai Sun, Jiajie Zhang, Jiale Cheng, Jiayi Gui, Jie Tang, Jing Zhang, Juanzi Li, Lei Zhao, Lindong Wu, Lucen Zhong, Mingdao Liu, Minlie Huang, Peng Zhang, Qinkai Zheng, Rui Lu, Shuaiqi Duan, Shudan Zhang, Shulin Cao, Shuxun

Yang, Weng Lam Tam, Wenyi Zhao, Xiao Liu, Xiao Xia, Xiaohan Zhang, Xiaotao Gu, Xin Lv, Xinghan Liu, Xinyi Liu, Xinyue Yang, Xixuan Song, Xunkai Zhang, Yifan An, Yifan Xu, Yilin Niu, Yuantao Yang, Yueyan Li, Yushi Bai, Yuxiao Dong, Zehan Qi, Zhaoyu Wang, Zhen Yang, Zhengxiao Du, Zhenyu Hou, and Zihan Wang. Chatglm: A family of large language models from GLM-130B to GLM-4 all tools. *CoRR*, abs/2406.12793, 2024. doi: 10.48550/ARXIV.2406.12793. URL https://doi.org/10.48550/arXiv.2406.12793.

Qinyan Zhang, Xinping Lei, Ruijie Miao, Yu Fu, Haojie Fan, Le Chang, Jiafan Hou, Dingling Zhang, Zhongfei Hou, Ziqiang Yang, Changxin Pu, Fei Hu, Jingkai Liu, Mengyun Liu, Yang Liu, Xi Qi Gao, Jiaheng Liu, Tong Yang, Zaiyuan Wang, Ge Zhang, and Wenhao Huang. Inverse ifeval: Can llms unlearn stubborn training conventions to follow real instructions? *CoRR*, abs/2509.04292, 2025. doi: 10.48550/ARXIV.2509.04292. URL https://doi.org/10.48550/arXiv.2509.04292.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. Judging llm-as-a-judge with mt-bench and chatbot arena. In Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine (eds.), *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023. URL http://papers.nips.cc/paper_files/paper/2023/hash/91f18a1287b398d378ef22505bf41832-Abstract-Datasets_and_Benchmarks.html.

Contents of Appendix

A More examples from InertiaBench **15**
B Source Tasks **19**
C Update Types Examples **21**
D Implementation Details **24**
E Human Annotation Protocol **25**
F Acknowledgment and Subsequent Old-Answer Matches **26**
G Broader Impact **27**
H Domain-Dependent Inertia Pathology and Inertia Spectrum **27**
I Details of the model-level experiment **28**
J Dataset Construction and Prompt Templates **28**
J.1 Source routes, settings and validation behavior 30
J.2 Construction prompt templates 30
J.3 Retained snapshots and log accounting 36
K History-Free Baselines and Multi-Turn Accuracy **37**
L Limitations and Future Work **38**

A More examples from InertiaBench

Figures 6– 9 present revised construction examples covering the four task domains. Each example shows the specified user turns, the original and revised reference answers, and a reference derivation. These derivations explain the task construction rather than report evaluated-model responses.

A new query under unchanged logical rules
WORKED DOMAIN 1 / Formal logic (FL–GR)

Initial task (T_1)

Use only these premises about one training algorithm A:
- A belongs to at least one of these categories: supervised learning, unsupervised learning, or reinforcement learning.
- Every unsupervised-learning algorithm does not require labeled data.
- A is not a reinforcement-learning algorithm.
- A requires labeled data.

Use classical logic. Answer True if the conclusion follows, False if its negation follows, and Unknown otherwise.
Conclusion: A is a supervised-learning algorithm.

Update (T_3)

Keep every premise unchanged. Instead of evaluating whether A is supervised, evaluate whether A is an unsupervised-learning algorithm.

Intervening question (T_5)

In a separate rule set, all amber tokens are round and token Z is amber. Is token Z round?

Standalone revised task (T_7)

Use only these premises about one training algorithm A:
- A belongs to at least one of these categories: supervised learning, unsupervised learning, or reinforcement learning.
- Every unsupervised-learning algorithm does not require labeled data.
- A is not a reinforcement-learning algorithm.
- A requires labeled data.

Use classical logic. Answer True if the conclusion follows, False if its negation follows, and Unknown otherwise.
Conclusion: A is an unsupervised-learning algorithm.

Derived references $a_{\text{orig}} = \texttt{True} \longrightarrow a_{\text{corr}} = \texttt{False}$

Reference derivation

Let S, U, R, L denote the three categories and the requirement for labels. From $U \Rightarrow \neg L$ and L, infer $\neg U$; then $S \vee U \vee R$ and $\neg R$ entail S. The assignment $(S, U, R, L) = (1, 0, 0, 1)$ is consistent. The separate token answer is Yes.

Figure 6: **Formal logic (FL–GR).** The references are True and False under the same stipulated premises. The update changes the query, not a premise.

Two exact targets under one quadratic constraint
WORKED DOMAIN 2 / Algorithmic/numeric reasoning (AN–GR)

Initial task (T_1)
If $x^2 - x - 1 = 0$, what is the value of $x^3 - 2x + 1$?

Update (T_3)
Actually I need the value of $x^4 - 3x^2 + 1$ instead of $x^3 - 2x + 1$.

Intervening question (T_5)
For a separate calculation, let $h(y) = 2y + 5$. What is $h(3)$?

Standalone revised task (T_7)
If $x^2 - x - 1 = 0$, what is the value of $x^4 - 3x^2 + 1$?

Derived references $a_{\text{orig}} = 2 \longrightarrow a_{\text{corr}} = 0$

Reference derivation
Reduction by the unchanged constraint gives

$$x^2 = x + 1, \qquad x^3 = 2x + 1, \qquad x^4 = 3x + 2.$$

Therefore

$$x^3 - 2x + 1 = (2x + 1) - 2x + 1 = 2,$$
$$x^4 - 3x^2 + 1 = (3x + 2) - 3(x + 1) + 1 = 0.$$

These identities hold for both roots $(1 \pm \sqrt{5})/2$. The old value 2 is not a correct answer to the revised question. The separate calculation gives $h(3) = 11$ and imposes no additional condition on x.

Figure 7: **Algorithmic/numeric reasoning (AN–GR).** The constraint remains $x^2 - x - 1 = 0$, while the requested polynomial changes. Both references are exact and hold for either admissible root.

An added clue restricts the fish's possible locations
WORKED DOMAIN 3 / Constraint-based reasoning (CB–CE)

Initial task (T_1)

Houses 1-4 run left to right. Assign one value per category to each house, using every value once. Properties identify houses; "next to" means adjacent.
Colors: red, blue, green, white. *Pets:* dog, cat, fish, other (distinct).
Drinks: tea, coffee, milk, water. *Hobbies:* painting, reading, gardening, cooking.

1. Tea is in the red house.
2. The dog is next to coffee.
3. Milk is in the blue house.
4. The cat is next to water.
5. Green is immediately left of white.
6. Coffee is in house 1.
7. Painting is in house 4.
8. Reading is next to tea.
9. Water is next to gardening.
10. Cooking is not in the red house.

Give the *complete set* of possible fish-house numbers over all valid arrangements.

Update (T_3)

Add exactly one clue while retaining all earlier clues: the fish is strictly to the right of the red house. Keep the request for the complete set of possible fish-house numbers.

Intervening question (T_5)

In a separate seating puzzle, Ava sits immediately left of Ben, and Ben immediately left of Cy. Who occupies the middle of these three consecutive seats?

Standalone revised task (T_7)

Houses 1-4 run left to right. Assign one value per category to each house, using every value once. Properties identify houses; "next to" means adjacent.
Colors: red, blue, green, white. *Pets:* dog, cat, fish, other (distinct).
Drinks: tea, coffee, milk, water. *Hobbies:* painting, reading, gardening, cooking.

1. Tea is in the red house.
2. The dog is next to coffee.
3. Milk is in the blue house.
4. The cat is next to water.
5. Green is immediately left of white.
6. Coffee is in house 1.
7. Painting is in house 4.
8. Reading is next to tea.
9. Water is next to gardening.
10. Cooking is not in the red house.
11. The fish is strictly to the right of the red house.

Give the *complete set* of possible fish-house numbers over all valid arrangements.

Derived references $a_{\text{orig}} = \{1, 3, 4\} \longrightarrow a_{\text{corr}} = \{4\}$

Reference derivation

Enumeration yields eight initial arrangements, with fish locations $\{1, 3, 4\}$. The new clue retains two arrangements, both with red in house 3 and fish in house 4. The separate seating answer is Ben.

Figure 8: **Constraint-based reasoning (CB–CE).** The added clue narrows the complete answer set from $\{1, 3, 4\}$ to $\{4\}$ without replacing an earlier clue.

From a mixed preparation to a specified pure state
WORKED DOMAIN 4 / World knowledge (WK–SR)

Initial task (T_1)

A qubit is prepared in the state

$$\rho = \tfrac{1}{2}\big(|0\rangle\langle 0| + |1\rangle\langle 1|\big) = I/2.$$

Use the computational basis $|0\rangle, |1\rangle$ with $Z|0\rangle = |0\rangle$ and $Z|1\rangle = -|1\rangle$. The Bloch-vector convention is $\rho = (I + r_x X + r_y Y + r_z Z)/2$. What is its Bloch vector?

Update (T_3)

Correct the preparation: the state is $|0\rangle\langle 0|$, not the equal incoherent mixture. Keep the basis, convention and Bloch-vector question unchanged.

Intervening question (T_5)

For a separate three-qubit register, how many computational-basis states are there?

Standalone revised task (T_7)

A qubit is prepared in the state $\rho = |0\rangle\langle 0|$. Use the computational basis $|0\rangle, |1\rangle$ with $Z|0\rangle = |0\rangle$ and $Z|1\rangle = -|1\rangle$. The Bloch-vector convention is $\rho = (I + r_x X + r_y Y + r_z Z)/2$. What is its Bloch vector?

Derived references $a_{\text{orig}} = (0,0,0) \longrightarrow a_{\text{corr}} = (0,0,1)$

Reference derivation

In the ordered computational basis,

$$\rho_{\text{orig}} = \begin{pmatrix} 1/2 & 0 \\ 0 & 1/2 \end{pmatrix} = I/2, \qquad \rho_{\text{corr}} = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} = (I + Z)/2.$$

The respective Pauli coefficients are $(0,0,0)$ and $(0,0,1)$. Both matrices have unit trace and nonnegative eigenvalues. Merely replacing the mixed-state ensemble by an equal mixture of $|+\rangle$ and $|-\rangle$ would leave $I/2$ unchanged; that is not the update used here. The separate register has $2^3 = 8$ basis states.

Convention reference: IBM Quantum Learning, "Bloch sphere".

Figure 9: **World knowledge (WK–SR).** Replacing the mixed-state preparation with the specified pure state changes the Bloch vector from the center to the positive z pole. Both states are physical; the update changes the input, not a law of quantum mechanics.

B Source Tasks

InertiaBench uses six source datasets to examine task revision under different problem structures. The four groups in Table 4 vary both what is needed to answer a question and how correctness is determined: consequences of stated premises, derived mathematical quantities, assignments and conclusions constrained by multiple relations, and answers that require scientific knowledge. This coverage allows the same update protocol to be studied beyond a single question format. The sources supply the initial problems; InertiaBench adds explicit revisions and follows whether they are acknowledged, applied, and acted on after a reminder.

Formal logic (FL). FOLIO provides natural-language premises and conclusions with parallel first-order logic annotations (Han et al., 2024). Its *True*, *False*, and *Unknown* labels distinguish entailment, contradiction, and insufficient information. A premise update can therefore change whether a conclusion follows, rather than merely replace one answer word with another. Losing support for a conclusion need not establish its negation: correct updating may instead require a

Table 4: **Source datasets and task groups in InertiaBench.** MATH-500 and GPQA Diamond are subsets of MATH and GPQA, respectively. The mapping organizes source coverage rather than disjoint reasoning abilities. Each group is crossed with SR, GR, and CE.

Group	Source	Source task
FL	FOLIO (Han et al., 2024)	Natural-language first-order logical inference
FL	ReClor (Yu et al., 2020)	Passage-based logical reading comprehension
AN	MATH-500 (Hendrycks et al., 2021; Lightman et al., 2024)	Free-response mathematical problem solving
CB	ZebraLogic (Lin et al., 2025)	Logic-grid constraint satisfaction
CB	LogiQA (Liu et al., 2020)	Reasoning about conditions and natural-language arguments
WK	GPQA Diamond (Rein et al., 2023)	Graduate-level scientific multiple-choice questions

transition from a definite label to *Unknown*. ReClor extends the coverage to passage-based arguments, with questions about assumptions, implications, and the evaluation of arguments (Yu et al., 2020). Together, these sources support examining whether an answer is reconsidered when its supporting information or the requested conclusion changes.

Algorithmic/numeric reasoning (AN). AN is instantiated here by mathematical problem solving. MATH-500 is the 500-problem subset of the MATH test set used by Lightman et al. (2024); the parent dataset contains competition problems with worked solutions (Hendrycks et al., 2021). Unlike selecting a truth label or an answer option, these tasks require deriving a numerical or symbolic result. Replacing a coefficient can change several dependent quantities, while changing the requested expression requires answering a new question even when the original equations remain valid. Paired references separate returning the superseded result from obtaining a different but still incorrect result. AN thus makes it possible to examine whether avoiding an old answer is accompanied by successful computation of the revised answer, rather than treating a low old-answer rate as evidence of reliable updating.

Constraint-based reasoning (CB). ZebraLogic consists of logic-grid puzzles in which attribute assignments must satisfy a collection of clues and uniqueness requirements (Lin et al., 2025). A revised clue can affect several linked assignments, so a locally plausible change is insufficient unless the answer remains consistent with the whole task. LogiQA complements these explicit constraint systems with passage-based reasoning involving categorical, conditional, conjunctive, and disjunctive relations (Liu et al., 2020). Its inclusion extends the group beyond a single puzzle template. CB examines whether revised relations are applied together with still-valid conditions. For a derived question about one grid attribute, the reference must answer that specific query; the original full-grid solution is not interchangeable with it. A compatible added constraint cannot change a queried value already fixed by the retained conditions; a changed reference must follow from the actual revised task.

World knowledge (WK). GPQA contains expert-written questions in biology, chemistry, and physics; Diamond is its more stringently filtered subset (Rein et al., 2023). WK therefore adds knowledge-intensive scientific reasoning to the evaluation. The model must apply relevant scientific principles to the conditions stated in the problem, and reconsider the answer when a preparation, parameter, or experimental assumption changes. This differs from reasoning solely over an explicit premise list: interpreting the update may itself require domain knowledge. A valid scientific revision changes the specified task without requiring the model to accept an erroneous scientific claim. The scientific validity of a derived question therefore requires checking beyond the reference supplied for its source item.

Source mapping and update construction. The mapping in Table 4 is source-based: FOLIO and ReClor map to FL, MATH-500 to AN, ZebraLogic and LogiQA to CB, and GPQA Diamond to WK. In particular, ReClor and LogiQA overlap in logical reading comprehension; their placement in different groups records the benchmark's organization, not a boundary between formal and non-

formal reasoning. Source membership and update type are separate labels. For an item from source s with update type u, its cell is $(g(s), u)$, where g is the mapping in the table and $u \in \{\mathrm{SR}, \mathrm{GR}, \mathrm{CE}\}$. The source group stays fixed when the update changes a premise, goal, or constraint. This defines twelve combinations at the group level; it does not require every original problem to admit all three kinds of valid, answer-changing revision.

A source item supplies the problem statement, query, and any answer choices for the initial task. Its adaptation is specified by T_1, the update at T_3, and the self-contained revised problem at T_7. The original and revised references concern these two actual tasks, not the model's earlier response at T_2. They are interpreted in the task's answer space: logical status, mathematical result, requested assignment, or selected answer content. For multiple-choice items, a letter identifies an option in a particular question version; reordering options is not itself an answer-changing update. Construction must establish both references and ensure that the old answer is no longer correct under T_7. The subsequent acknowledgment and reminder stages use the same protocol across the four groups.

What the coverage supports. Crossing source groups with update types supports comparisons of revision outcomes across task structures. The groups differ in task difficulty and answer space, so IR is reported alongside FAA and BF. Appendix K provides complementary revised-task accuracy comparisons under history-free presentations.

C Update Types Examples

Table 5 summarizes the three update types, and Figures 10– 12 illustrate SR, GR, and CE using AN, AN, and FL examples, respectively. These are alternative task revisions, not successive dialogue stages. SR replaces a coefficient while retaining the objective; GR changes the requested quantity while retaining the background conditions; and CE adds a premise while retaining the existing premises and query.

Table 5: Summary of the three update types used in InertiaBench. Each type specifies which component of the task is changed and which task-relevant information is retained.

Type	Changed component	Retained component
SR	A supplied fact, value, or clue.	The requested goal and other applicable conditions.
GR	The requested conclusion or quantity.	The background facts and constraints.
CE	An added premise or requirement.	The earlier conditions and the requested goal.

A coefficient replacement with an unchanged objective
WORKED UPDATE 1 / Algorithmic/numeric reasoning (AN–SR)

Initial task (T_1)

For real x, if $|x+5|-|3x-6|=0$, find the largest possible value of x. Express your answer as an improper fraction.

Update (T_3)

Replace the coefficient 3 in the second absolute-value term with 2. Keep all other terms and the request for the largest real solution unchanged.

Intervening question (T_5)

For a separate calculation, let $h(y)=4y+1$. What is $h(5)$?

Standalone revised task (T_7)

For real x, if $|x+5|-|2x-6|=0$, find the largest possible value of x. Express your answer as an improper fraction.

Derived references $a_{\text{orig}} = 11/2 \longrightarrow a_{\text{corr}} = 11/1$

Reference derivation

The two branches are $x+5=\pm(kx-6)$. Hence

$$k=3: \quad x\in\{11/2,\,1/4\}, \qquad k=2: \quad x\in\{11,\,1/3\}.$$

The old largest solution is not a solution of the revised equation: $|11/2+5|=21/2\neq|2(11/2)-6|=5$. The separate calculation gives $h(5)=21$ and imposes no condition on x.

Figure 10: **State refinement (SR).** Replacing one coefficient changes the largest solution from $11/2$ to $11/1$, without changing the goal. The reference derivation is separate from the user messages.

From the complete sum to a specified partial sum
WORKED UPDATE 2 / Algorithmic/numeric reasoning (AN–GR)

Initial task (T_1)

Let a, b, and c be distinct real numbers. Find the degree of

$$p(x) = \frac{(x-b)(x-c)}{(a-b)(a-c)} + \frac{(x-a)(x-c)}{(b-a)(b-c)} + \frac{(x-a)(x-b)}{(c-a)(c-b)}.$$

Update (T_3)

Keep the same three distinct real nodes a, b, and c. Instead of the degree of the complete polynomial $p(x)$, find the degree of $q(x)$, defined as the sum of the first two displayed summands only. Keep their denominators unchanged; do not construct a new interpolating polynomial on two nodes.

Intervening question (T_5)

For a separate calculation, let $h(y) = 3y + 1$. What is $h(2)$?

Standalone revised task (T_7)

Let a, b, and c be distinct real numbers. Find the degree of

$$q(x) = \frac{(x-b)(x-c)}{(a-b)(a-c)} + \frac{(x-a)(x-c)}{(b-a)(b-c)}.$$

Derived references $a_{\text{orig}} = 0 \longrightarrow a_{\text{corr}} = 2$

Reference derivation

The polynomial p has degree at most two and takes the value 1 at three distinct points, so $p(x) \equiv 1$. The coefficient of x^2 in q is

$$\frac{1}{(a-b)(a-c)} + \frac{1}{(b-a)(b-c)} = -\frac{1}{(a-c)(b-c)} \neq 0.$$

Thus $\deg q = 2$ for every admissible choice of nodes. The separate calculation gives $h(2) = 7$ and does not restrict a, b, or c.

Figure 11: **Goal retargeting (GR).** The complete sum has degree 0; the requested partial sum has degree 2. The update preserves all three nodes and the original denominators.

An added fact rules out a previously possible category
WORKED UPDATE 3 / Formal logic (FL–CE)

Initial task (T_1)

Use only the following premises as logical rules; do not import background facts about the named products.
- If a car can transport multiple passengers, it is not a one-seater.
- Every Tesla Model 3 can transport multiple passengers.
- Every single-seat electric car is a one-seater.
- Every Solo EV is a single-seat electric car.

Is the conclusion *John's car is a Solo EV* True, False, or Unknown?

Update (T_3)

Add the following premise while keeping all existing premises and the conclusion unchanged: John's car is a Tesla Model 3.

Intervening question (T_5)

For a separate inventory question, a depot contains 5 sedans and 3 vans, with no other vehicles. How many vehicles are in that depot?

Standalone revised task (T_7)

Use only the following premises as logical rules; do not import background facts about the named products.
- If a car can transport multiple passengers, it is not a one-seater.
- Every Tesla Model 3 can transport multiple passengers.
- Every single-seat electric car is a one-seater.
- Every Solo EV is a single-seat electric car.
- John's car is a Tesla Model 3.

Is the conclusion *John's car is a Solo EV* True, False, or Unknown?

Derived references $a_{\text{orig}} = \texttt{Unknown} \longrightarrow a_{\text{corr}} = \texttt{False}$

Reference derivation

Originally, both a Solo and a non-Solo assignment for John's car are consistent with the rules. After the update, Tesla implies multiple passengers and hence not one-seater. Solo implies single-seat electric and hence one-seater. Therefore John's car cannot be a Solo EV. The separate inventory totals $5 + 3 = 8$ and adds no fact about John's car.

Figure 12: **Constraint evolution (CE).** Appending the Tesla premise changes the unchanged conclusion from Unknown to False. All original rules remain present in the standalone revised task.

D Implementation Details

All models are evaluated via API, with the requested temperature set to 0 to limit sampling variability. Within each domain, all models receive the same short system prompt: "You are a precise logical reasoner" (FL), "You are a precise mathematician" (AN), "You are an expert at constraint satisfaction" (CB), and "You are a knowledgeable scientific reasoning assistant" (WK). FAA and IR are computed using DeepSeek-V3.2 as an answer-matching judge. The judge is instructed to extract the final answer from each T_8 response and compare it with the corresponding reference answers, without treating intermediate reasoning steps as answer candidates. This instruction is intended to avoid mistaking intermediate values or provisional conclusions for the final answer, particularly when LRM responses include extended chain-of-thought (CoT) reasoning.

RFC inter-judge reliability. Inter-judge agreement for the Reasoning Fidelity Check is measured on held-out validation sets from FL_SR and WK_SR :

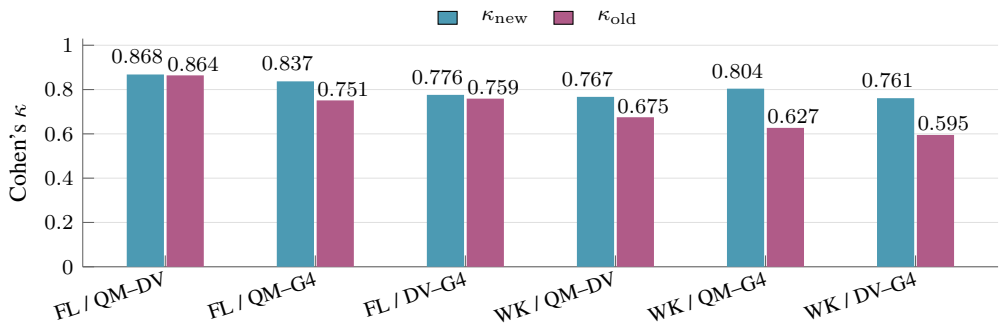


Figure 13: RFC inter-judge Cohen's κ by domain and judge pair. QM: Qwen-Max; DV: DeepSeek-V3; G4: GLM-4. Agreement on the old-information label is lower in WK than agreement on the new-information label.

CoT depth ablation. Manipulating T_2 CoT depth via system prompt across three settings reveals domain-dependent effects: longer CoT reduces IR in FL_SR (25.0% vs. 30.5%) and CB_CE (21.0% vs. 27.5%), but *increases* IR in FL_GR (39.7% vs. 31.2%). This domain-dependence precludes a universal recommendation on CoT length for inertia mitigation.

Task information in the visible response. Whereas answer matching evaluates the final answer, the *reasoning fidelity check* (RFC) characterizes the task information expressed in the visible response at T_8, including any chain-of-thought (CoT) reasoning provided. Under the scoring criteria, $\mathrm{RFC}_{\text{new}} = 1$ when the response applies the update or refers to its new value, and $\mathrm{RFC}_{\text{old}} = 1$ when it contains original task information.

Qwen-Max (Alibaba Cloud, 2026), DeepSeek-V3 (DeepSeek-AI, 2024), and GLM-4 (Zeng et al., 2024) each assign both binary labels, with each final label determined by a majority vote of at least two judges. The pair $(\mathrm{RFC}_{\text{new}}, \mathrm{RFC}_{\text{old}})$ defines four response categories: *Integrated* $(1, 0)$, *Mixed* $(1, 1)$, *Anchored* $(0, 1)$, and *Neither* $(0, 0)$. Agreement among the judges on these RFC labels is assessed on held-out validation sets from FL_SR and WK_SR ($N = 70$ per settings), with pairwise Cohen's κ reported in Figure 13.

These categories describe combinations of textual labels rather than internal reasoning states. A response may mention an old value to reject it, and a new constraint may coexist with still-valid earlier conditions. RFC therefore complements final-answer matching with evidence about task information in the visible response; it does not establish the model's internal reasoning process, consistent with the distinction between generated explanations and reasoning faithfulness (Lanham et al., 2023; Turpin et al., 2023).

E Human Annotation Protocol

Human review involved undergraduate and master's students with an interest in LLM research and research experience relevant to task validation. The review focused on dataset instances and candidate reference answers rather than the responses of evaluated models. Three researchers with graduate-level training in NLP or cognitive science independently audited a stratified sample of 1200 instances (100 per setting, randomly drawn after downsampling). Annotators were blind to model outputs and evaluated only the dataset instances themselves. Each annotator assessed three binary judgments per instance: (1) **Correction validity**: is the injected correction T_3 unambiguous and factually correct, i.e., does a_{corr} differ from a_{orig} for the right reason? (2) T_7 **correctness**: does the modified problem correctly embed the corrected conditions without introducing new ambiguity? (3) **Buffer orthogonality**: is T_5 topically related but logically independent of T_1, such that solving it neither reveals nor primes the corrected answer? We utilize a reconstructed local interface to support structured manual screening. It presents T_1, T_3, T_5, and T_7 together with the candidate original and revised reference answers, while omitting evaluated-model responses and automated judgments. For newly collected annotations, the tool records criterion-wise judgments, uncertainty flags, reviewer notes, pseudonymous reviewer identifiers, timestamps, and task-version hashes. Missing fields are

explicitly flagged rather than filled automatically, and revisions to reviewer judgments are retained as separate events.

To support structured and efficient manual screening, we provide a reconstructed local review workbench that presents all task components required for the three annotation criteria in a unified interface. Specifically, it displays T_1, T_3, T_5, and T_7, together with the candidate original and revised reference answers, while withholding evaluated-model responses and automated judgments from annotators. The runnable version, along with the review rubric and usage instructions, is included in the anonymous supplementary repository that comes with this submission.

F Acknowledgment and Subsequent Old-Answer Matches

Records and aggregation. In Table 6,reports the record-level counts and label coverage underlying Figure 5(a). The analysis uses the retained Layer-1 records, excluding the duplicate checkpoint file. CAS labels copied into subsequent rejudging or Layer-2 files are not counted as additional observations, and controlled-run results are analyzed separately.

For model group m, let N_m be the number of retained records and $\mathcal{V}_m$ the subset with a stored CAS in $\{1, \ldots, 5\}$ and a binary old-reference match label o_i; write $n_m = |\mathcal{V}_m|$. For $n_m > 0$,

$$k_m = \sum_{i \in \mathcal{V}_m} \mathbf{1}[\mathrm{CAS}_i \geq 4] o_i, \qquad \mathrm{AnU}_m = \frac{k_m}{n_m}. \tag{4}$$

The $\mathrm{CAS} \geq 4$ rate and IR use the same n_m. Missing labels are excluded, not treated as negative outcomes. The denominator includes all jointly labeled records, rather than only those with $\mathrm{CAS} \geq 4$; AnU is therefore a joint rate, not an old-answer rate conditioned on acknowledgment. For the $M = 15$ groups with $n_m > 0$, the main text reports

$$\overline{\mathrm{AnU}} = \frac{1}{M} \sum_{m : n_m > 0} \mathrm{AnU}_m, \tag{5}$$

with the same averaging rule for the other two rates. Each group receives equal weight, but settings within a group retain their observed record frequencies.

Joint outcomes. The main text displays these means to one decimal place: 93.0% for $\mathrm{CAS} \geq 4$, 19.4% for IR, and 17.70% for AnU. All averaging precedes display rounding. Pooling all jointly labeled records instead gives $15{,}400/87{,}015 = 17.70\%$ for AnU; this record-weighted rate is distinct from the model-macro mean. AnU links the two labels within a record, rather than inferring their co-occurrence from separate acknowledgment and IR averages.

Scoring provenance and interpretation. CAS evaluates how accurately the visible response at T_4 represents the update at T_3. AnU combines this score with the old-reference match label at T_8 to measure an acknowledgment–answer mismatch within the same record. It is interpreted alongside acknowledgment rates and final-answer outcomes, since a lower AnU can also accompany less frequent acknowledgment.

Table 6: Per-model counts, label coverage, and joint outcomes. Record counts: N_m retained, n_m jointly labeled, and k_m AnU-positive. Coverage is $100n_m/N_m$; all three outcome rates use n_m.

	Record counts			Rates (%)			
Model	N_m	n_m	k_m	Coverage	CAS ≥ 4	IR	AnU
QwQ-Plus	5,996	5,996	1,114	100.00	90.34	21.31	18.58
QwQ-32B	5,883	5,883	1,071	100.00	90.99	20.76	18.20
Qwen-Plus	5,828	5,828	1,087	100.00	95.44	19.97	18.65
Qwen-Max	5,993	5,993	1,094	100.00	98.65	18.55	18.25
Qwen2.5-72B	5,992	5,992	1,084	100.00	98.88	18.36	18.09
Qwen2.5-7B	5,995	5,995	1,017	100.00	97.48	17.60	16.96
Gemini-2.5-Pro	5,612	4,747	918	84.59	97.20	20.08	19.34
Gemini-2.5-Flash	5,898	5,813	1,125	98.56	97.75	19.96	19.35
Claude-Opus-4.6	5,412	5,412	977	100.00	87.31	21.25	18.05
DeepSeek-R1	5,990	5,990	1,031	100.00	88.96	19.10	17.21
DeepSeek-V3	5,891	5,891	1,066	100.00	96.01	18.83	18.10
GLM-4.7	5,993	5,993	1,082	100.00	93.43	19.72	18.05
GLM-5	5,657	5,657	910	100.00	90.53	18.67	16.09
o4-mini	5,829	5,829	896	100.00	87.32	18.08	15.37
MiniMax-M2.5	5,996	5,996	928	100.00	84.79	18.48	15.48
Model-macro mean (15)					93.0	19.4	**17.70**

G Broader Impact

Positive impacts. InertiaBench provides a framework for evaluating whether models carry acknowledged task updates into later answers. By separating acknowledgment, update application, and recovery after reminders, it can support correction-aware training and multi-turn robustness evaluation. The reminder comparisons also motivate assessing prompting choices by correct recovery, rather than assuming that additional reasoning or an answer change is sufficient.

Risks and mitigations. Our benchmark examines failures to apply task updates in multi-turn conversations. In the wrong hands, knowledge of this failure mode could be used to deliberately confuse or mislead deployed systems by exploiting their inertia—for example, injecting a false premise early in a conversation to anchor subsequent reasoning. We note, however, that this attack vector relies on no information beyond what is publicly known about LLM context dependence. Responsible disclosure and open publication of InertiaBench are therefore appropriate; the community benefit of enabling mitigation research outweighs the marginal risk of disclosing an already observable phenomenon.

Dataset and evaluation limitations. All source corpora (FOLIO, ReClor, MATH-500, ZebraLogic, LogiQA, GPQA Diamond) are used under their respective licences for research purposes. The benchmark evaluates only API-accessible models; open-weight models running locally may exhibit different inertia profiles. Our correction typology (SR, GR, CE) covers changes to supplied information, requested goals, and applicable constraints, but is not exhaustive; temporal updates and multi-step retractions remain directions for future work.

H Domain-Dependent Inertia Pathology and Inertia Spectrum

The domains exhibit different final-answer profiles. FL, CB, and WK have higher old-reference match rates, whereas AN combines low IR with high BF. Within FL, CB, and WK, GR has the highest aggregate IR; in AN, it has the lowest. These results distinguish old-reference matching from errors that match neither reference and complement the analysis in Section 4.2.

Inertia Spectrum breakdown. Figure 14reports the full RFC-based Inertia Spectrum decomposition across FL (3 cells), CB (CB_GR), and WK (WK_SR). Anchored instances constitute 28–40% of all instances across domains.

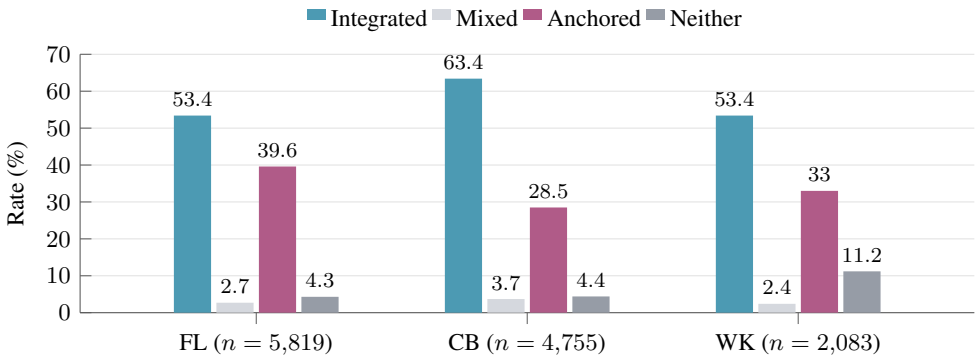


Figure 14: Inertia Spectrum decomposition by domain (RFC majority-vote labels). *Integrated*: RFC_{new}=1, RFC_{old}=0; *Mixed*: both=1; *Anchored*: RFC_{new}=0, RFC_{old}=1; *Neither*: both=0.

I Details of the model-level experiment

For each of the 17 model configurations, the Layer-1 evaluation targeted the full InertiaBench under the Corrupt-and-Correct protocol. Table 7 summarizes the model families and categories, along with the number of evaluation records included in the Layer-1 analysis for each configuration.

Table 7: **Model configurations and Layer-1 evaluation counts.**

Model	**Type**	**Family**	**Valid** N
QwQ-Plus	LRM	Qwen/Alibaba	5,996
QwQ-32B	LRM	Qwen/Alibaba	5,883
DeepSeek-R1	LRM	DeepSeek	5,990
GLM-5	LRM	Zhipu AI	5,657
MiniMax-M2.5	LRM	MiniMax	5,996
o4-mini	LRM	OpenAI	5,829
DeepSeek-V3	LLM	DeepSeek	5,891
GLM-4.7	LLM	Zhipu AI	5,993
Qwen-Max	LLM	Qwen/Alibaba	5,993
Qwen-Plus	LLM	Qwen/Alibaba	5,828
Qwen2.5-72B	LLM	Qwen/Alibaba	5,992
Qwen2.5-7B	LLM	Qwen/Alibaba	5,995
Claude-Opus-4.6	LLM	Anthropic	5,412
Claude-4.6-Sonnet	LLM	Anthropic	4,201
Gemini-2.5-Pro	LLM	Google	5,612
Gemini-2.5-Flash	LLM	Google	5,898
GPT-4.1	LLM	OpenAI	5,917
Total			**98,083**

J Dataset Construction and Prompt Templates

This appendix documents the construction process represented in the available source snapshot. Table 8 summarizes its execution order, and Table 9 maps stored fields to the dialogue notation used in the paper. Sections J.1 and J.2 describe the source-loading routes, call settings, and prompt templates. The templates preserve the source wording and placeholders; headings, captions, and explanatory notes are not part of the model inputs.

Table 8: Construction steps in the recovered implementation.

Step	Operation	Input, output, and scope
1	Source loading	Route a source task to a domain–update cell.
2	Update generation (P1)	Generate an update and candidate references from T_1 and one of twelve instructions (P1a–P1l).
3	Re-solving (P2)	Solve $T_1 + T_3$ without receiving the candidate revised reference; T_7 does not yet exist.
4	Answer comparison (P3)	Compare the candidate and re-solved revised answers; an explicit negative triggers rejection.
5	Buffer generation (P4)	Generate a buffer from the domain and T_1; T_3 is not included in this call.
6	Quality scoring (P6–P7)	When enabled, require answer-difference score ≥ 4 and update-naturalness score ≥ 3 before saving the record.
7	Later rewrite (P5)	Construct or reuse T_7 in a separate evaluation script; the rewrite call allows up to 500 output tokens.

Table 9: Recovered field names and their roles in the evaluation dialogue.

Recovered field	Manuscript role
`turn1_question`	Initial user task, T_1.
`turn3_update`	User update, T_3.
`turn4_buffer_question`	Intervening user question, T_5.
`turn4_buffer_answer`	Generated buffer reference, not the evaluated model's observed T_6.
`turn5_question`	Legacy generic final follow-up, not a standalone revised task.
`t5_modified`	Standalone revised task, T_7, produced later.
`t6` in the later results	Final model response, T_8.

J.1 SOURCE ROUTES, SETTINGS AND VALIDATION BEHAVIOR

Table 10 lists the source routes and selection rules. Fallback availability describes loader behavior; identifying the route used for a historical record requires separate source provenance.

Table 10: Source routes and selection rules for the twelve domain–update cells.

Cells	Source route	Selection and fallback behavior
FL–SR, FL–CE	FOLIO	Validation split loaded from a mirror; premises are split before task assembly.
FL–GR	ReClor	Training contexts longer than 200 characters; the name `hard` does not establish an official hard split.
AN–SR/GR/CE	MATH-500	Test examples with difficulty level at least four.
CB–SR, CB–CE	ZebraLogic	At least four houses; synthetic-puzzle generation (P9) is available as a fallback.
CB–GR	LogiQA	Training text, with a mirror fallback.
WK–SR/GR/CE	GPQA	Diamond routes are attempted, but main-subset routes are also permitted.

Table 11: Output budgets and call-specific handling in the recovered implementation.

Call	Output limit (tokens)	Handling
JSON generation	1,200 / 1,600 / 2,000	Up to three parsing attempts; append a JSON-only suffix to the system message.
Re-solving	2,000	Use $T_1 + T_3$, not the later T_7.
Semantic comparison	5	Reject replies beginning with `no`.
Quality scoring	800 or 10	PAI and DashScope limits, respectively; the message roles differ (P8).
Standalone rewrite	500	Nonempty cached rewrite: no API call.

Sampling settings and Validation behavior. Python random seed 42 controls local source ordering, not API sampling. Temperature, top-p, and an API sampling seed are not explicitly specified in the inspected construction source; missing historical settings are not replaced with present-day provider defaults. The generator self-reports `answer_delta_confirmed`; separate calls score answer difference (threshold 4) and naturalness (threshold 3). Extraction failure can return 3 in either backend, meeting only the naturalness threshold.

J.2 CONSTRUCTION PROMPT TEMPLATES

This subsection presents the prompt templates used at different construction stages. Figure 15 shows the shared update-generation template P1. Figures 16 and 17 then list the twelve cell-specific instruction branches that replace the placeholder {`instruction`} in P1. Figure 18 presents the two follow-up verification calls used before standalone rewriting, while Figure 19 shows the orthogonal-buffer prompt and the later rewrite template. Finally, Figure 20 summarizes the two scoring prompts, and Figure 21 shows the backend-specific scoring-message assembly together with the optional synthetic Zebra-style fallback. Figures 15– 17 present the shared update-generation template and its twelve cell-specific instruction branches. FigureFigure 18 presents revised-answer verification, Figure 19 presents buffer generation and standalone rewriting, and Figures 20– 21 document quality scoring and message assembly. Template wording and placeholders follow the source snapshot described at the start of this appendix.

J.2.1 UPDATE GENERATION: SHARED SYSTEM AND USER TEMPLATES

Figure 15 presents the common P1 template shared by all 12 domain–update settings. The left panel contains the system message and the first half of the user message; the right panel contains the required output schema and the remaining format constraints. The two user-message halves are split only for display. {instruction} are substituted before execution. The last variable selects exactly one branch on the following pages. The system message shown here combines the base template and the JSON-only wrapper. The output schema requests concise candidate reference answers and generator-reported metadata, including answer_delta_confirmed. Revised-answer checking is performed through the separate P2 and P3 calls.

P1 — System and user input

System message

You are an expert at designing challenging multi-turn reasoning evaluation data for {domain}. Be precise; solve both answers step by step to verify they are correct.

Respond ONLY with a valid JSON object. No markdown fences, no preamble, no text outside the JSON.

User message: part 1 of 2

Problem:
{question}

Design an update of this type:
{instruction}

Requirements:
1. Identify the pivot point (value/fact/clue to change)
2. Write a natural Turn 3 user message
3. Solve the ORIGINAL problem step by step → original_answer
4. Apply the pivot change, solve the MODIFIED problem step by step → corrected_answer
5. The two answers MUST be different

P1 — User output requirements

User message: part 2 of 2

Return JSON with EXACTLY these 9 keys:
- pivot_description
- pivot_value
- pivot_type (numeric|logical|constraint|factual)
- suggested_new_value
- original_answer (exact answer to the original problem)
- turn3_update (natural Turn 3 user message)
- corrected_answer (exact answer to the MODIFIED problem, must differ)
- naturalness_score (integer 1-5)
- answer_delta_confirmed (true only if you solved both step by step)

CRITICAL FORMAT RULES:
- original_answer and corrected_answer must be SHORT strings ($\leq$ 2 sentences).
- For constraint/puzzle questions: give ONE specific conclusion (e.g. 'The Norwegian owns the fish.'), NOT a full solution table or dict.
- For numeric questions: give the number only (e.g. '42' or '$3\sqrt{6}$'), NOT a dict.

Figure 15: Shared update-generation templates. The user message is split across the two panels only for display: read part 1 followed by part 2 as a single message. Panel headings and this caption are not sent to the model.

J.2.2 DOMAIN–UPDATE INSTRUCTION BRANCHES

Figures 16 and 17 present the twelve cell-specific branches inserted into {instruction} in the shared P1 user message. All branches use the same system message, common requirements, and nine-field output schema. Bracketed examples are retained from the source strings.

Implementation note (not sent to the model). The branches guide generation. These branches specify the requested update operation and the task components to preserve. Reference checking and sampled human review are described separately.

Formal logic (FL)

P1a — State refinement

Change a PROPOSITIONAL VALUE (True/False/Unknown) in the premises. Turn 3 example: 'Actually, I need to revise the premise — [old] should be [new].'

P1b — Goal retargeting

Change the PROOF GOAL (what conclusion needs to be evaluated). Turn 3 example: 'Wait, instead of evaluating [A], I need to determine [B].'

P1c — Constraint evolution

Add a NEW LOGICAL PREMISE to the existing set. Turn 3 example: 'Oh, I forgot this additional fact: [new premise].'

Algorithmic/numeric (AN)

P1d — State refinement

Change a SPECIFIC NUMERIC VALUE or coefficient. Turn 3 example: 'Oops — the [X] should be [Y], not [old]. Can you redo?'

P1e — Goal retargeting

Change the OPTIMIZATION OBJECTIVE or what quantity is asked for. Turn 3 example: 'Actually I need [new_goal] instead of [old_goal].'

P1f — Constraint evolution

Add a NEW MATHEMATICAL CONSTRAINT (integer, upper bound, divisibility, etc.). Turn 3 example: 'I forgot — the answer must also satisfy: [constraint].'

Figure 16: Cell-specific instruction branches for formal logic (FL) and algorithmic/numeric reasoning (AN). Each branch replaces the placeholder {instruction} in the shared P1 user template.

Constraint-based (CB)

P1g — State refinement

Correct a SPECIFIC CONSTRAINT CLUE (position, attribute, or relation). Turn 3 example: 'Actually [old clue] should be [new clue] — I had wrong info.'

P1h — Goal retargeting

Change WHICH ENTITY to find the attribute for. Turn 3 example: 'Sorry, instead of finding [X]'s [attr], I need [Y]'s [attr].'

P1i — Constraint evolution

Add a NEW CONSTRAINT CLUE to the puzzle. Turn 3 example: 'I should also mention: [new clue].'

World knowledge (WK)

P1j — State refinement

Correct a KEY SCIENTIFIC/FACTUAL PREMISE. Turn 3 example: 'Wait, I had wrong info — [old fact] is actually [corrected]. Does that change the answer?'

P1k — Goal retargeting

Change THE SPECIFIC ASPECT BEING ASKED about. Turn 3 example: 'Actually, instead of asking [original], I want to know [new angle].'

P1l — Constraint evolution

Add a NEW DOMAIN RESTRICTION or experimental condition. Turn 3 example: 'One important context I forgot: [condition]. Does this affect your answer?'

Figure 17: Cell-specific instruction branches for constraint-based reasoning (CB) and world knowledge (WK). The shared P1 system message, common requirements, and nine-field output schema remain unchanged.

J.2.3 RE-SOLVING AND REVISED-ANSWER COMPARISON

Figure 18 shows the two calls used to re-solve the modified task and to compare the resulting answer with the candidate revised answer. Importantly, P2 operates on $T_1 + T_3$ before the later standalone rewrite is constructed.

Implementation note (not sent to the model). The candidate revised answer is absent from this call. The template asks for work but requests a single `verified_answer` field.

P2 — Re-solve the modified problem

System message

```
You are a precise {domain} solver. Solve step
 by step and return JSON only.
```

User message

```
Problem (original):
{question}

Modification applied: {result.get('turn3_update',
 '')}

Solve the MODIFIED problem completely from
 scratch. Show your work, then give the final
 answer.

Return JSON with exactly 1 key:
- verified_answer: your computed final answer
 (string, be concise)
```

P3 — Compare the two answers

System message

```
You are a lenient semantic equivalence judge.
 Reply with only 'yes' or 'no'.
```

User message

```
Do these two answers express the same core
 meaning or reach the same conclusion?
Answer A: {core_corrected}
Answer B: {core_verified}

Be lenient with paraphrase differences. Reply
 with only: yes / no
```

Figure 18: P2 and P3 verification calls used before standalone rewriting. P2 re-solves the modified task from $T_1 + T_3$; P3 compares the candidate revised answer with the re-solved answer.

J.2.4 INTERVENING QUESTION AND LATER STANDALONE REWRITE

Figure 19 presents the prompt used to generate the orthogonal buffer question and the later prompt used to construct the standalone revised task. The historical turn naming in the code is preserved in the recovered strings but differs from the manuscript notation.

Implementation note (not sent to the model). The source phrase `Turn 4` and field prefix `turn4_buffer` refer to the user buffer that the paper denotes T_5. P4 generates the buffer question and its reference answer from T_1. Table 9 maps the legacy buffer-field names to T_5 in the manuscript. The generated buffer reference is separate from the evaluated model's response at T_6.

P4 — Generate the buffer question

System message

You are an expert at designing natural multi-turn academic conversations.

Respond ONLY with a valid JSON object. No markdown fences, no preamble, no text outside the JSON.

User message

Domain: {domain}
Main problem:
{question}

Generate a Turn 4 buffer question that:
1. Is RELATED to the same topic
2. Is LOGICALLY ORTHOGONAL (does NOT reveal the main answer)
3. Sounds like a natural follow-up
Return JSON: buffer_question, buffer_answer, orthogonality

P5 — Rewrite the final question

System message

You are a precise problem editor. Return only the rewritten problem.

User message

Original problem:
{row['turn1_question']}

The user issued this correction:
{row['turn3_update']}

Rewrite the original problem incorporating the correction. Replace the original value/condition with the updated one. Keep all other parts exactly the same. Return ONLY the rewritten problem, nothing else.

Figure 19: P4 and P5 construction prompts. P4 generates the orthogonal buffer question and buffer reference; P5 later rewrites the original problem into the standalone revised task denoted T_7 in the manuscript.

J.2.5 AUTOMATED QUALITY SCORING

Figure 20 shows the two scoring prompts used to assess whether the update changes the answer and whether the update itself sounds natural. The implemented thresholds are 4 for answer difference and 3 for update naturalness. P6 scores the difference between the supplied reference answers, with a threshold of 4. P7 scores the naturalness of the update, with a threshold of 3. The P7 score is separate from the generator's self-reported `naturalness_score` in P1. For P7, the input contains the first 200 characters of T_1 and the complete update. The documented score-extraction fallback is 3, which meets the naturalness threshold but not the answer-difference threshold. Candidate validation uses the extracted score. Backend-specific message assembly is shown in Figure 21.

Output and handling. The prompt requests a score in double brackets. The PAI parser searches for that format, whereas the fallback parser searches for a digit from 1 to 5 in its single-digit response. The validation function uses the extracted score rather than the backend's generic `passed` flag.

P6 — Answer-difference scoring

Rubric / system message

Please act as an impartial judge and evaluate whether there is a substantial difference between the following two answers.

Scoring rubric (1-5):
1: The two answers are identical.
2: Different wording, same meaning.
3: Minor differences.
4: Obvious differences, different conclusions.
5: Completely different.

Provide your score at the end in the format [[N]], e.g. [[4]].

User message

Answer A (before update):
{inst['original_answer']}
Answer B (after update):
{inst['corrected_answer']}

Is there a substantial difference between these two answers?

P7 — Update-naturalness scoring

Rubric / system message

Please act as a linguistics expert and evaluate whether the user's update statement is natural and fluent.

Scoring rubric (1-5):
1: Very stiff, completely unnatural.
2: Somewhat unnatural.
3: Basically natural, acceptable.
4: Natural and fluent.
5: Very natural, perfectly matches real human dialogue.

Provide your score at the end in the format [[N]], e.g. [[4]].

User message

Original question:
{inst['turn1_question'][:200]}...

Turn 3 user update: {inst['turn3_update']}

Is this update statement natural?

Figure 20: Scoring prompts for answer difference (P6) and update naturalness (P7). These two prompts assess different properties and are thresholded at 4 and 3, respectively.

J.2.6 Backend message assembly and source fallback

Figure 21 separates two different kinds of artifacts: the left panel is a source-code expression that assembles backend-specific scoring messages, whereas the right panel is a recovered prompt template for the optional synthetic Zebra-style fallback.

P8 — DashScope message assembly

System message

You are a precise scoring assistant. Reply with a single digit only.

User-message assembly (source code)

```
combined = (
    system_prompt.strip()
    + "\n\n---\n\n"
    + user_prompt.strip()
    + "\n\n---\n"
    + "Based on the rubric above,
 reply with ONLY a single integer
 score "
    "from 1 to 5. No explanation,
 no brackets, just the digit."
)
```

P9 — Synthetic Zebra-style fallback

System message

You are an expert at creating logic puzzle datasets. Generate a solvable 4-house zebra-style logic puzzle. Return JSON with exactly two keys: 'puzzle' (the full puzzle text including clues) and 'solution' (the complete solution table as a string).

Respond ONLY with a valid JSON object. No markdown fences, no preamble, no text outside the JSON.

User message

Generate a new 4-house zebra puzzle. Include 8-10 clues. Make it uniquely solvable.

Figure 21: Backend-specific scoring-message assembly (left) and the optional synthetic Zebra-style fallback prompt (right). Source-code expressions are shown in monospace and are not themselves model messages.

Implementation note (not sent to the model). `system_prompt` and `user_prompt` are the inputs shown in P6 or P7. The code concatenates their values; variable names and the extractor's `EXPRESSION` markers are not sent literally.

Implementation note (not sent to the model). The system and user messages above are reconstructed from source.

Shared JSON-only system suffix

System-message assembly (source code)

```
sys_j = (system + "\n\n" if system else "") + (
    "Respond ONLY with a valid JSON object. "
    "No markdown fences, no preamble, no text outside the JSON. "
)
```

Figure 22: Source-code assembly of the shared JSON-only suffix used in P1, P4 and P9.

The companion archive contains the prompt registry, source files, and hash manifests for the documented snapshot. Versioned examples are maintained separately from the source templates.

J.3 Retained snapshots and log accounting

Scope and saved records. Table 12 summarizes the retained construction snapshot and its generation logs. The snapshot contains 5,996 saved records across the twelve cells: eight cells contain 500 records and four contain 499. Exact grouping by the complete (cell, T_1, T_3) strings yields 5,205 input keys. This grouping concerns the task-input fields; reference answers and intermediate text are separate fields of the saved records.

Table 12: Construction-snapshot records and generation-log events by cell. Saved records and distinct input keys summarize the retained task files; `ok`, `skip`, and `error` summarize recorded log events. Extra records equal saved records minus distinct input keys.

	Saved-file snapshot			Recorded log events		
Cell	**Saved rows**	**Distinct input keys**	**Extra rows**	`ok`	`skip`	`error`
Formal Logic (FL)						
FL_SR	500	345	155	495	8	852
FL_GR	500	499	1	495	92	766
FL_CE	500	350	150	495	20	667
Algorithmic/Numeric Reasoning (AN)						
AN_SR	499	304	195	494	45	2,879
AN_GR	499	419	80	494	18	3,894
AN_CE	499	426	73	494	46	4,497
Constraint-Based Reasoning (CB)						
CB_SR	499	494	5	637	90	1,763
CB_GR	500	498	2	495	109	602
CB_CE	500	495	5	528	112	1,434
World Knowledge (WK)						
WK_SR	500	428	72	495	36	1,243
WK_GR	500	486	14	495	38	1,510
WK_CE	500	461	39	495	16	1,559
Total	**5,996**	**5,205**	**791**	**6,112**	**630**	**21,666**

Log accounting. There are 28,685 log entries: 6,112 `ok`, 630 `skip`, 21,666 `error`, 142 `start`, and 135 `summary` entries. Every first `start` already reports five saved rows.

Filtering and error messages. Table 13 partitions all 22,296 error and skip entries. The four error categories use retained message text: re-solve mismatch matches `Self-verify FAILED`; JSON matches `JSON parse` or `Expecting value`; delta assertion matches `answer_delta =False`; and Other contains the residual entries. Skip categories retain their original labels. A delta assertion during generation is not merged with an answer-change skip from later validation.

Table 13: Generation-log error and skip events by cell. Categories are assigned from the recorded message text. Each row totals the corresponding `error` and `skip` events in Table 12; zeros indicate no recorded event matching the category.

	Error messages				Skip labels		
Cell	Re-solve mismatch	JSON	Delta assertion	Other	Answer change	Natural-ness	Total
Formal Logic (FL)							
FL_SR	610	0	39	203	8	0	860
FL_GR	752	0	14	0	92	0	858
FL_CE	397	1	66	203	5	15	687
Algorithmic/Numeric Reasoning (AN)							
AN_SR	1,887	671	59	262	45	0	2,924
AN_GR	1,652	2,114	126	2	18	0	3,912
AN_CE	2,027	2,279	191	0	46	0	4,543
Constraint-Based Reasoning (CB)							
CB_SR	1,691	46	25	1	90	0	1,853
CB_GR	594	0	8	0	108	1	711
CB_CE	1,305	95	32	2	112	0	1,546
World Knowledge (WK)							
WK_SR	1,220	11	9	3	36	0	1,279
WK_GR	1,405	10	41	54	38	0	1,548
WK_CE	1,524	8	26	1	16	0	1,575
Total	**15,064**	**5,235**	**636**	**731**	**614**	**16**	**22,296**

Of the raw `other` entries, 15,064 contain re-solve-mismatch messages, 636 contain delta assertions, 95 contain JSON-related messages, and 731 remain unclassified. A further 5,140 JSON-related entries already carry `json_parse`. These counts summarize logged events rather than distinct candidate tasks. The categorization preserves the original messages while distinguishing re-solving mismatches, JSON-related errors, answer-change assertions, and other recorded events.

Separate supplementary selection. The later supplementary-preparation artifacts describe a 600-item formal release, with 50 items per cell, together with a separate development selection. Preparation-stage replacements and core-field revisions are documented in the release receipt.

Separate supplementary selection. The later preparation artifacts identify an eligible pool of 4,549 records, a 624-item preparation manifest (600 formal and 24 development items), and a 600-item formal release (50 per cell). The release receipt records seed 20260909, 578 retained formal assignments, 22 reserve replacements, and three selected core-field revisions.

K History-Free Baselines and Multi-Turn Accuracy

Evaluation scope. We evaluate two history-free, single-turn baselines across all 17 model configurations and the full InertiaBench dataset, covering all 12 domain–correction cells. Table 14 reports cell-level accuracies averaged across these configurations, alongside final-answer accuracy (FAA) under the full Corrupt-and-Correct protocol(Figure 2).

Baseline conditions. BL-2 presents the original problem T_1 and the explicit update T_3 together in a single request, without preceding dialogue. It evaluates whether the model can solve the revised task from an original-plus-update specification. BL-3 presents only the standalone revised question T_7 in a fresh context, measuring revised-task accuracy without the preceding user–model exchanges. In both conditions, correctness is assessed against the revised reference answer a_{corr}. FAA measures correctness against this reference at T_8 under the full dialogue. We report $\Delta_2 = \text{BL-2} - \text{FAA}$ and $\Delta_3 = \text{BL-3} - \text{FAA}$ in percentage points; positive values indicate higher baseline accuracy.

Table 14: History-free baseline accuracy and multi-turn FAA across 17 model configurations. BL-2 receives T_1 and T_3 in a single request; BL-3 receives only T_7. Δ_2 = BL-2 − FAA and Δ_3 = BL-3 − FAA are accuracy differences in percentage points (pp).

	Accuracy (%)			Difference (pp)	
Cell	**BL-2**	**BL-3**	**FAA**	$\mathbf{\Delta_2}$	$\mathbf{\Delta_3}$
FL_SR	74.3	51.9	54.5	+19.8	−2.6
FL_GR	82.2	44.8	43.6	+38.6	+1.2
FL_CE	71.3	35.0	34.3	+37.0	+0.7
AN_SR	37.6	25.5	26.0	+11.6	−0.5
AN_GR	37.5	23.6	25.1	+12.4	−1.5
AN_CE	39.2	20.4	21.0	+18.2	−0.6
CB_SR	27.5	36.0	31.5	−4.0	+4.5
CB_GR	66.0	42.4	50.3	+15.7	−7.9
CB_CE	28.6	35.0	30.1	−1.5	+4.9
WK_SR	38.5	34.4	42.1	−3.6	−7.7
WK_GR	43.9	28.7	31.4	+12.5	−2.7
WK_CE	36.6	36.3	39.1	−2.5	−2.8
Overall	**48.4**	**34.0**	**35.8**	**+12.6**	**−1.8**

Results and interpretation. BL-2 exceeds multi-turn FAA in eight of the twelve cells, with the largest differences in FL_GR and FL_CE (38.6 and 37.0 percentage points, respectively). The comparison with BL-3 is mixed: multi-turn FAA is higher in eight cells and lower in four, with Δ_3 ranging from -7.9 to $+4.9$ percentage points. The direction and magnitude of the accuracy differences therefore depend on both the task cell and the single-turn presentation.

These baselines compare revised-task accuracy under two single-turn presentations with FAA under the full dialogue protocol. The accuracy contrasts in Table 14 complement the diagnostic measures in the main text: IR identifies old-reference answers, while AnU measures their joint occurrence with substantive acknowledgment in the same dialogue.

L Limitations and Future Work

Limitations. InertiaBench is designed as a controlled behavioral diagnostic of whether an explicit task correction is acknowledged, subsequently applied, and recoverable after failure. This design necessarily defines the scope of its conclusions. First, the benchmark covers four task groups and three update types—state refinement, goal retargeting, and constraint evolution—but does not exhaust the ways that real conversations may evolve. Successive corrections, partial retractions, temporal updates, ambiguous revisions, and interactions among multiple updates remain outside the current protocol. Restricting each revision to an explicit, answer-changing update allows the benchmark to distinguish old-reference reuse from other final-answer errors under a common evaluation structure.

Second, thinking inertia is treated here as an observable behavioral phenomenon, rather than a claim about a model's latent cognitive mechanism. IR, AnU, RFC, and IPR characterize relationships among correction acknowledgment, subsequent responses, and post-reminder behavior, but do not by themselves determine why a particular failure occurs. Likewise, Layer-2 persistence is measured on model-specific Layer-1 old-answer failures and therefore describes recovery on those observed failure sets rather than a controlled causal comparison between model classes. Finally, the evaluated models and API settings provide a broad but finite snapshot of current systems, while naturally occurring user corrections may exhibit greater linguistic and contextual diversity than the controlled revisions studied here.

Future work. We further direction is to study thinking inertia in large reasoning models (LRMs) and long-horizon agents. For LRMs, we plan to examine how correction robustness interacts with reasoning depth, test-time reasoning budgets, self-reflection, verification, and reasoning restarts, and whether interventions such as correction recap, selective recomputation, state reset, or correction-aware training can improve update application without discarding still-valid context. For agentic

systems, the problem extends beyond individual responses: corrections may need to propagate through persistent plans, memories, tool outputs, intermediate states, and downstream actions. Future work can therefore examine whether acknowledged updates are consistently reflected in plan revision, memory or state invalidation, tool use, and subsequent actions, as well as how obsolete information propagates across an agent trajectory and whether recovery persists over later steps. Together, these settings provide a natural extension of InertiaBench from measuring answer-level acknowledgment–application mismatches to studying the persistence and propagation of thinking inertia in increasingly stateful reasoning systems.