Failing to Falsify: Evaluating and Mitigating Confirmation Bias in Language Models

Ayush Rajesh Jhaveri[1,2], **Anthony GX-Chen**[1], **Ilia Sucholutsky**[1], **Eunsol Choi**[1]
[1]New York University
[2]Capital One*
{ayush.rajesh.jhaveri,eunsol}@nyu.edu

Abstract

Confirmation bias, the tendency to seek evidence that supports rather than challenges one's belief, hinders one's reasoning ability. We examine whether large language models (LLMs) exhibit confirmation bias by adapting the rule-discovery study from human psychology: given a sequence of three numbers (a "triple"), an agent engages in an interactive feedback loop where it (1) proposes a new triple, (2) receives feedback on whether it satisfies the hidden rule, and (3) guesses the rule. Across eleven LLMs of multiple families and scales, we find that LLMs exhibit confirmation bias, often proposing triples to confirm their hypothesis rather than trying to falsify it. This leads to a slower and less frequent discovery of the hidden rule. We further explore intervention strategies developed for humans (e.g., encouraging the agent to consider counterexamples). We find that prompting LLMs with such instructions consistently decreases confirmation bias in LLMs, improving rule discovery rates from 42% to 56% on average. Lastly, we mitigate confirmation bias by distilling intervention-induced behavior into LLMs, showing generalization to a new domain (the Blicket test) and beyond rule discovery (mathematical self-correction). Our work shows that confirmation bias is a limitation of LLMs in hypothesis exploration, and that it can be mitigated via injecting interventions designed for humans.

1 Introduction

Humans often form an initial hypothesis and then seek evidence that supports their hypothesis rather than refutes it [26]. For example, a doctor who suspects a certain diagnosis may focus on specific symptoms that support it and interpret positive test results as confirmation. Such selective exploration, known as **confirmation bias**, leads to inefficient learning and decision-making. Figure 1 exemplifies such a scenario, where consistently using confirmatory examples leads to failed rule discovery [39].

We aim to understand confirmation bias for LLMs acting as agents [43, 38, 35]. In these scenarios, LLMs perform tasks that require exploratory reasoning: generating hypotheses, testing them, and refining beliefs in iterative loops. An agent that only verifies its current guess will be suboptimal compared to an agent which also seeks evidence that can falsify its current guess. To this end, we adapt a simplified rule discovery study in humans [39] and synthetically generate large-scale data, where an initial triple can match multiple valid rules. We then evaluate the LLMs' ability to identify the hidden rule by proposing and testing new triples. This setup enables step-by-step observation of how models revise hypotheses based on observations. We further quantify a model's confirmation bias by measuring the ratio between incompatible and compatible tests it proposes.

We experiment with eleven LLMs of different families and scales. Models exhibit varying extents of confirmation bias and task success rates (from 6% to 78%, within a maximum turn limit of

*This research was completed prior to joining Capital One.

You can find data and code for this paper here: https://github.com/ayushjhaveri/llm-confirmation-bias

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

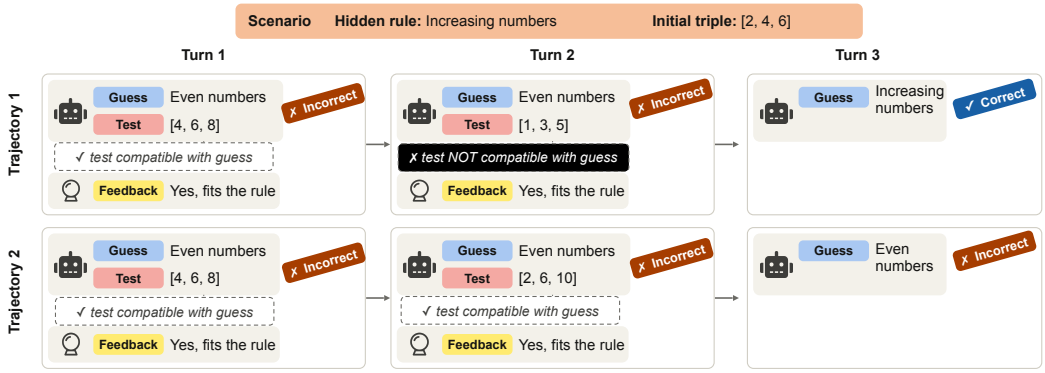


Figure 1: **Experimental Setting to Measure Confirmation Bias.** We show two trajectories for a rule discovery task, in which an agent aims to infer a hidden rule over multiple turns. Starting from an initial triple, the agent (🤖) states a guess about the rule and proposes a new triple to test, then receives binary feedback on whether that triple satisfies the hidden rule. A **compatible test** is consistent with the agent's current guess, whereas an **incompatible test** contradicts it. Trajectory 1 introduces an incompatible test in the second turn, which eliminates the incorrect guess and leads to discovery of the correct rule. In contrast, Trajectory 2 proposes compatible tests in both turns, showing confirmation bias, and keeps the incorrect guess.

45 interactions). We find recent LLMs with longer reasoning traces [33, 42] show much stronger performance, as well as lower confirmation bias. As was in humans, we observe a negative correlation between the extent of confirmation bias and task success rates.

We then study methods to decrease confirmation bias, using two well-established interventions originally developed for humans: Dual-Goal [14] and Think-in-Opposites [5]. Prompting LLMs with these strategies leads to meaningful performance gains. We then investigate whether fine-tuning LLMs via symbolic knowledge distillation [40] to use these intervention strategies helps decrease confirmation bias, alleviating the need for task-specific inference-time interventions. Finally, we study whether the fine-tuned models generalize to a new rule-discovery domain, the Blicket Test [15], and beyond rule discovery to mathematical reasoning [30, 6]. On the Blicket Test, the fine-tuned models exhibit lower confirmation bias and improved task success, while on mathematical reasoning tasks, they show improved self-correction after incorrect solutions.

Together, our framework enables studying whether LLMs exhibit confirmation bias during hypothesis exploration. Intervention studies further present promising results in how psychologically inspired framings can reduce this bias and improve task performance. By internalizing debiasing behaviors through symbolic knowledge distillation, we show that these behaviors can generalize across rule-discovery domains and beyond rule discovery to broader reasoning tasks.

2 Background

We first describe Wason's seminal 1960 study which quantified the notion of confirmation bias.

Setting Human participants were shown the triple of integers (i.e., `2-4-6`), told it conformed to a hidden rule (which was any three numbers in increasing order), and asked to discover that rule by proposing new triples. After each proposal, an experimenter responded `YES` or `NO`, indicating whether the proposed triple conformed to the hidden rule. Participants could guess the rule at any time; incorrect guesses did not terminate the experiment. The experiment continued until the participant correctly guessed the rule, expressed a wish to give up, or the time limit (45 minutes) was reached.

Measuring Confirmation Bias Confirmation bias was quantified by measuring the ratio between the number of turns when a participant tests triples that contradict (*incompatible*) versus confirm (*compatible*) their current hypothesis. Participants were required to record the reason for each proposed triple, which were used to infer their current hypothesis. Higher ratios indicate more disconfirmatory testing and thus lower confirmation bias.

Results The study included 29 participants. $6/29$ participants announced the correct rule on their first attempt, and $21/29$ eventually discovered the correct rule. Participants who were correct on their first announcement exhibited more disconfirmatory exploration, with a mean I:C of 1.79, compared to 0.24 for those whose first announcement was incorrect. Among who initially failed, those who succeeded on the *second* announcement showed a higher I:C (0.50) than those who did not (0.19). These results show that in humans, higher I:C correlates with earlier and more frequent rule discovery.

Caveats Later follow up work [17] refined Wason study, arguing positive testing strategy does not necessarily mean confirmation bias. Compatible test can falsify the current hypothesis if the hidden hypothesis is narrower than the current hypothesis. In this study, we follow Wason's study design, as target hypotheses are generally broad.

Studies of Wason 2-4-6 in LLMs O'Leary [31] conducts a small-scale Wason 2-4-6 study using two-guess sessions, making qualitative judgments based on similarity between successive guesses. They do not evaluate exploration through test-triple selection or consider multiple rules and instances. Another study [3] adapted Wason 2-4-6 to evaluate multi-turn reasoning in LLMs. Here, each episode initializes a hidden Boolean rule over three numbers, and the model may propose up to 30 test triples, receiving binary feedback before making a rule guess. They report the guessing accuracy, framing it as a small-scale evaluation benchmark (N=50) for reasoning and hypothesis exploration abilities for LLMs without focusing specifically on confirmation bias. The evaluated LLMs achieve at most a 26% task success rate. We expand this approach substantially by introducing a larger-scale dataset and a framework to measure the confirmation bias. We use this framework to study mitigation strategies for reducing LLM confirmation bias and improving performance.

3 Framework to Identify Confirmation Bias in LLMs

Notation Summary

Test Scenario	$(\mathbf{x}_{\text{ini}}, \mathbf{r}^*)$	(Initial triple of integers (input), Hidden target rule)
Episode	e	A full sequence of rule discovery game: initial triple followed by 45 guess, test, feedbacks turns
Guess	$\mathbf{r}_i$	Model's guessed rule at turn i
Test	$\mathbf{x}_i$	Triple of integers proposed at test turn i
Feedback	$y_i = \mathbf{r_i}(\mathbf{x_i})$	A binary label whether triple $\mathbf{x_i}$ conforms to rule $\mathbf{r_i}$
First correct turn	$t_e^\star$	Turn index of the first correct rule announcement in episode e, if any

Setting We simulate a multi-turn reasoning game for LLMs. Each game scenario consists of an initial triple of integers ($\mathbf{x}_{\text{ini}}$) and a hidden rule $\mathbf{r}^*$. Initially, the LLM is given $\mathbf{x}_{\text{ini}}$ and the task instruction.[2] Then, LLM takes alternating `Guess` and `Test` turns. On i-th `Guess` turn, the LLM states the guessed hidden rule $\mathbf{r}_i$ in text; on the subsequent (i-th) `Test` turn, it proposes a new triple $\mathbf{x}_i$ for evaluation. We denote that a triple $\mathbf{x}$ conforms to a rule $\mathbf{r}$ as $\mathbf{r}(\mathbf{x}) = 1$, and $\mathbf{r}(\mathbf{x}) = 0$ otherwise.

Then, the environment provides binary feedback y_i whether the proposed triple conforms to the hidden rule ($\mathbf{r_i}(\mathbf{x_i})$). We continue this for 45 turns. For episodes in which the model correctly identifies the hidden rule, let $t_e^\star$ denote the turn index of the first correct announcement, i.e., $Eq(\mathbf{r}_{t_e^\star}, \mathbf{r}^*) = 1$. We describe how our setting differs from Wason study on humans in the appendix A.6.

Evaluation Metrics We report reasoning process success, efficiency of the reasoning process, as well as the extent of confirmation bias exhibited during the multi-turn games.

(1) Task Success (Success ratio % ↑) We report the percentage of scenarios in which the LLM guesses the correct rule during the conversation. For each episode, we measure whether any of i-th guessed rule $\mathbf{r}_i$ matches the hidden rule $\mathbf{r}^*$ using LLM-as-a-judge for each scenario, $\mathbf{1}\Big(\bigvee_{i=1}^{n} Eq(\mathbf{r}_i, \mathbf{r}^*)\Big)$, where n is the maximum number of turns.

(2) # Turns until Success (↓) We report the number of turns LLMs took to guess the correct rule. This is only computed for episodes where the model successfully guessed the correct rule within the

[2]The full task instruction can be found in Appendix B.

Table 1: Example rule sets (each set has 4 rules) and sampled triples. Each rule is defined over integer triples (a, b, c). All triples are sampled from the feasible intersection of the rules.

Rule group	Initial triple samples
All odd; b is (strict) max; mixed signs; at least one multiple of 3	$(-9,\ 55,\ -71)$, $(-77,\ 75,\ -47)$
All prime numbers; non-increasing; all positive; contains a prime	$(79,\ 43,\ 31)$, $(89,\ 67,\ 2)$

allowed maximum number of turns. More formally, $\frac{1}{|\mathcal{E}_{\text{succ}}|}\sum_{e\in\mathcal{E}_{\text{succ}}} t_e^*$ where $\mathcal{E}_{\text{succ}}$ denotes the set of episodes in which the model correctly identifies the hidden rule.

(3) # Tokens per Turn (↓) We report the average number of tokens generated by the model per `Guess` or `Test` turn across all episodes. This metric serves as a proxy for the amount of compute spent on reasoning and hypothesis testing. At comparable task performance, lower values indicate more concise and efficient reasoning.

(4) Confirmation Bias Status (Incompatible:Compatible Ratio (I:C) ↑). Following Wason [39], we quantify the model's confirmation bias by measuring the ratio between the number of turns when it tests triples that contradict (*incompatible*) versus confirm (*compatible*) its current hypothesis.

For episode e, let C_e and I_e denote the numbers of *compatible* and *incompatible* tests, respectively:

$$C_e = \sum_{t=1}^{t_e^\star} \mathbb{1}[\mathbf{r_t}(\mathbf{x_t}) = 1]\,, \quad I_e = \sum_{t=1}^{t_e^\star} \mathbb{1}[\mathbf{r_t}(\mathbf{x_t}) = 0]\,, \quad \text{I:C} = \frac{\sum_{e\in\mathcal{E}} I_e}{\sum_{e\in\mathcal{E}} C_e}$$

where $\mathcal{E}$ denotes the total set of evaluation episodes.

To compute the compatibility of a triple with a rule, we first use an LLM to translate each rule into a Python function. Then we execute this Python function to decide the compatibility between the rule and the triple. The implementation details are in Appendix C.4.

Data Each game scenario requires an initial triple $\mathbf{x}_{\text{ini}}$, the target rule $\mathbf{r}^*$, and a set of N other rules $\{\mathbf{r}_i\}_{i=1}^N$ that satisfies the initial triple. We ensure there are other plausible rules that match the initial triple so that the LLM tests other triples to identify the hidden rule. We aim for a diverse, plausible rule set. Table 1 contains example rule sets and sampled initial triples.

We develop 4 rule sets for training, 2 for validation, and 4 for testing, with each rule set containing 4 rules. For each rule set, we sample triples that satisfy all four rules, and evaluate each triple against each rule as the target, yielding one episode per triple–rule pairing. Training uses 100 triples per rule set (1,600 evaluation episodes), validation uses 2 triples per rule set (16 episodes), and testing uses 5 triples per rule set (80 episodes). All reported evaluation metrics are computed on the held-out test split unless otherwise specified. The training and validation splits are used for distillation (Section 5). We describe the data generation process below.

Rule Set Generation We prompt an LLM to generate 40 game scenarios. First, we prompt the LLM to generate ten sets of four plausible rules over integer triples $(a, b, c) \in [-99, 100]^3$. The prompt encourages rules from several conceptual families: *ordering*, *arithmetic structure*, *parity or divisibility*, *sign*, and *number properties*. We instruct that there should be non-empty set of triples that can satisfy all four rules. After initial generation, we apply heuristic checks (e.g., detecting multi-step arithmetic dependencies or rules relying on a single variable). Rule sets that fail these checks are revised via reprompting, resulting in approximately 33% of the initial generations being corrected.

After filtering, we replace one of the LLM-generated rules in each rule set with a human-derived rule from prior work [4, 36]. The replacement is chosen such that there remains at least one triple satisfying all four rules. The data details are in Appendix A.

4 Replicating Human Studies

We now evaluate off-the-shelf LLMs with three interaction settings: Baseline, Dual-Goal, and Think-in-Opposites. The first simulates the Wason's study, and the other two mimic two intervention studies respectively. We ask the following two research questions: (1) Do LLMs exhibit confirmation bias? (2) Are intervention strategies to reduce the confirmation bias in humans effective for LLMs?

4.1 Interaction Settings

Baseline This adapts the instruction from Wason's classic 2–4–6 rule-discovery task [39] as is. The prompt and example interaction in Appendix B.1. Each episode begins with an initial `Guess` turn, where the model states its first guess at the rule. It continues with the model's `Test`, feedback on the test from the environment, then a new `Guess` turn.

Think-in-Opposites (TiO) This adapts the study of Branchini et al. [5] and instructs the LLM to focus on one property of the previous **Test** example triple and construct a new triple that is opposite with respect to that property. The prompt and example interaction for this intervention is in Appendix B.3. Except for the prompt, it follows the same interaction protocol as the baseline condition.

Dual-Goal We introduce a prompt adapted from the Dual-Goal paradigm [14] (prompt and example in Appendix B.2). The interaction protocol is identical to the baseline condition, with two modifications. First, during the `Guess` turn, the LLM must output two test rules: one for the DAX rule and one for the MED rule. The `DAX` rule corresponds to the original hidden rule, while the `MED` rule is its logical complement, capturing triples that violate the `DAX` rule. Second, during the `Test` turn, the LLM proposes a triple. The environment will evaluate this the same way (whether it fits the hidden rule or not), but instead of `YES`/`NO` feedback, it will output `DAX`/`MED` respectively.

4.2 Models Evaluated

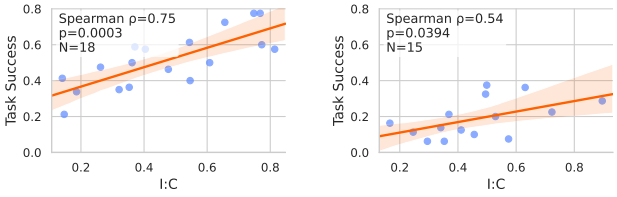
We evaluate 11 LLMs, including those with thinking mode, which generate intermediate reasoning traces, and LLMs without it. Decoding parameters, post-processing rules, and judge prompts are in Appendix C and we manually validate LLM-as-a-judge performance in Appendix J.

LLMs (standard inference) Qwen3-8B, Qwen3-14B, and Qwen3-32B [42]; Llama-3.3-70B-Instruct [24]; and GPT-4o [29]. For Qwen3 models, we disable thinking by setting `enable_thinking=False` in the model's chat template.

LLMs with Thinking Mode Gemini-2.5-Pro [10]; Qwen3-8B, Qwen3-14B, and Qwen3-32B [42]; QwQ-32B [33]; and DeepSeek-R1-Distill-Llama-70B [12]. Thinking-mode models generate reasoning traces before an answer. For Qwen3 models, we enable thinking by setting `enable_thinking=True` in the model's chat template. Gemini-2.5-Pro, QwQ-32B, and DeepSeek-R1-Distill-Llama-70B generate reasoning as part of their default inference behavior.

4.3 Results

Table 2 presents the experimental results in eleven LLMs. We see larger model variants are largely more successful within the same model family. In the first three row blocks, we can compare three models (Qwen3-8B, 14B, 32B) in thinking vs. non-thinking mode. We find that thinking mode largely improves task success rates over non-thinking mode. However, the number of turns to success is not always fewer with thinking models; potentially, they explore more to identify harder rules, which non-thinking models fail to guess. We also find that switching non-thinking to thinking mode does not come with increase in I:C, indicating that gains from thinking mode are not driven by reduced confirmation bias.



(a) Thinking models (b) Non-thinking models

Figure 2: **Confirmation bias correlates with task success.** Higher I:C (more disconfirmatory testing) is associated with higher task success. Each point represents a (model, variant) averaged over 80 episodes, and shaded regions show 95% confidence intervals.

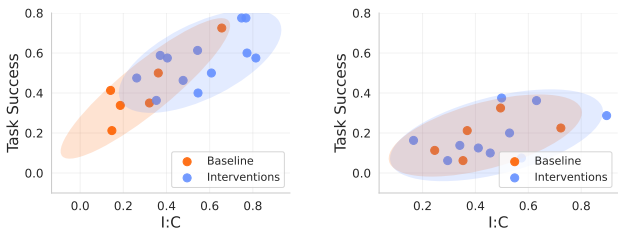
Additional diagnostics, such as first attempt guess rate and results broken down by rule category, are in Appendix D. We verify that re-prompting frequency does not affect results in Appendix K.1, that task success reflects genuine exploration quality rather than prior alignment in Appendix K.2, and that the model's announcement generally reflects its current hypothesis in Appendix K.3.

Table 2: **LLMs task performance and the extent of confirmation bias (I:C ratio) on Wason's rule discovery game.** We observe varying success rates for LLMs. Overall, larger models with thinking mode show higher task success rates. TiO represents the Think-in-Opposites intervention.

	Standard Inference				Thinking Mode Inference			
	Task Suc. % ↑	**Avg. # Turns until Suc. ↓**	**Avg. # Tkns per Turn ↓**	**I:C ↑**	**Task Suc. % ↑**	**Avg. # Turns until Suc. ↓**	**Avg. # Tkns per Turn ↓**	**I:C ↑**
	Qwen3-8B				**Qwen3-8B**			
Baseline	0.06	5.00	8.4	0.35	0.21	10.88	1088.2	0.15
TiO	0.06	15.25	8.4	0.30	0.36	9.01	1716.1	0.35
Dual-Goal	0.08	0.00	13.4	0.57	0.40	7.37	2142.7	0.55
	Qwen3-14B				**Qwen3-14B**			
Baseline	0.23	3.00	7.6	0.72	0.34	7.39	1179	0.19
TiO	0.29	4.12	7.0	0.90	0.48	7.08	1332.4	0.26
Dual-Goal	0.13	0.50	15.0	0.41	0.58	7.66	1191.7	0.40
	Qwen3-32B				**Qwen3-32B**			
Baseline	0.33	4.93	6.6	0.50	0.41	10.83	1029.8	0.14
TiO	0.36	3.76	6.1	0.63	0.61	6.38	1450.0	0.54
Dual-Goal	0.10	0.00	15.4	0.46	0.60	10.29	1472.3	0.77
	LLaMA-3.3-70B-Ins				**DeepSeek-R1-LLaMA-70B**			
Baseline	0.21	9.37	14.1	0.37	0.35	8.10	957.6	0.32
TiO	0.38	10.03	14.7	0.50	0.46	10.07	1072.8	0.48
Dual-Goal	0.20	12.87	23.7	0.53	0.50	8.39	1211.2	0.61
	GPT-4o				**QwQ-32B**			
Baseline	0.11	0.00	10.4	0.25	0.50	11.75	1757.5	0.36
TiO	0.16	2.50	10.7	0.17	0.59	10.28	1464.5	0.37
Dual-Goal	0.14	6.75	17.1	0.34	0.58	10.76	2277.2	0.81
	–				**Gemini-2.5-Pro**			
Baseline	–	–	–	–	0.73	11.14	–	0.66
TiO	–	–	–	–	0.78	8.52	–	0.75
Dual-Goal	–	–	–	–	0.78	10.10	–	0.77

Lower confirmation bias correlates with higher task success Figure 2 plots I:C ratios (x-axis) against task success rates (y-axis), separately for thinking and non-thinking models. We observe statistically significant positive correlations in both cases, with a stronger trend for thinking models. Overall, this indicates that reduced confirmation bias is associated with higher task success, suggesting that mitigating confirmation bias in LLMs can improve its performance.

Impact of interventions Overall, the intervention strategies improves task success. Dual-Goal improves task success in eight out of eleven model–inference scenarios, while Think-in-Opposites improves task success in all. Both interventions tend to increase I:C ratios (18 out of 22 cases). Figure 3 visualizes these trends.



(a) Thinking models (b) Non-thinking models

Figure 3: **Interventions shift exploration in thinking but not non-thinking models.** Red denotes Baseline; blue denotes intervention runs (Dual-Goal / Think-in-Opposites). Ellipses show 95% covariance regions. *Each point represents a (model, variant) averaged over 80 episodes.*

Statistical significance is assessed using paired permutation tests with 50,000 permutations, with p-values adjusted via Benjamini–Hochberg correction; more details in Appendix I.1. We found the improvements to be statistically significant for thinking models, whereas for non-thinking models the results are mixed (no significant change for I:C, small improvement in accuracy for TiO and small drop in accuracy for Dual-Goal).

5 Injecting Intervention-Guided Reasoning into LLMs

Having observed that the intervention prompt improves the LLM's task performance by reducing confirmation bias, we study how to internalize this without providing it at inference time as a prompt. We use a symbolic knowledge distillation approach [40]. We generate input-output pairs from settings

that exhibit lower confirmation bias and higher task success rate: outputs produced under intervention prompts and outputs from larger teacher models. We perform supervised fine-tuning on that data. We describe the set-up below.

Distillation setup To target exploration behavior rather than rule-guessing accuracy, we train on `Test` **turns** produced by a teacher, including unsolved episodes and solved ones. Each training example consists of the dialogue history up to a `Test` turn and the teacher's next-turn response, which includes reasoning tokens.

More formally, let x_t denote the dialogue history up to the t-th `Test` turn. Let $T_{\text{teacher}}(x_t)$ and $T_{\text{student}}(x_t)$ denote the prompts used for the teacher and student, respectively. The teacher model p_{θ_T} produces a next-turn response $y^T = (y_1^T, \ldots, y_{|y^T|}^T)$ under $T_{\text{teacher}}(x_t)$. For distillation, the student model p_θ is conditioned on the same dialogue history formatted with $T_{\text{student}}(x_t)$ and trained to imitate the teacher output token by token. Specifically, we optimize the log-likelihood:

$$\mathcal{L}_{\text{SFT}}(\theta) = -\mathbb{E}_{(x_t, y^T)\sim\mathcal{D}} \sum_{i=1}^{|y^T|} \log p_\theta\left(y_i^T \mid T_{\text{student}}(x_t), y_{<i}^T\right)$$

We construct the distillation dataset using the train/validation/test split defined in Section 3. Teacher-generated episodes from the training split yield $4 \times 4 \times 100 = 1600$ episodes, each with 45 `Test` turns, resulting in 72,000 training examples. Validation uses teacher-generated episodes from the validation split, yielding $16 \times 45 = 720$ validation examples. Training details are in Appendix E.

Teacher–student Set-ups We focus on Think-in-Opposites for distillation, as Dual-Goal alters the output format (DAX/MED rules). We study the following teacher–student configurations:

- **Intervention:** a model distilling from its own outputs generated with Think-in-Opposites prompts.
- **Cross-scale:** a student distilling from a larger teacher model, with the same baseline prompt.
- **Cross-scale and Intervention:** a smaller student distilled from a larger model using Think-in-Opposites prompts.

Results Figure 4 shows the distillation performance (full results in Appendix G.1). We observe that intervention distillation improves task success within the same base model, raising it to a comparable performance to its teacher setting.

Cross-scale distillation improves both task performance and I:C ratio. Unsurprisingly, the largest gain was achieved from cross-scale, intervention distillation, almost tripling the task success rates and more than doubling the I:C ratio. Overall, we report evidence of within-task generalization across different rules.

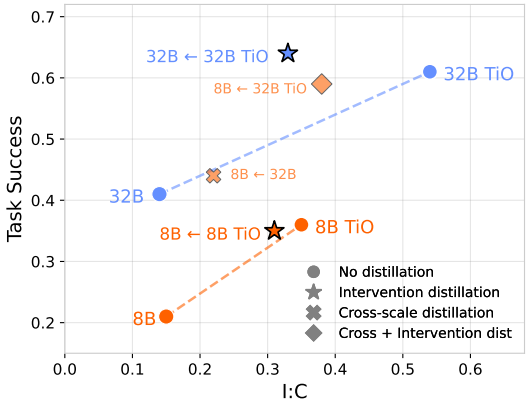


Figure 4: Distillation results for Qwen3 models on the Wason task. Distillation is denoted as *Student* ← *Teacher*. Red: 8B student; Blue: 32B student.

We further perform a sanity-check whether this distillation hurts other abilities of LLMs, by evaluating it on a mathematical reasoning benchmark (GSM8K [9]). We observe little (at most 2%) or no degradation (results in Appendix G.3). We next investigate whether these distilled models generalize to a new task where LLMs suffer from confirmation bias.

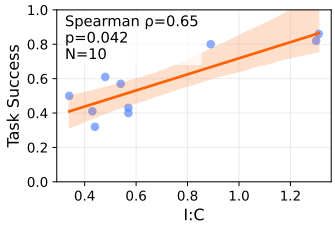
6 Generalization to a New Task Domain: Blicket Test

We study the Blicket Test environment from GX-Chen et al. [16], another task that requires hypothesis testing. Designed by developmental psychologists to test children's ability for causal reasoning [15], this setting has recently been adapted into text-based games to evaluate LLMs [32, 16].

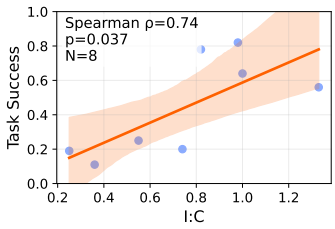
Task It involves N objects and a "Blicket detector" machine. A subset of objects (the "Blickets") activates the machine according to an unobserved rule: under the OR rule, the machine turns on when

Table 3: **LLM Task performance in the Blicket Task.**

	Non-Thinking			Thinking		
	Task Suc. % ↑	**# Turns until Suc. ↓**	**I:C ↑**	**Task Suc. % ↑**	**# Turns until Suc. ↓**	**I:C ↑**
	Qwen3-8B			**Qwen3-8B**		
Baseline	0.11	10.24	0.36	0.32	10.26	0.44
TiO	0.19	12.14	0.25	0.40	10.73	0.57
	Qwen3-14B			**Qwen3-14B**		
Baseline	0.25	10.38	0.55	0.50	9.48	0.34
TiO	0.20	10.73	0.74	0.61	12.33	0.48
	Qwen3-32B			**Qwen3-32B**		
Baseline	0.64	14.00	1.00	0.57	12.18	0.54
TiO	0.56	13.89	1.33	**0.86**	12.98	**1.31**
	LLaMA-3.3-70B-Ins			**DeepSeek-R1-LLaMA-70B**		
Baseline	0.78	12.91	0.82	0.41	12.93	0.43
TiO	0.82	13.50	0.98	0.43	17.41	0.57
	–			**QwQ-32B**		
Baseline	–	–	–	0.80	12.27	0.89
TiO	–	–	–	0.82	14.68	1.30



(a) Thinking models



(b) Non-thinking models

Figure 5: **Correlation between the extent of confirmation bias and task success.**

any Blicket object is placed on it, while under the AND rule, it turns on only when *all* Blicket objects are placed on it. This tests an agent's ability to perform exploration and causal discovery.

We adapt the Blicket test to our interactive hypothesis-testing framework (see Appendix F.1 for comparison with the original experiment). Each **episode** starts with an initial object placement and the machine being ON or OFF, followed by 45 alternating `Guess` and `Test` turns, mirroring our Wason setup. On `Test` turns, the model selects objects to place on the machine and receives feedback whether it turns "ON" or "OFF". On `Guess` turns, the model guesses the blickets and hidden rule; no feedback is provided. A correct guess must identify both the full set of blickets and the correct rule. We evaluate a Baseline prompt and an adapted Think-in-Opposites (TiO) prompt (provided in Appendix F.3 with example interactions). We do not evaluate Dual-Goal because it requires an opposite hypothesis, while a Blicket hypothesis (set of blickets + rule) has no well-defined opposite.

Data We vary the task along three dimensions: (1) number of objects (4 or 8), (2) number of blickets (2 or 3), and (3) hidden rule (AND, OR, or XOR). For each configuration, we generate 16 episodes by varying the blicket assignments and initial object placements. This yields $2 \times 2 \times 3 \times 16 = 192$ evaluation episodes in total. Dataset construction details and statistics are provided in Appendix F.2.

Evaluation Following the evaluation setup in the Wason task (described in Sec 3), we report: (1) task success, (2) # turns until success, and (3) I:C ratio. We keep the same hyperparameters as the Wason task, and modify the LLM-as-a-Judge prompt (provided in Appendix F.4 with examples).

Does confirmation bias correlate negatively with task success in this domain? We first evaluate whether the relationship between confirmation bias and task performance observed in the Wason task generalizes to the Blicket test. Table 3 reports overall performance across models, and Figure 5 plots I:C ratios against task success. We observe significant positive correlations between I:C and task success for both thinking and non-thinking models, indicating that models that show reduced confirmation bias tend to solve the task more successfully. Finally, we find that lower confirmation bias is associated with better task performance overall. This trend is visible for both thinking and non-thinking models, and is slightly stronger among non-thinking models in terms of rank correlation.

Does the intervention strategy to reduce confirmation bias enhance task performance in this domain? We next evaluate whether Think-in-Opposites (TiO) improves performance in the Blicket task. Across thinking models, TiO increases both task success and I:C on average, and these improvements are statistically significant under paired permutation tests (Appendix I.1). For non-thinking models, the effects are substantially smaller.

7 Generalization of Distilled Model

We evaluate four models: base Qwen3-8B and Qwen3-32B, and these models further fine-tuned on outputs generated with Think-in-Opposites prompts (Qwen3-8B + 8B TiO and Qwen3-32B + 32B TiO) from Section 5.

7.1 Generalization to Blicket Test

Would training to decrease confirmation bias in the number game generalize to the new object-based domain? Figure 6 shows Blicket performance for models distilled on the Wason task, with base model results shown for comparison (full results in Appendix G.2). Same-scale intervention distillation (8B TiO $\rightarrow$ 8B student; 32B TiO $\rightarrow$ 32B student) yields improvements over the corresponding base models, with statistical test results provided in Appendix I.2. However, performance remains below applying TiO prompting in the Blicket task. This suggests that falsification-oriented exploration learned in the Wason task transfers to a new domain and leads to measurable gains, although it does not fully match the improvements from task-specific intervention prompts.

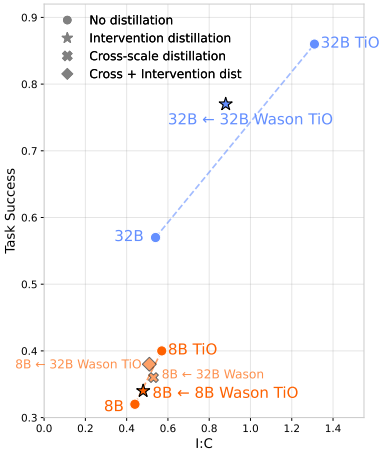


Figure 6: Blicket task performance for **Qwen3 models** distilled on the Wason task. Distillation is denoted as *Student* $\leftarrow$ *Teacher*.

7.2 Generalization to Mathematical Self Correction

We next ask whether this behavior generalizes beyond rule discovery. In mathematical reasoning, a model given another attempt after an incorrect answer may repeat its original reasoning rather than explore an alternative solution path, arriving at the same incorrect answer. This behavior is analogous to confirmation bias, where an agent seeks evidence supporting an incorrect hypothesis rather than testing alternatives. We therefore evaluate whether models distilled on the Wason task are better able to revise their solutions after being told their previous answer was incorrect.

Setting We evaluate on AIME 2025 [30] and Beyond AIME [6], two mathematical reasoning benchmarks. Their problems are sufficiently challenging that models often fail on the first attempt, creating opportunities for iterative revision. For each problem, the model first attempts a solution. If its answer is incorrect, we retain the conversation history, inform the model that its previous answer was incorrect, and ask it to review its previous reasoning and try again for up to five times. Full prompt and evaluation details are provided in Appendix H.

Results Figure 7 shows that, for the 8B model, the distilled version improves final accuracy over the base model on both benchmarks (0.60 vs. 0.33 on AIME 2025; 0.40 vs. 0.30 on Beyond AIME). Using a two-sided exact McNemar test on final solved/unsolved outcomes, this improvement is statistically significant on both ($p = 0.0386$ and $p = 0.0414$, respectively). For the 32B model, there are smaller gains (0.80 vs. 0.73 on AIME 2025; 0.54 vs. 0.52 on Beyond AIME), but these are not statistically significant ($p = 0.6875$ and $p = 0.7539$, respectively), likely reflecting less room for self-correction at the higher baseline accuracy. These results suggest that the debiasing signal learned on the Wason task carries over to more effective self-correction in math.

8 Related Work

Adapting tools from cognitive science for identifying cognitive biases in humans can be an effective approach for identifying similar issues in LLMs. LLMs have been shown to reflect a wide range of biases found in humans, such as content effects [11], threshold priming [8], overthinking [19], overestimating rationality [18], forming implicit associations [2], linguistic illusions [21], skewed representations of numbers [22], conformity under social pressure [44], and causal reasoning [16].

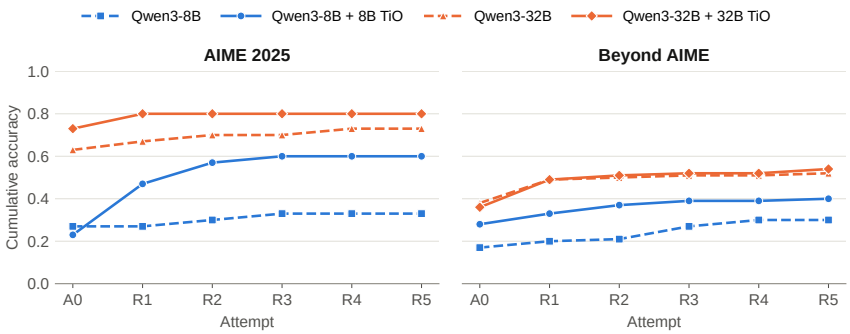


Figure 7: **Mathematical reasoning accuracy across retries.** Accuracy is cumulative: at each retry, a problem is counted as solved if the model has produced the correct answer in any attempt up to that point. A0 denotes the initial attempt (Attempt 0), and R1–R5 denote Retries 1–5. Distilled models (Qwen3-8B + 8B TiO, Qwen3-32B + 32B TiO) were trained on the Wason task.

Confirmation bias is one of well-established cognitive biases in humans, from seminal Wason's 2-4-6 rule-discovery task [39]. In the subsequent decades, cognitive scientists have studied identifying and mitigating confirmation bias in humans [14, 5]. Recent work has explored confirmation bias in LLMs as part of broader bias evaluations [20], in chain-of-thought reasoning [37], in mitigating hallucinations [28], in LLM-assisted security code review [25], and in relation to framing and position biases [13]. However, these studies primarily examine confirmation bias at the level of stance or answer generation—for example, whether a model favors supporting over contradicting arguments once a claim or hypothesis has already been presented. Such settings measure confirmation bias in evaluative reasoning.

In contrast, we study confirmation bias during the exploration process. Our setting requires models to iteratively generate hypotheses, select which evidence to test, and revise beliefs based on environment feedback. Here, confirmation bias manifests not in how evidence is interpreted, but in how evidence is selected. This distinction is critical for agentic systems that must autonomously explore hypothesis spaces, such as in AI-driven scientific discovery [34, 41, 27, 7, 1].

Related work has studied hypothesis exploration through experiment design. Piriyakulkij et al. [32] use expected information gain to identify informative tests given a distribution over hypotheses, requiring an explicit hypothesis space and posterior.

9 Conclusion

We adapt the experimental settings from cognitive psychology to study confirmation bias in LLMs during interactive hypothesis exploration. Introducing a controlled rule-discovery framework and the I:C metric, we show that all 11 LLMs we evaluate favor confirmatory tests, and that reducing this bias leads to gains in rule discovery. Distilled models further generalize this falsification-oriented behavior to a new rule-discovery domain without additional prompting and beyond rule discovery to mathematical self-correction, suggesting that debiasing can be internalized at the parameter level.

Limitations and Future Work Our framework shows how to measure and mitigate confirmation bias with a focus on rule-discovery settings.

While we find that the benefits of distillation extend beyond rule discovery to mathematical self-correction, an important direction for future work is evaluating whether similar improvements generalize to other real-world settings, such as scientific discovery, question answering, and agentic tasks involving planning, tool use, and debugging.

Acknowledgements and Disclosures

We thank Nishant Balepur, Kristina Fujimoto, and John (Yueh-Han) Chen for their feedback. This work was supported in part through the NYU IT High Performance Computing resources, services, and staff expertise. The work is partially funded by NSF CAREER award 2443271. We also thank Google for providing Gemini API credits. The authors declare no competing interests.

References

[1] Eser Aygün, Anastasiya Belyaeva, Gheorghe Comanici, Marc Coram, Hao Cui, Jake Garrison, Renee Johnston Anton Kast, Cory Y McLean, Peter Norgaard, Zahra Shamsi, et al. An ai system to help scientists write expert-level empirical software. *arXiv preprint arXiv:2509.06503*, 2025.

[2] Xuechunzi Bai, Angelina Wang, Ilia Sucholutsky, and Thomas L Griffiths. Explicitly unbiased large language models still form biased associations. *Proceedings of the National Academy of Sciences*, 122(8):e2416228122, 2025.

[3] Eryk Banatt, Jonathan Cheng, Skanda Vaidyanath, and Tiffany Hwu. Wilt: A multi-turn, memorization-robust inductive logic benchmark for llms. *arXiv preprint arXiv:2410.10998*, 2024.

[4] Eric J. Bigelow and Steven T. Piantadosi. Large Dataset of Generalization Patterns in the Number Game, 2018. URL `https://doi.org/10.7910/DVN/A8ZWLF`.

[5] Erika Branchini, Roberto Burro, and Ivana Bianchi. Training people to think in opposites facilitates the falsification process in wason's rule discovery task. *Journal of Intelligence*, 11:91, 05 2023. doi: 10.3390/jintelligence11050091.

[6] ByteDance-Seed. BeyondAIME: Advancing Math Reasoning Evaluation Beyond High School Olympiads. `https://huggingface.co/datasets/ByteDance-Seed/BeyondAIME`, 2025.

[7] Pablo Samuel Castro, Nenad Tomasev, Ankit Anand, Navodita Sharma, Rishika Mohanta, Aparna Dev, Kuba Perlin, Siddhant Jain, Kyle Levin, Noémi Éltető, et al. Discovering symbolic cognitive models from human and animal behavior. *bioRxiv*, pages 2025–02, 2025.

[8] Nuo Chen, Jiqun Liu, Xiaoyu Dong, Qijiong Liu, Tetsuya Sakai, and Xiao-Ming Wu. Ai can be cognitively biased: An exploratory study on threshold priming in llm-based batch relevance assessment. In *Proceedings of the 2024 Annual International ACM SIGIR Conference on Research and Development in Information Retrieval in the Asia Pacific Region*, SIGIR-AP 2024, page 54–63, New York, NY, USA, 2024. Association for Computing Machinery. ISBN 9798400707247. doi: 10.1145/3673791.3698420. URL `https://doi.org/10.1145/3673791.3698420`.

[9] Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

[10] Gheorghe Comanici, Eric Bieber, Mike Schaekermann, Ice Pasupat, Noveen Sachdeva, Inderjit Dhillon, Marcel Blistein, Ori Ram, Dan Zhang, Evan Rosen, et al. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. *arXiv preprint arXiv:2507.06261*, 2025.

[11] Ishita Dasgupta, Andrew K Lampinen, Stephanie CY Chan, Antonia Creswell, Dharshan Kumaran, James L McClelland, and Felix Hill. Language models show human-like content effects on reasoning. *arXiv preprint arXiv:2207.07051*, 2(3), 2022.

[12] DeepSeek-AI. Deepseek-r1: Incentivizing reasoning capability in large language models. `https://huggingface.co/deepseek-ai/DeepSeek-R1-Distill-Llama-70B`, 2025. Model card and technical report.

[13] Liana Ermakova, Anton Firsov, and Jaap Kamps. Confirmation, framing, and position biases in llm responses. In *Proceedings of the 2026 Conference on Human Information Interaction and Retrieval*, pages 480–484, 2026.

[14] Maggie Gale and Linden Ball. Dual-goal facilitation in wason's 2-4-6 task: What mediates successful rule discovery? *Quarterly journal of experimental psychology (2006)*, 59:873–85, 06 2006. doi: 10.1080/02724980543000051.

[15] Alison Gopnik and David M Sobel. Detecting blickets: How young children use information about novel causal powers in categorization and induction. *Child Development*, 2000.

[16] Anthony GX-Chen, Dongyan Lin, Mandana Samiei, Doina Precup, Blake Aaron Richards, Rob Fergus, and Kenneth Marino. Language agents mirror human causal reasoning biases. how can we help them think like scientists? In *Second Conference on Language Modeling*, 2025.

[17] Joshua Klayman and Young-Won Ha. Confirmation, disconfirmation, and informa-tion in hypothesis testing. 1987. URL https://api.semanticscholar.org/CorpusID:10853196.

[18] Ryan Liu, Jiayi Geng, Joshua Peterson, Ilia Sucholutsky, and Thomas L. Griffiths. Large language models assume people are more rational than we really are. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=dAeET8gxqg.

[19] Ryan Liu, Jiayi Geng, Addison J. Wu, Ilia Sucholutsky, Tania Lombrozo, and Thomas L. Griffiths. Mind your step (by step): Chain-of-thought can reduce performance on tasks where thinking makes humans worse. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=J3gzdbYZxS.

[20] Simon Malberg, Roman Poletukhin, Carolin Schuster, and Georg Groh Groh. A comprehensive evaluation of cognitive biases in llms. In *Proceedings of the 5th International Conference on Natural Language Processing for Digital Humanities*, pages 578–613, 2025.

[21] Raja Marjieh, Pol van Rijn, Ilia Sucholutsky, Harin Lee, Thomas L Griffiths, and Nori Jacoby. A rational analysis of the speech-to-song illusion. *arXiv preprint arXiv:2402.06992*, 2024.

[22] Raja Marjieh, Veniamin Veselovsky, Thomas L Griffiths, and Ilia Sucholutsky. What is a number, that a large language model may know it? *arXiv preprint arXiv:2502.01540*, 2025.

[23] Michael McCloskey and Neal J. Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. volume 24 of *Psychology of Learning and Motivation*, pages 109–165. Academic Press, 1989. doi: https://doi.org/10.1016/S0079-7421(08)60536-8. URL https://www.sciencedirect.com/science/article/pii/S0079742108605368.

[24] Meta AI. Meta llama 3.3 70b instruct. https://huggingface.co/meta-llama/Llama-3.3-70B-Instruct, 2024. Pretrained instruction-tuned LLM, released Dec 6 2024.

[25] Dimitris Mitropoulos, Nikolaos Alexopoulos, Georgios Alexopoulos, and Diomidis Spinellis. Measuring and exploiting confirmation bias in llm-assisted security code review. *arXiv preprint arXiv:2603.18740*, 2026.

[26] Raymond S. Nickerson. Confirmation bias: A ubiquitous phenomenon in many guises. *Review of General Psychology*, 2:175 – 220, 1998. URL https://api.semanticscholar.org/CorpusID:8508954.

[27] Alexander Novikov, Ngân Vũ, Marvin Eisenberger, Emilien Dupont, Po-Sen Huang, Adam Zsolt Wagner, Sergey Shirobokov, Borislav Kozlovskii, Francisco JR Ruiz, Abbas Mehrabian, et al. Alphaevolve: A coding agent for scientific and algorithmic discovery. *arXiv preprint arXiv:2506.13131*, 2025.

[28] Changwon Ok, Eunkyeong Lee, and Dongsuk Oh. Synthetic paths to integral truth: Mitigating hallucinations caused by confirmation bias with synthetic data. In *Proceedings of the 31st international conference on computational linguistics*, pages 5168–5180, 2025.

[29] OpenAI. Gpt-4o technical overview. https://platform.openai.com/docs/models/gpt-4o, 2025. Accessed: 2026-01.

[30] OpenCompass. AIME 2025. https://huggingface.co/datasets/opencompass/AIME2025, 2025.

[31] Daniel E O'Leary. Confirmation and specificity biases in large language models: An explorative study. *IEEE Intelligent Systems*, 40(1):63–68, 2025.

[32] Top Piriyakulkij, Cassidy Langenfeld, Tuan Anh Le, and Kevin Ellis. Doing experiments and revising rules with natural language and probabilistic reasoning. *Advances in Neural Information Processing Systems*, 37:53102–53137, 2024.

[33] Qwen Team. Qwq-32b: Reasoning-enhanced large language model. https://huggingface.co/Qwen/QwQ-32B, 2024. Model card and technical description.

[34] Bernardino Romera-Paredes, Mohammadamin Barekatain, Alexander Novikov, Matej Balog, M Pawan Kumar, Emilien Dupont, Francisco JR Ruiz, Jordan S Ellenberg, Pengming Wang, Omar Fawzi, et al. Mathematical discoveries from program search with large language models. *Nature*, 625(7995):468–475, 2024.

[35] Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. *Advances in Neural Information Processing Systems*, 36:8634–8652, 2023.

[36] Joshua Tenenbaum. Rules and similarity in concept learning. In S. Solla, T. Leen, and K. Müller, editors, *Advances in Neural Information Processing Systems*, volume 12. MIT Press, 1999. URL https://proceedings.neurips.cc/paper_files/paper/1999/file/86d7c8a08b4aaa1bc7c599473f5dddda-Paper.pdf.

[37] Yue Wan, Xiaowei Jia, and Xiang Lorraine Li. Unveiling confirmation bias in chain-of-thought reasoning. *arXiv preprint arXiv:2506.12301*, 2025.

[38] Lei Wang, Chen Ma, Xueyang Feng, Zeyu Zhang, Hao Yang, Jingsen Zhang, Zhiyuan Chen, Jiakai Tang, Xu Chen, Yankai Lin, et al. A survey on large language model based autonomous agents. *Frontiers of Computer Science*, 18(6):186345, 2024.

[39] Peter C Wason. On the failure to eliminate hypotheses in a conceptual task. *Quarterly journal of experimental psychology*, 12(3):129–140, 1960.

[40] Peter West, Chandra Bhagavatula, Jack Hessel, Jena D. Hwang, Liwei Jiang, Ronan Le Bras, Ximing Lu, Sean Welleck, and Yejin Choi. Symbolic knowledge distillation: from general language models to commonsense models, 2022. URL https://arxiv.org/abs/2110.07178.

[41] Yutaro Yamada, Robert Tjarko Lange, Cong Lu, Shengran Hu, Chris Lu, Jakob Foerster, Jeff Clune, and David Ha. The ai scientist-v2: Workshop-level automated scientific discovery via agentic tree search. *arXiv preprint arXiv:2504.08066*, 2025.

[42] An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

[43] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. In *The eleventh international conference on learning representations*, 2022.

[44] Xiaochen Zhu, Caiqi Zhang, Tom Stafford, Nigel Collier, and Andreas Vlachos. Conformity in large language models. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3854–3872, 2025.

A Dataset Construction Details

A.1 Prompt for Rule Generation

We generate rule groups using a large language model prompted to propose relational hypotheses over integer triples $(a, b, c) \in [-99, 100]^3$. The model is instructed to generate **ten rule groups** in a single response, where each group contains **four distinct rules**. All four rules within a group are required to share a non-empty feasible intersection (i.e., at least one triple satisfies all four rules).

You are generating relational rules over integer triples (a, b, c) for a reasoning task.

Each rule should describe a simple, interpretable relation or shared property among the three integers. Avoid unary conditions that describe only one of the numbers (e.g., "b is even" or "a is positive"). Involve all three variables jointly — for example, patterns or equations that connect a, b, and c together — rather than isolated or purely pairwise comparisons.

Rules may belong to any of the following families:
- Ordering (ascending, descending, non-decreasing, non-monotone, etc.)
- Arithmetic structure (equal differences, geometric relations, increasing or decreasing gaps, sum relationships, etc.)
- Parity or divisibility (even/odd, divisibility chains, modular patterns, etc.)
- Sign (all positive, all negative, mixed signs)
- Number properties (multiples, primes, squares, or related characteristics)
- Comparative balance (averages, sums, or ratios involving all three numbers)
- Symmetry or modular relations (mirror patterns, alternating parities, repeating residues)

Do not use rules that depend on numeric ranges (e.g., "between 10 and 50" or "less than 100").

Each rule should naturally vary in both:
- Breadth: the proportion of triples in $[-99, 100]^3$ that satisfy it.
- Rule-guess difficulty: how easily a human or model might infer it.

Now construct a table of rule groups as follows:
- Each group contains four distinct rules.
- All four rules in a group must have a non-empty feasible intersection.
- All rules across all groups must be distinct (no repetitions or paraphrases).
- Rules in a group may belong to different families.
- Avoid groups where all rules are extremely narrow; ensure at least moderate feasible overlap.

Generate 10 groups of 4 rules each. Output only the Markdown table, no LaTeX or math mode.

Rules depending on explicit numeric ranges are disallowed, as they introduce arbitrary constraints rather than abstract relational structure.

A.2 Post-filtering and Revision

After automatic generation, we apply structured post-filtering and revision. A rule group is rejected and repaired if it violates any of the following criteria:

1. **Insufficient feasible intersection:** the feasible intersection set contains fewer than five valid triples.
2. **Unary dependence:** the rule constrains only a single variable (e.g., "c is prime"). Conjunctive properties applied uniformly to all three variables (e.g., "all odd") are retained.
3. **Excessive complexity:** the rule requires multi-step arithmetic dependencies unlikely to arise during hypothesis testing.

When a group fails any criterion, we re-prompt the language model to replace only the offending rule(s) while keeping the remaining rules fixed. This process is repeated until all constraints are satisfied. Approximately one-third of initially generated groups required at least one revision.

After filtering, we randomly shuffle the order of the rule groups using a fixed seed (1337), without modifying the rules within each group.

A.3 Human Rule Injection

To incorporate human-like hypotheses, we inject one human-derived rule into each rule group. Human rules are drawn from prior studies on number generalization, including the Large Dataset

of Generalization Patterns in the Number Game [4], which builds on Tenenbaum's Number Game paradigm [36].

For each unique triple in the source dataset, multiple human participants provided pattern descriptions. We select the most frequent description per triple and remove hypotheses referring to explicit numeric ranges. From the remaining pool, we curate ten interpretable human hypotheses: *even*, *odd*, *prime*, *cube*, *ends with 1*, *ends with 6*, *ends with 9*, *divisible by 3*, *divisible by 4*, and *divisible by 5*.

For each rule group, we replace one LLM-generated rule with a selected human-derived rule. The replacement is accepted only if the resulting group maintains a non-empty feasible intersection. Human-derived rules are indicated explicitly in Table 7.

A.4 Feasible Intersection and Triple Sampling

For each rule group g with rules $\{r_{g,i}\}_{i=1}^{4}$, we define the feasible set $S_g = \{(a, b, c) \in [-99, 100]^3 \mid r_{g,i}(a, b, c) = 1 \; \forall i\}$.

We compute $|S_g|$ by enumerating all triples in $[-99, 100]^3$ and evaluating all four rule predicates. The feasible set size for each group is reported in Table 7.

We uniformly sample triples from S_g using a fixed random seed (1337). Each sampled triple serves as the initial triple for four reasoning episodes, one per rule in the corresponding group treated as the target rule.

A.5 Dataset Splits and Scenario Counts

We divide the ten rule groups into **4 Train**, **2 Validation**, and **4 Test** groups. Training uses 100 triples per group, yielding $4 \times 100 \times 4 = 1600$ training scenarios. Validation uses 2 triples per group, yielding $2 \times 2 \times 4 = 16$ validation scenarios. Testing uses 5 triples per group; since each triple is evaluated against all four target rules, this yields $4 \times 5 \times 4 = 80$ test scenarios.

Rule definitions, executable predicates, rule groupings, feasible intersection sizes, and sampled test instances are provided in Tables 4–8.

Table 4: Rules (Train).

Rule	**Predicate (over a,b,c)**
All even	`a%2==0 and b%2==0 and c%2==0`
Each divides next	`a!=0 and (b%a==0) and b!=0 and (c%b==0)`
Exactly two equal	`len({a,b,c})==2`
At least one even	`(a%2==0) or (b%2==0) or (c%2==0)`
All end with 6	`abs(a)%10==6 and abs(b)%10==6 and abs(c)%10==6`
Increasing differences	`0 < (b-a) < (c-b)`
a is min	`a<=b and a<=c`
All distinct	`len({a,b,c})==3`
All divisible by 5	`a%5==0 and b%5==0 and c%5==0`
a is max	`a>=b and a>=c`
Non-monotone (middle between ends)	`(a-b)*(b-c)<0`
At least two multiples of 5	`((a%5==0)+(b%5==0)+(c%5==0))>=2`
All divisible by 3	`a%3==0 and b%3==0 and c%3==0`
Alternating parity (ends same)	`(a%2)==(c%2) and (b%2)!=(a%2)`
Ascending	`a<b<c`
Non-decreasing	`a<=b<=c`

Table 5: Rules (Validation).

Rule	Predicate (over a,b,c)
All end with 9	`abs(a)%10==9 and abs(b)%10==9 and abs(c)%10==9`
Non-decreasing differences	`(b-a) <= (c-b)`
c is max	`c>=a and c>=b`
Arithmetic progression (AP)	`(b-a)==(c-b)`
All divisible by 7	`a%7==0 and b%7==0 and c%7==0`
Exactly two even	`((a%2==0)+(b%2==0)+(c%2==0))==2`
At least one multiple of 4	`(a%4==0) or (b%4==0) or (c%4==0)`
At least two distinct	`len({a,b,c})>=2`

Table 6: Rules (Test).

Rule	Predicate (over a,b,c)
All end with 1	`abs(a)%10==1 and abs(b)%10==1 and abs(c)%10==1`
c is min	`c<=a and c<=b`
All negative	`a<0 and b<0 and c<0`
Descending	`a>b>c`
All odd	`a%2==1 and b%2==1 and c%2==1`
b is (strict) max	`b>a and b>c`
Mixed signs	`((a<0)+(b<0)+(c<0)) in {1,2}`
At least one multiple of 3	`(a%3==0) or (b%3==0) or (c%3==0)`
All prime numbers	`is_prime(a) and is_prime(b) and is_prime(c)`
Non-increasing	`a>=b>=c`
All positive	`a>0 and b>0 and c>0`
Contains a prime	`is_prime(a) or is_prime(b) or is_prime(c)`
All cube numbers	`is_cube(a) and is_cube(b) and is_cube(c)`
Exactly two odd	`((a%2==1)+(b%2==1)+(c%2==1))==2`
Decreasing gaps	`(b-a)>(c-b)`
At least one odd	`(a%2==1) or (b%2==1) or (c%2==1)`

Table 7: Rule groups and their feasible intersections. The rightmost column reports the number of triples $|S_g|$ in $[-99, 100]^3$ that satisfy all four rules in the group.

ID	Split	Rule 1 (Human)	Rule 2	Rule 3	Rule 4	Feasible $\lvert S_g \rvert$
1	Train	All even	Each divides next	Exactly two equal	At least one even	1,629
2	Train	All end with 6	Increasing differences	a is min	All distinct	5,394
3	Train	All divisible by 5	a is max	Non-monotone (middle between ends)	At least two multiples of 5	128,080
4	Train	All divisible by 3	Alternating parity (ends same)	Ascending	Non-decreasing	194
5	Validation	All end with 9	Non-decreasing differences	c is max	Arithmetic progression (AP)	2,550
6	Validation	All divisible by 7	Exactly two even	At least one multiple of 4	At least two distinct	72
7	Test	All end with 1	c is min	All negative	Descending	1,225
8	Test	All odd	b is (strict) max	Mixed signs	At least one multiple of 3	176,715
9	Test	All prime numbers	Non-increasing	All positive	Contains a prime	3,071
10	Test	All cube numbers	Exactly two odd	Decreasing gaps	At least one odd	205

Table 8: Sampled Test instances (groups 7–10). Each triple is used as a starting point for four episodes, one per rule in the corresponding group.

Split	Group ID	Triple
Test	7	$(-1, -81, -91)$
Test	7	$(-61, -71, -91)$
Test	7	$(-1, -21, -81)$
Test	7	$(-21, -41, -91)$
Test	7	$(-11, -31, -41)$
Test	8	$(-77, 75, -47)$
Test	8	$(-9, 55, -71)$
Test	8	$(-51, 37, -87)$
Test	8	$(-67, 75, 25)$
Test	8	$(-39, 81, 3)$
Test	9	$(97, 67, 37)$
Test	9	$(97, 73, 67)$
Test	9	$(89, 67, 13)$
Test	9	$(79, 43, 31)$
Test	9	$(89, 67, 2)$
Test	10	$(-27, 8, 27)$
Test	10	$(8, 27, -27)$
Test	10	$(-27, 0, 1)$
Test	10	$(-1, 64, -1)$
Test	10	$(-64, 1, -1)$

A.6 Divergence from Original Wason Study

The task setting diverges from the original Wason *2-4-6* study in three ways. First, human participants flexibly choose between testing and guessing, whereas LLMs alternate between `Guess` and `Test` turns. This design is to make it easier to evaluate whether the LLM proposed a triple that is confirmatory or not to the current hypothesis. Second, feedback is provided only on `Test` turns, while `Guess` turns receive no immediate feedback. Third, human sessions terminate upon a correct rule announcement, the participant giving up, or a time limit, whereas LLM episodes run for a fixed number of turns independent of guess correctness.

B Task Instruction Prompts and Examples

This appendix lists the task instructions provided to models in each interaction setting, with example continuations.

B.1 Baseline Prompt

You will be given three numbers which conform to a simple rule that I have in mind. This rule is concerned with the common attributes or relations between any three numbers, and not with their absolute magnitude – for example, it is not a rule like "all numbers above (or below) 50."

Your goal is to discover this hidden rule by proposing sets of three numbers. You will then be told if the triple conforms with the hidden rule or not.

At each turn, you may perform one of two actions:
- Test: propose a new set of three integers, denoted [a,b,c]. You will then be told whether that triple conforms with the hidden rule ("YES" or "NO").
- Announce: based on all previous tests, state what you believe the hidden rule is. You will not receive any feedback for announcements.

Each instruction you receive will specify what to do.

Format (must follow exactly):
- If the instruction is Turn - Announce, output exactly one line: Announce: <one short sentence naming the rule>
- If the instruction is Turn - Test, output exactly one line: Check: [a,b,c]

A triple that conforms with the hidden rule is: [12, -36, -36].
Let's begin.
Turn - Announce.

Example interaction (hidden rule: all numbers are even). *The following transcript continues directly from the final "Turn–Announce" in the initial prompt above.*

```
[model] Announce: The second and third numner are equal.

[environment] Turn - Test

[model] Check: [2, 4, 4]

[environment] YES. Turn - Announce

[model] Announce: The second and third numner are equal.

[environment] Turn - Test

[model] Check: [4, 6, 6]

[environment] YES. Turn - Announce

[model] Announce: The second and third numner are equal.
```

B.2 Dual-Goal Prompt

You will be given three numbers which conform to a simple rule that I have in mind. Triples that fit this rule are called DAX, and triples that do not fit this rule are called MED.

These rules are concerned with the common attributes or relations between any three numbers, and not with their absolute magnitude – for example, they are not rules like "all numbers above (or below) 50."

Your goal is to discover both rules: the rule for DAX triples and the rule for MED triples. You will then be told if the triple conforms with the DAX rule or the MED rule.

At each turn, you may perform one of two actions:
- Test: propose a new set of three integers, denoted [a,b,c]. You will then be told whether that triple conforms with the DAX or MED rule ("DAX" or "MED").
- Announce: based on all previous tests, state what you believe the two hidden rules are. You will not receive any feedback for announcements.

Each instruction you receive will specify what to do.

Format (must follow exactly):
- If the instruction is Turn - Announce, output exactly two lines:
Announce: DAX rule - <one short sentence >
Announce: MED rule - <one short sentence >

- If the instruction is Turn - Test, output exactly one line:
Check: [a,b,c]

A DAX triple is: [12, -36, -36].
Let's begin.
Turn - Announce.

Example interaction (hidden rule: all numbers are even). *The following transcript continues directly from the final "Turn–Announce" in the initial prompt above.*

```
[model] Announce: DAX rule - The second and third number are equal.
Announce: MED rule - The second and third number are not equal.

[environment] Turn - Test

[model] Check: [2, 4, 6]

[environment] DAX. Turn - Announce

[model] Announce: DAX rule - All three numbers are even.
Announce: MED rule - All three numbers are not even.

[environment] Turn - Test

[model] Check: [2, 4, 5]

[environment] MED. Turn - Announce

[model] Announce: DAX rule - All three numbers are even.
Announce: MED rule - All three numbers are not even.
```

B.3 Think-in-Opposites Prompt

You will be given three numbers which conform to a simple rule that I have in mind. This rule is concerned with the common attributes or relations between any three numbers, and not with their absolute magnitude – for example, it is not a rule like "all numbers above (or below) 50."

Your goal is to discover this hidden rule by proposing sets of three numbers. You will then be told if the triple conforms with the hidden rule or not.

Before proposing each new triple, look at the example and think about one feature it has. Then imagine a triple that is opposite with respect to that feature, and test whether it still conforms with the rule. If it does, that feature is probably irrelevant; if it does not, that feature might be crucial.
In short: to discover the rule, test triples that both confirm and contradict your current idea.

At each turn, you may perform one of two actions:
- Test: propose a new set of three integers, denoted [a,b,c]. You will then be told whether that triple conforms with the hidden rule ("YES" or "NO").
- Announce: state what you believe the hidden rule is. Each instruction you receive will specify what to do.

Format (must follow exactly):
- If the instruction is Turn - Announce, output exactly one line: Announce: <one short sentence naming the rule >
- If the instruction is Turn - Test, output exactly one line: Check: [a,b,c]

A triple that conforms with the hidden rule is: [12, -36, -36].
Let's begin.
Turn - Announce.

Example interaction (hidden rule: all numbers are even). *The following transcript continues directly from the final "Turn–Announce" in the initial prompt above.*

```
[model] Announce: The second and third number are equal.

[environment] Turn - Test

[model] Check: [2, 4, 6]

[environment] NO. Turn - Announce

[model] Announce: All three numbers are even.

[environment] Turn - Test

[model] Check: [3, 4, 6]

[environment] NO. Turn - Announce

[model] Announce: All three numbers are even.
```

Format enforcement. If a model response did not follow the required format for the current turn, we automatically re-prompted the model with the format requirement before proceeding; such retries were not counted as turns and were not included in the conversation history.

C Evaluation Implementation Details

This appendix describes the evaluation pipeline used in our experiments, including decoding parameters, post-processing rules, and the LLM-based judge components used to compute task success and confirmation bias metrics. We use two judges: an announcement correctness judge and a compatibility judge.

C.1 Decoding Parameters

We use the following decoding hyperparameters: temperature = 0.6, top_p = 0.95, top_k = 20, and presence penalty = 0.0. Standard models (non-thinking) generate up to 256 new tokens per turn, while thinking-mode models generate up to 16K new tokens per turn. For thinking-mode models, intermediate `<think>` traces are stripped from the dialogue history before the next prompt.

C.2 Post-processing

All models must follow the exact output protocol (`Check: [a,b,c]` or `Announce: ...`). Any formatting violation triggers resampling for the same turn, and the invalid output is not added to the dialogue history.

C.3 Announcement Correctness Judge

Purpose. This judge determines whether a model's announced hypothesis is semantically equivalent to the ground-truth rule. It is used to compute *Task Success* and *# Turns till Success*.

Judge model. We use `Llama-3.3-70B-Instruct` with greedy decoding.

Setup. At each `Announce` turn, the judge is given the ground-truth rule, the model's announcement, and lightweight rule-specific guidance (`{rule_guidance}`). It outputs a binary decision indicating semantic equivalence.

Rule guidance. The `{rule_guidance}` field provides short natural-language *ACCEPT* and *REJECT* criteria describing equivalent statements and common near-misses (broader, narrower, or different rule families).

C.3.1 Correctness Judge Prompt (Baseline / Think-in-Opposites)

You are a strict evaluator of SHORT RULE DESCRIPTIONS for number triples.

You will receive two short natural-language rules:
(1) the model's announced rule, and (2) the ground-truth rule.
Decide if they describe the same underlying rule.

Guidance specific to the ground-truth rule:
{rule_guidance}

Rules:
- Focus on logical meaning, not phrasing.
- If rule families differ (e.g., ordering vs. parity) or one scope is broader/narrower, answer NO.
- Ignore superficial differences (synonyms, word order).

Respond with ONLY one token: YES or NO.

ANNOUNCE TEXT:
"{announced_rule}"

GROUND-TRUTH RULE:
"{ground_truth_rule}"

C.3.2 Correctness Judge Prompt (Dual-Goal)

You are a strict evaluator of SHORT RULE DESCRIPTIONS for number triples.

Context:
- This is the DAX–MED experiment. DAX is the positive class; MED is "not DAX".
- You will receive the participant's full Announce line (may contain both DAX and MED).
- Your task: EXTRACT ONLY THE DAX RULE and decide if it has the SAME MEANING as the ground-truth DAX rule.

How to extract the DAX rule (in order):
1) If the text contains "DAX:", "DAX is", "A DAX triple is", or "The DAX rule is", take the phrase following that up to the next ";", ".", ",", or the start of a MED clause.
2) Otherwise, if there are two clauses separated by ";" or ", and", take the first clause as the DAX rule and ignore the rest.
3) Otherwise, treat the whole line as the DAX rule.

Guidance specific to the ground-truth rule:
{rule_guidance}

Rules:
- Ignore phrasing/synonyms if meaning is identical.
- If scopes differ (broader/narrower) or the family differs, answer NO.

Respond with ONLY one token: YES or NO.

ANNOUNCE TEXT:
"{announced_rule}"

GROUND-TRUTH DAX RULE:
"{ground_truth_rule}"

C.3.3 Announcement Judge Examples

Baseline / Think-in-Opposites. Ground-truth rule: All even

Rule guidance (excerpt):

> ACCEPT: "all even", "each number is even".
> REJECT: any rule that allows odds, says "at least one even", or uses divisibility instead of parity.

Model announcement: `All three numbers are even.`
Judge output: `YES`

Near-miss announcement: `At least one number is even.`
Judge output: `NO`

Dual-Goal (DAX extraction). Ground-truth DAX rule: Ascending

Model announcement: `Announce: DAX rule - strictly increasing;Announce: MED rule - not strictly increasing.`

Judge output: `YES`

C.4 Compatibility and Incompatibility Judge

Purpose. This judge classifies each tested triple as compatible or incompatible with the model's current hypothesis, enabling computation of the incompatible-to-compatible ratio (I:C).

Judge model. We use `Qwen-2.5-Coder-32B-Instruct` with decoding temperature 0.2 for stable rule-to-Python conversion.

Procedure. For each `Test` turn:

1. The model's current hypothesis is taken from the immediately preceding `Announce` turn.
2. The hypothesis is converted into an executable Python function using an LLM.
3. The tested triple (a, b, c) is evaluated using this function.
4. If the function returns `True`, the test is compatible; otherwise incompatible.

C.4.1 Compatibility Judge Prompt (Rule-to-Python)

Your task is to write a Python function that determines whether three integers (a, b, c) satisfy a given natural-language rule.

The function should directly implement the logical condition described in the rule.
Use basic arithmetic and comparisons. If the rule mentions "prime", "square", or "cube" numbers, you may define small helper functions (e.g., is_prime, is_cube) nested inside the main function.
Do not import any modules. Use pure Python only.

Handle negative and zero values safely — especially for divisibility (avoid divide-by-zero errors).
Return True if the triple satisfies the rule, otherwise False.

—
FORMAT & SAFETY CONSTRAINTS (MANDATORY):

- Output ONLY the function definition. No prose, no explanations, no code fences.
- Signature must be exactly:
def rule(a: int, b: int, c: int) -> bool:
- You may define small helper functions NESTED INSIDE rule().
- Do NOT use: import, from, global, nonlocal, with, try/except, raise, class, lambda, eval, exec, file or network I/O, or context managers.
- Use only pure Python integer arithmetic, comparisons, loops, and these builtins: abs, min, max, range, len, all, any, sum.
- You may only call functions that are:
* the parameters a, b, c,
* local variables or helper functions defined inside rule, or
* the allowed builtins listed above.

If a name is not in this list, you MUST NOT use it.
In particular, do NOT use: sorted, set, list, dict, pow, divmod, enumerate, zip, map, filter, or any other builtin.

- Handle negatives and zero safely; avoid division/modulo by zero.

- Before you output the function, silently check that you did not use any disallowed syntax and did not call any builtin other than the allowed ones above.
Then output ONLY the final function.

—

Rule:
{HYPOTHESIS}

Repair loop. If the generated function fails to execute (e.g., syntax errors or division-by-zero), the full error message is returned to the model and a corrected function is requested. This loop continues until a valid executable function is produced.

C.4.2 Compatibility Judge Examples

Natural-language hypothesis.

```
The gaps strictly increase:  (b-a) < (c-b) and both are
positive.
```

Generated function.

```
def rule(a: int, b: int, c: int) -> bool:
    return (b - a) > 0 and (c - b) > 0 and (b - a) < (c - b)
```

Compatibility evaluation.

$(1, 2, 4) \rightarrow$ `compatible`, $(1, 3, 5) \rightarrow$ `incompatible`.

C.5 Hypothesis Agreement Analysis

Our I:C metric treats the model's announced rule as its current hypothesis. To assess whether a single announcement reliably reflects the model's hypothesis, we perform a resampling analysis on 100 announcements from Qwen3-8B in thinking mode across interaction settings. For each announcement, we resample the model 10 times from the same conversation history and use Gemini-2.5-Pro to determine whether each resampled hypothesis agrees with the original announcement.

Across the 100 examples, the average agreement rate is 81.6%. Moreover, for 88 of the 100 examples, the original announcement agrees with the majority of the resampled hypotheses. These results suggest that the announced rule generally reflects the model's dominant hypothesis.

D Full Model Results

Tables 9 and 10 report the full results for all models. Column definitions are provided below.

Column definitions.

- **Task Success**: Fraction of episodes in which the model eventually guessed the hidden rule correctly.
- **First Guess**: Fraction of episodes in which the model's initial hypothesis was correct, prior to any testing.
- **# Turns till Succ.**: Average number of test turns performed before the first correct hypothesis, computed over solved episodes only.
- **I:C$_{\text{sol}}$**: Micro-averaged ratio of incompatible to compatible tests *before the first correct hypothesis*, computed over episodes that were eventually solved.
- **I:C$_{\text{uns}}$**: Micro-averaged ratio of incompatible to compatible tests over all test turns in episodes that were not solved.
- **I:C$_{\text{all}}$**: Micro-averaged ratio of incompatible to compatible tests aggregated over both solved (up to the first correct hypothesis) and unsolved episodes.
- **# Tkns per Turn (Avg$_{\text{sol}}$)**: Average number of tokens generated per model turn, computed over turns up to the first correct hypothesis in solved episodes.
- **# Tkns per Turn (Avg$_{\text{uns}}$)**: Average number of tokens generated per model turn, computed over all turns in unsolved episodes.
- **# Tkns per Turn (Avg$_{\text{all}}$)**: Average number of tokens generated per model turn, aggregated over both solved and unsolved episodes.

Table 9: Results across all rules and episodes for Thinking Models.

Model	Variant	Task	First	# Turns until	I:C_{sol}	I:C_{uns}	I:C_{all}	# Tkns per Turn		
		Suc. ↑	Guess	Suc. ↓	↑	↑	↑	Avg_{sol}	Avg_{uns}	Avg_{all}
Qwen3-8B	Baseline	0.212	0.087	10.875	0.101	0.150	0.147	958.2	1123.2	1088.2
	Dual-Goal	0.400	0.075	7.365	0.813	0.525	0.546	1763.2	2394.1	2141.7
	Think-in-Opposites	0.363	0.000	9.008	0.826	0.314	0.353	1547.9	1811.9	1716.1
Qwen3-14B	Baseline	0.338	0.125	7.393	0.130	0.192	0.186	991.5	1274.1	1178.6
	Dual-Goal	0.575	0.138	7.661	0.609	0.369	0.404	1029.1	1411.8	1191.7
	Think-in-Opposites	0.475	0.125	7.082	0.435	0.243	0.262	1198.5	1453.6	1332.4
Qwen3-32B	Baseline	0.413	0.150	10.833	0.235	0.131	0.141	842.9	1161.4	1029.8
	Dual-Goal	0.600	0.125	10.287	0.780	0.771	0.773	1191.8	1893.0	1472.3
	Think-in-Opposites	0.613	0.125	6.381	1.092	0.457	0.544	1397.9	1532.4	1450.0
DeepSeek-R1 -Llama-70B	Baseline	0.350	0.075	8.103	0.416	0.315	0.321	830.2	1026.3	957.6
	Dual-Goal	0.500	0.075	8.385	0.725	0.588	0.608	1150.9	1271.4	1211.2
	Think-in-Opposites	0.463	0.107	10.073	0.596	0.461	0.480	998.6	1136.7	1072.8
QwQ-32B	Baseline	0.500	0.113	11.752	0.657	0.313	0.362	1269.5	2245.5	1757.5
	Dual-Goal	0.575	0.100	10.755	1.065	0.753	0.814	1679.5	3085.8	2277.2
	Think-in-Opposites	0.588	0.088	10.282	0.913	0.269	0.371	1386.9	1575.3	1464.5
Gemini-2.5-Pro	Baseline	0.725	0.087	11.136	0.849	0.564	0.656	–	–	–
	Dual-Goal	0.775	0.113	10.098	0.819	0.734	0.768	–	–	–
	Think-in-Opposites	0.775	0.062	8.520	1.069	0.585	0.748	–	–	–

Table 10: Results across all rules and episodes for Non-thinking Models.

Model	Variant	Task	First	# Turns until	I:C_{sol}	I:C_{uns}	I:C_{all}	# Tkns per Turn		
		Suc. ↑	Guess	Suc. ↓	↑	↑	↑	Avg_{sol}	Avg_{uns}	Avg_{all}
Qwen3-8B	Baseline	0.062	0.013	5.000	0.174	0.354	0.353	8.5	8.3	8.4
	Dual-Goal	0.075	0.075	0.000	–	0.574	0.574	21.0	12.7	13.4
	Think-in-Opposites	0.062	0.000	15.250	0.171	0.298	0.295	7.3	8.5	8.4
Qwen3-14B	Baseline	0.225	0.175	3.000	0.600	0.724	0.723	3.8	8.7	7.6
	Dual-Goal	0.125	0.113	0.500	0.000	0.411	0.411	13.7	15.2	15.0
	Think-in-Opposites	0.287	0.163	4.120	2.211	0.878	0.896	4.1	8.1	7.0
Qwen3-32B	Baseline	0.325	0.225	4.933	0.544	0.494	0.495	5.0	7.3	6.6
	Dual-Goal	0.100	0.100	0.000	–	0.456	0.456	19.9	14.9	15.4
	Think-in-Opposites	0.362	0.200	3.760	0.441	0.641	0.631	5.4	6.5	6.1
LLaMA-3.3- 70B-Instruct	Baseline	0.212	0.050	9.370	0.382	0.368	0.369	14.2	14.0	14.1
	Dual-Goal	0.200	0.025	12.869	0.709	0.521	0.529	17.8	25.1	23.7
	Think-in-Opposites	0.375	0.062	10.026	0.699	0.478	0.499	12.7	15.9	14.7
GPT-4o	Baseline	0.113	0.113	0.000	–	0.246	0.246	7.5	10.8	10.4
	Dual-Goal	0.138	0.100	6.750	0.265	0.342	0.341	12.8	17.8	17.1
	Think-in-Opposites	0.163	0.100	2.500	0.097	0.167	0.166	9.1	11.0	10.7

D.1 Results by Rule Category

We further break down performance by rule category for thinking models. Table 11 reports Task Success and I:C for ordering, parity, sign, and number-property rules. In most cases, moving from Baseline to Dual-Goal or Think-in-Opposites (TiO) increases both Task Success and I:C, suggesting that the intervention effects are not concentrated in a single rule category.

E Distillation Training Details

This section provides implementation details for the context distillation experiments in Section 5.

Table 11: **Wason task performance by rule category for thinking models.** We report Task Success and I:C for each interaction setting across ordering, parity, sign, and number-property rules.

Model	Variant	Ordering		Parity		Sign		Number Properties	
		Task Suc.	I:C	Task Suc.	I:C	Task Suc.	I:C	Task Suc.	I:C
DeepSeek-R1-Llama-70B	Baseline	0.160	0.29	0.133	0.27	0.067	0.22	0.640	0.53
	Dual-Goal	0.360	0.67	0.333	0.63	0.600	0.39	0.520	0.63
	Think-in-Opposites	0.360	0.63	0.333	0.48	0.067	0.25	0.680	0.62
Gemini-2.5-Pro	Baseline	0.600	0.66	0.867	0.75	0.400	0.51	0.920	0.83
	Dual-Goal	0.600	0.79	0.533	0.88	0.933	0.52	0.880	0.84
	Think-in-Opposites	0.720	0.80	0.933	1.05	0.400	0.58	0.960	0.98
Qwen3-8B	Baseline	0.080	0.13	0.000	0.15	0.000	0.08	0.480	0.23
	Dual-Goal	0.320	0.49	0.267	0.74	0.533	0.83	0.320	0.44
	Think-in-Opposites	0.240	0.39	0.400	0.55	0.133	0.29	0.440	0.32
Qwen3-14B	Baseline	0.280	0.24	0.067	0.23	0.067	0.03	0.520	0.24
	Dual-Goal	0.400	0.30	0.267	0.27	0.733	0.27	0.640	0.71
	Think-in-Opposites	0.360	0.31	0.333	0.34	0.133	0.17	0.680	0.29
Qwen3-32B	Baseline	0.360	0.14	0.067	0.16	0.267	0.09	0.560	0.21
	Dual-Goal	0.480	0.62	0.400	0.60	0.800	0.85	0.560	1.04
	Think-in-Opposites	0.520	0.52	0.467	0.59	0.533	0.23	0.760	0.92
QwQ-32B	Baseline	0.400	0.39	0.400	0.25	0.067	0.14	0.760	0.93
	Dual-Goal	0.560	0.67	0.467	0.72	0.733	0.57	0.520	1.53
	Think-in-Opposites	0.560	0.40	0.533	0.31	0.267	0.19	0.760	0.67

E.1 Training Data

We train on **all `Test` turns** produced by the teacher, using *untrimmed* trajectories (including unsuccessful episodes and all turns after the first correct announcement). Each training example consists of the dialog history up to a `Test` turn (formatted with the *baseline* prompt) and the teacher's next-turn response (generated under the *intervention* prompt). We use the same train/validation/test rule splits as in Appendix A.

E.2 Learning-rate Sweeps and Selected Checkpoints

We fine-tune students with LoRA-based supervised fine-tuning.

Other fixed hyperparameters. For Qwen3-8B students, we use an effective batch size 64, epochs 5, LoRA rank r=16 (α=32), warmup ratio 0.05, and weight decay 0.0. For the Qwen3-32B student, we use an effective batch size of 64, epochs 5, LoRA rank r=32 (α=64), warmup ratio 0.05, and weight decay 0.0.

Checkpoint selection. Within each teacher–student setting, we select the checkpoint with lowest validation loss and use it for the evaluations reported in the main paper. Table 12 summarizes the learning-rate sweep ranges explored for each teacher–student pair and the selected checkpoint used in all reported evaluations.

Table 12: Learning-rate sweeps and selected checkpoints for distillation.

Student ← Teacher	**LR sweep range**	**Best LR**	**Best step (epoch)**
Qwen3-8B ← Qwen3-8B (Think-in-Opposites)	$[1\times10^{-5}, 5\times10^{-4}]$ (7 values)	7.5×10^{-5}	3400 (3.02)
Qwen3-8B ← Qwen3-32B (Baseline)	$[1\times10^{-4}, 2\times10^{-3}]$ (6 values)	2×10^{-4}	3200 (2.84)
Qwen3-8B ← Qwen3-32B (Think-in-Opposites)	$[1\times10^{-5}, 5\times10^{-4}]$ (6 values)	2×10^{-4}	4400 (3.91)
Qwen3-32B ← Qwen3-32B (Think-in-Opposites)	$[2\times10^{-6}, 5\times10^{-5}]$ (6 values)	5×10^{-5}	2200 (1.96)

Compute. All models are fine-tuned on H200 GPUs (140GB each). Qwen3-8B students use 1 GPU and Qwen3-32B students use 2 GPUs, with training runs taking 2 to 4 days depending on validation loss convergence.

F Blicket Task

F.1 Original Experiment vs. Our Adaptation

The original Blicket detector experiment [15] is a developmental psychology paradigm designed to study human's causal reasoning. In the setup, humans are presented with **four objects** and a machine that activates according to a hidden causal rule. **Two of the objects are designated as blickets**, meaning they causally activate the machine. The hidden rule is either a conjunctive(AND) or disjunctive(OR) combination of the two blickets. Children observe experimenter-provided evidence and, in some conditions, are allowed to intervene by placing objects on the machine themselves. The interaction is flexible and does not follow a fixed turn structure. Participants have to identify the objects that are blickets, though are not required to explicitly state the hidden rule.

Our adaptation diverges from this original setting in two principal ways.

Structured interaction. We impose a fixed 45-turn alternating `Guess`–`Test` protocol, mirroring our structured adaptation of Wason's 2–4–6 rule-discovery task. On each `Test` turn, the model proposes a subset of objects and receives deterministic ON/OFF feedback. On each `Guess` turn, the model must explicitly announce both (i) the set of objects that are blickets and (ii) the underlying hidden rule. This structured format differs from the flexible exploratory interaction in the original experiment and enables systematic measurement of confirmation bias via the incompatible-to-compatible (I:C) test ratio.

Expanded data design. Whereas the original paradigm uses a small object set (four objects in the canonical setup) with a limited causal structure, we expand the hypothesis space along multiple dimensions. Similar to how we broadened Wason's single hidden ascending rule into a diverse rule family, we increase complexity in the Blicket setting by:

- varying the number of objects ($N \in \{4, 8\}$),
- varying the number of blickets ($K \in \{2, 3\}$),
- introducing an additional XOR rule alongside AND and OR.

F.2 Dataset Statistics

Across all configurations, we generate 192 evaluation episodes. An *episode* is defined as a single environment instance with: (i) a fixed set of objects, (ii) a designated subset of blickets, (iii) a hidden logical rule (AND, OR, or XOR), and (iv) a 45-turn alternating Guess–Test interaction.

For each combination of object count (N), blicket count (K), and rule type, we generate 16 distinct episodes by varying:

- the assignment of which objects are designated as blickets, and
- the initial object placement shown at the start of the interaction.

This results in a balanced data design summarized in Table 13, yielding $2 \times 2 \times 3 \times 16 = 192$ evaluation episodes.

F.3 Task Instruction Prompts and Examples

Below we provide the task instruction prompts used to initialize each episode of the Blicket task. As in the Wason setup, these instructions are provided once at the beginning of an interaction, after which the model alternates between `Announce` and `Test` turns. Across episodes, the initial object configuration varies (i.e., which objects are placed on the device), and the device may initially be either ON or OFF.

Table 13: Blicket evaluation dataset.

Objects (N)	Blickets (K)	Rule	# Episodes
4	2	AND	16
4	2	OR	16
4	2	XOR	16
4	3	AND	16
4	3	OR	16
4	3	XOR	16
8	2	AND	16
8	2	OR	16
8	2	XOR	16
8	3	AND	16
8	3	OR	16
8	3	XOR	16
		Total	192

We evaluate two variants: a baseline prompt and a Think-in-Opposites (TiO) prompt. The TiO version augments the baseline instructions with an explicit strategy encouraging the model to test opposing features of its current hypothesis, adapted to the object-combination setting.

F.3.1 Baseline prompt

You are playing a game with several objects and a device.

Placing a set of objects on the device causes it to turn on. This depends on which objects are placed on the device, according to a hidden rule.

Your goal is to discover:
- which objects are relevant to turn the device on
- the hidden rule that determines when the device turns on.

At each turn, you will be instructed to do one of two things:
- Test: propose a set of objects to place on the device. You will then receive feedback on the device's state: ON or OFF.
- Announce: based on all previous tests, state which objects you believe are relevant to turn the device on and describe the hidden rule. You will not receive any feedback for announcements.

Each instruction you receive will specify what to do.

Format (must follow exactly):
- If the instruction is Turn - Announce, output exactly one line:
Announce: relevant=[object A, object B, object C]; rule=<one short description of the rule >
(there can be any number of relevant objects)
- If the instruction is Turn - Test, output exactly one line:
Test: [object A, object B, object C]
(you can test any number and combination of objects)

Important rules:
- Always refer to objects using the exact name "object N".
- Do not include any extra text.

Initial configuration (this is not the result of a test):

You are in a room. You see a device at the center of this room.
There are also 4 objects scattered around the room. You observe them:
object 0 is on the floor, object 1 is on the device, object 2 is on the floor, object 3 is on the floor.
The device is off. You wonder if there is a relationship between the objects and the device.

Turn - Announce

Example interaction (blickets: object 0 and object 1, hidden rule: AND *The following transcript continues directly from the final "Turn–Announce" in the initial prompt above.*

```
[model] Announce: relevant=[object 0, object 2, object 3];
        rule=at least two of these objects must be on the device

[environment] Turn - Test

[model] Test: [object 0, object 1, object 2]

[environment] ON. Turn - Announce

[model] Announce: relevant=[object 0, object 2, object 3];
        rule=at least two of these objects must be on the device

[environment] Turn - Test

[model] Test: [object 0, object 1, object 3]
```

```
[environment] ON. Turn - Announce

[model] Announce: relevant=[object 0, object 2, object 3];
        rule=at least two of these objects must be on the device
```

F.3.2 Think-in-Opposites prompt

You are playing a game with several objects and a device.

Placing a set of objects on the device causes it to turn on. This depends on which objects are placed on the device, according to a hidden rule.

Your goal is to discover:
- which objects are relevant to turn the device on
- the hidden rule that determines when the device turns on.

At each turn, you will be instructed to do one of two things:
- Test: propose a set of objects to place on the device. You will then receive feedback on the device's state: ON or OFF.
- Announce: based on all previous tests, state which objects you believe are relevant to turn the device on and describe the hidden rule. You will not receive any feedback for announcements.

Each instruction you receive will specify what to do.

Strategy: At every test step, follow this reasoning procedure:
1. Form a concise current hypothesis of the relevant objects and hidden rule.
2. Identify one feature implicit in that hypothesis.
3. Test the OPPOSITE of that feature.
4. If the hypothesis still holds, the feature is probably irrelevant; if it does not, that feature might be crucial.
5. In short, prefer tests that both confirm and contradict your current idea.

Format (must follow exactly):
- If the instruction is Turn - Announce, output exactly one line:
Announce: relevant=[object A, object B, object C]; rule=<one short description of the rule >
(there can be any number of relevant objects)
- If the instruction is Turn - Test, output exactly one line:
Test: [object A, object B, object C]
(you can test any number and combination of objects)

Important rules:
- Always refer to objects using the exact name "object N".
- Do not include any extra text.

Initial configuration (this is not the result of a test):

You are in a room. You see a device at the center of this room.
There are also 4 objects scattered around the room. You observe them:
object 0 is on the floor, object 1 is on the device, object 2 is on the floor, object 3 is on the floor.
The device is off. You wonder if there is a relationship between the objects and the device.

Turn - Announce

Example interaction (blickets: object 0 and object 1, hidden rule: AND *The following transcript continues directly from the final "Turn–Announce" in the initial prompt above.*

```
[model] Announce: relevant=[object 0, object 2, object 3];
```

```
        rule=at least two of these objects must be on the device

[environment] Turn - Test

[model] Test: [object 0, object 1]

[environment] ON. Turn - Announce

[model] Announce: relevant=[object 0, object 2, object 3];
        rule=at least one of these objects must be on the device

[environment] Turn - Test

[model] Test: [object 1, object 2, object 3]

[environment] OFF. Turn - Announce

[model] Announce: relevant=[object 0, object 1];
        rule=both of these objects must be on the device
```

F.4 Judge Prompts

F.4.1 Correctness Judge Prompt

Decide whether the agent's ANNOUNCEMENT is COMPLETELY CORRECT.

Ground truth:
- True relevant objects: {true_blickets}
- True rule type: {true_rule}

Rule semantics:
- If true_rule_type is 'disjunctive', the device turns on iff at least ONE of the relevant objects is on the device.
- If true_rule_type is 'conjunctive', the device turns on iff ALL of the relevant objects are on the device.
- If true_rule_type is 'xor', the device turns on iff EXACTLY ONE of the relevant objects is on the device (not zero, not two+).

Agent announcement (verbatim):
{announce_text}

Mark True ONLY IF BOTH conditions hold:

1) The set of guessed relevant objects matches the true relevant objects exactly (same members; order doesn't matter; no extras; no missing).
- The agent may say "relevant objects are ..." OR directly name the correct objects; both are acceptable.

2) The stated rule matches the true rule semantics exactly.
- If the agent states OR / ANY / AT LEAST ONE when the true rule is conjunctive, that is False.
- If the agent states AND / ALL / NEEDS BOTH when the true rule is disjunctive, that is False.
- If the agent states XOR / EXACTLY ONE / A OR B BUT NOT BOTH / ONE AND ONLY ONE when the true rule is xor, that is True.
- If the agent states relevant object A on and relevant object B off, but does not specify the other direction (A off, B on) when the true rule is xor, that is False.
- If the agent states odd number of relevant objects on when there are 3 or more relevant objects and the true rule is xor, that is False.

If there is any ambiguity, missing detail, partial correctness, or mismatch, output False.

Output exactly one word: True or False.

Example Ground-truth rule: AND; Blickets = object 0 and object 1

Model announcement: `relevant=[object 0, object 1]; rule=both the relevant objects must be on the device`
Judge output: `YES`

Near-miss announcement: `relevant=[object 0, object 1]; rule=any of the relevant objects must be on the device`
Judge output: `NO`

F.4.2 Compatibility Judge Prompt

Your task is to write a Python function that determines whether a given device would be ON, according to a natural-language hypothesis about relevant objects and a rule.

The input is a list named state of length num_objects.
- state[i] is True iff "object i" is on the device.
- state[i] is False iff "object i" is NOT on the device.

The function should implement the ON-condition described by the hypothesis.
Return True if the hypothesis predicts the device is ON for the given state, otherwise False.

You may assume objects are named "object 0" ... "object (num_objects-1)".
Empty sets are allowed.

—
FORMAT & SAFETY CONSTRAINTS (MANDATORY):

- Output ONLY the function definition. No prose, no explanations, no code fences.
- Signature must be exactly:
def rule(state: list) ->bool:
- You may define small helper functions NESTED INSIDE rule().
- Do NOT use: import, from, global, nonlocal, with, try/except, raise, class, lambda, eval, exec, file or network I/O, or context managers.
- Use only pure Python booleans, indexing into state, comparisons, if/else, loops.
- Allowed builtins: abs, min, max, range, len, all, any, sum, int, bool, isinstance.
- You may only access:
* the parameter state,
* local variables / helper functions defined inside rule,
* the allowed builtins listed above.
- Before you output the function, silently check you did not use disallowed syntax or names. Then output ONLY the final function.

—
Hypothesis:
{HYPOTHESIS}

num_objects={NUM_OBJECTS}

Example Natural-language hypothesis.

`relevant=[object 0, object 2, object 3]; rule=at least two of these objects must be on the device`

Generated function.

```
def rule(state: list) -> bool:
    count = 0
    if state[0]:
        count += 1
    if state[2]:
        count += 1
    if state[3]:
        count += 1
    return count >= 2
```

Compatibility evaluation.

$(object0, object1) \rightarrow$ `incompatible`, $(object0, object1, object2) \rightarrow$ `compatible`.

G Distillation Results

G.1 Wason Distillation Results

Table 14: Model performance after distillation. We mark statistically significant improvements over the base model with superscript *.

Student (+ Teacher)	Prompt	Task Suc.	# Turns until Suc.	I:C
Qwen3-8B	–	0.21	10.88	0.15
Qwen3-8B	TiO	0.36	7.37	0.35
Qwen3-32B	–	0.41	10.83	0.14
Qwen3-32B	TiO	0.61	6.38	**0.54**
Intervention Distillation				
Qwen3-8B + 8B TiO	–	0.35	11.92	0.31*
Qwen3-32B + 32B TiO	–	**0.64***	9.55	0.33*
Cross-scale Distillation				
Qwen3-8B + 32B	–	0.44*	14.44	0.22
Cross-scale & Intervention Distillation				
Qwen3-8B + 32B TiO	–	0.59*	11.41	0.38*

G.2 Blicket Transfer Results

Table 15: Performance of distilled models trained on the Wason task, evaluated on the Blicket test. Statistically significant improvements over the base model are marked with *.

Student (+ Teacher)	Prompt	Task Suc.	# Turns until Suc.	I:C
Qwen3-8B	Base	0.32	10.26	0.44
Qwen3-8B	TiO	0.40	10.73	0.57
Qwen3-32B	Base	0.57	12.18	0.54
Qwen3-32B	TiO	**0.86**	12.98	**1.31**
Models Distilled on Wason Task				
Qwen3-8B + 8B TiO	Base	0.34	13.96	0.48
Qwen3-8B + 32B	Base	0.36	10.25	0.53
Qwen3-8B + 32B TiO	Base	0.38	11.84	0.51
Qwen3-32B + 32B TiO	Base	0.77*	11.02	0.88*
Models Distilled on Wason Task + TiO Prompt				
Qwen3-8B + 8B TiO	TiO	0.35	–	0.64
Qwen3-32B + 32B TiO	TiO	0.85	–	1.35

G.3 Catastrophic Forgetting Check

Fine-tuning on a narrow task can degrade general capabilities through catastrophic forgetting [23]. To check whether distillation on the Wason task harms pre-existing mathematical reasoning ability, we evaluate all Wason-distilled models on the GSM8K test set [9] (1,319 examples) alongside their base models. Since our distillation targets exploration behavior in multi-turn setups, we do not expect it to influence single-turn mathematical reasoning. The results below in Table 16 confirm this with differences of at most 2% relative to base models. We see small degradation for the 8B model, but no degradation for the 32B model. Distilling the 32B model into the 8B model for the Wason task even resulted in performance gains.

Table 16: GSM8K accuracy for base models and their Wason-distilled counterparts. Across all distilled models, accuracy changes by at most 2% relative to base models.

Student (+ Teacher)	**GSM8K Acc.**
Qwen3-8B	0.919
Qwen3-32B	0.930
Models Distilled with Wason Task	
Qwen3-8B + 8B TiO	0.899
Qwen3-8B + 32B Base	0.937
Qwen3-8B + 32B TiO	0.933
Qwen3-32B + 32B TiO	0.932

H Mathematical Reasoning Experiment Details

This section provides additional experimental details for the mathematical reasoning evaluation in Section 7.2.

H.1 Evaluation Setup

We evaluate Qwen3-8B and Qwen3-32B and their corresponding same-scale intervention-distilled models (8B $\leftarrow$ 8B TiO and 32B $\leftarrow$ 32B TiO) on AIME 2025 and Beyond AIME. The distilled models are trained exclusively on the Wason task as described in Section 5.

For each problem, the model first produces a solution and final answer. If the final answer is incorrect, we retain the full conversation history, inform the model that its previous answer was incorrect, and ask it to review its previous reasoning and try again. We allow up to five retries after the initial attempt, resulting in a maximum of six attempts per problem.

Once a problem is answered correctly, it is considered solved for that attempt and all subsequent attempts. Accordingly, the accuracy reported in Figure 7 is cumulative: at each attempt, we report the fraction of problems that have been solved at least once by that point.

We use the same retry procedure for the base and distilled models. No Think-in-Opposites instruction or other intervention is provided at inference time.

H.2 Prompts

For the initial attempt, we use the following prompt:

> Solve the following problem. Give concise final working with only the key steps, then write your final answer after ####.
> Problem:
> {problem}

After an incorrect answer, the model receives the full conversation history followed by:

> Your answer was incorrect. Please try again. Review your previous working, fix the mistake, and keep the corrected working concise. Give concise final working with only the key steps, then write your final answer after ####.

The same retry prompt is provided after each incorrect attempt, up to a maximum of five retries.

I Statistical Significance Testing

I.1 Intervention Effects

To assess whether intervention prompts improve performance relative to the baseline prompt, we evaluate statistical significance using paired permutation tests.

For each metric (Task Success and I:C), we first compute the metric under the baseline prompt and under the intervention prompt (Dual-Goal or Think-in-Opposites). Metrics are aggregated across all episodes. We then compute the observed difference between intervention and baseline:

$$\Delta_{\text{obs}} = M_{\text{intervention}} - M_{\text{baseline}}.$$

To construct a null distribution, we randomly permute the baseline and intervention labels at the episode level while keeping the pairing between corresponding episodes. After each permutation, we recompute the difference between the intervention and baseline conditions. Repeating this process many times produces a distribution of differences expected under the null hypothesis that the intervention has no effect.

We perform $50{,}000$ permutations. Let k denote the number of permuted differences that are greater than or equal to the observed difference Δ_{obs}. The p-value is computed as

$$p = \frac{k+1}{N+1},$$

where N is the number of permutations. The $+1$ correction prevents zero p-values and yields an unbiased estimate of significance.

Tables 17 and 18 report the observed mean change (Δ) relative to the baseline prompt and the corresponding p-values. We further apply Benjamini–Hochberg (BH) correction to control the false discovery rate across the 8 Wason intervention comparisons (2 interventions $\times$ 2 metrics $\times$ 2 model types), and report the resulting q-values (BH-adjusted p-values).

Table 17: **Effect of interventions on the Wason Task.** Δ denotes the change relative to the baseline prompt. Statistical significance is assessed using paired permutation tests over episode pairs ($50{,}000$ permutations). q-values are computed using Benjamini–Hochberg correction over the 8 Wason intervention comparisons.

Thinking models					**Non-thinking models**				
Metric	Intervention	Mean Δ	p	q	Metric	Intervention	Mean Δ	p	q
Task Success	Dual-Goal	+0.148	$< 2 \times 10^{-5}$	$< 4 \times 10^{-5}$	Task Success	Dual-Goal	-0.060	0.0065	0.0087
	TiO	+0.123	$< 2 \times 10^{-5}$	$< 4 \times 10^{-5}$		TiO	+0.063	0.0033	0.0053
I:C	Dual-Goal	+0.375	$< 2 \times 10^{-5}$	$< 4 \times 10^{-5}$	I:C	Dual-Goal	+0.048	0.2386	0.2727
	TiO	+0.168	$< 2 \times 10^{-5}$	$< 4 \times 10^{-5}$		TiO	+0.020	0.5995	0.5995

Table 18: **Effect of Think-in-Opposites (TiO) on the Blicket test.** Δ denotes the change relative to the baseline prompt. Statistical significance is assessed using paired permutation tests over episode pairs ($50{,}000$ permutations). q-values are computed using Benjamini–Hochberg correction over the 4 Blicket comparisons.

	Non-thinking			**Thinking**		
Metric	Δ	p	q	Δ	p	q
Task Success	+0.003	0.936	0.936	+0.092	$< 2 \times 10^{-5}$	$< 8 \times 10^{-5}$
I:C	+0.064	0.033	0.044	+0.231	$< 2 \times 10^{-5}$	$< 8 \times 10^{-5}$

I.2 Distillation Effects

We additionally assess whether the performance gains from intervention distillation are statistically significant relative to the corresponding baseline models.

We focus on same-scale intervention distillation settings, where a student model is trained on data generated using Think-in-Opposites prompts from a teacher of the same scale (e.g., 8B $\rightarrow$ 8B, 32B $\rightarrow$ 32B). These settings isolate the effect of distilling intervention-induced reasoning behavior without confounding scale differences.

Statistical significance is evaluated using the same paired permutation test procedure described in Section I.1. For each episode, we pair the baseline model output with the corresponding distilled model output, compute the observed difference in Task Success and I:C, and construct a null distribution by randomly swapping labels within each episode. We perform $50{,}000$ permutations and compute two-sided p-values with the $(k+1)/(N+1)$ correction.

Table 19: **Effect of distillation on the Wason Task.** Δ denotes the change relative to the corresponding baseline model. Statistical significance is assessed using paired permutation tests over episode pairs ($50{,}000$ permutations).

Metric	Model	Mean Δ	p
Task Success	Qwen3-8B + 8B TiO	+0.138	0.053
	Qwen3-32B + 32B TiO	+0.225	0.00032
	Qwen3-8B + 32B	+0.225	0.00078
	Qwen3-8B + 32B TiO	+0.438	$< 2 \times 10^{-5}$
I:C	Qwen3-8B + 8B TiO	+0.163	0.0018
	Qwen3-32B + 32B TiO	+0.185	$< 2 \times 10^{-5}$
	Qwen3-8B + 32B	+0.077	0.184296
	Qwen3-8B + 32B TiO	+0.310	$< 2 \times 10^{-5}$

Table 20: **Effect of Wason Distillation on the Blicket Test.** Models are distilled on the Wason task and evaluated on the Blicket test. Δ denotes the change relative to the corresponding baseline model on the Blicket test. Statistical significance is assessed using paired permutation tests over episode pairs ($50{,}000$ permutations).

Metric	Model	Mean Δ	p
Task Success	Qwen3-8B + 8B TiO	+0.021	0.674
	Qwen3-32B + 32B TiO	+0.203	$< 2 \times 10^{-5}$
	Qwen3-8B + 32B Baseline	+0.042	0.347
	Qwen3-8B + 32B TiO	+0.063	0.150
I:C	Qwen3-8B + 8B TiO	+0.039	0.396
	Qwen3-32B + 32B TiO	+0.345	$< 2 \times 10^{-5}$
	Qwen3-8B + 32B Baseline	+0.092	0.051
	Qwen3-8B + 32B TiO	+0.070	0.090

J Judge Validation

Our evaluation relies on judge LLMs to determine (i) whether a model's announced rule is correct and (ii) whether a proposed test is compatible with a previously announced rule. To verify the reliability of these judges, we conduct a manual audit on a sample of evaluation instances from both tasks.

For the Wason and Blicket tasks, across all models evaluated on the test set, we randomly sample 200 announcement judgments each (100 labeled correct and 100 labeled incorrect by the judge) and 200 compatibility judgments each (100 labeled compatible and 100 labeled incompatible by the judge). One of the authors manually annotated each sample and this is compared with the decision of the LLM-as-a-judge.

Table 21 summarizes the audit results. Across all four audited settings, the judge decisions achieve $\geq$ 98.5% agreement with human annotations. Tables 22– 25 show a few representative examples.

K Evaluation Validity Checks

K.1 Re-prompting Frequency Check

When a model response does not follow the required output format, we re-prompt with the format requirement only, no task description is re-inserted. To verify that this procedure does not introduce

Table 21: Summary of the manual audit for judge validation.

Setting	Sample size	Pos./Neg.	Agreement
Wason announcement	200	100 / 100	198 / 200 (99.0%)
Wason compatibility	200	100 / 100	197 / 200 (98.5%)
Blicket announcement	200	100 / 100	199 / 200 (99.5%)
Blicket compatibility	200	100 / 100	197 / 200 (98.5%)

Table 22: Representative examples from the manual audit of Wason announcement judgments.

Correct announcement	Model announcement	Judge	Human
All end with 1	All numbers end with the digit 1.	CORRECT	CORRECT
Mixed signs	The triple contains at least one negative number and at least one non-negative number.	CORRECT	CORRECT
Descending	The numbers are in decreasing order.	CORRECT	CORRECT
All cube numbers	The numbers are in a geometric progression with a common ratio of -3.	INCORRECT	INCORRECT
All positive	The three numbers form an arithmetic sequence.	INCORRECT	INCORRECT
Contains a prime	The sum of the first two numbers is twice the third.	INCORRECT	INCORRECT

Table 23: Representative examples from the manual audit of Wason compatibility judgments.

Previous announcement	Test triple	Judge	Human
All numbers are perfect cubes	[8, 27, 64]	TRUE	TRUE
All three numbers are prime	[5, 17, 19]	TRUE	TRUE
The numbers are in strictly decreasing order	[9, 6, 3]	TRUE	TRUE
All numbers are prime	[2, 4, 35]	FALSE	FALSE
Exactly one even and two odd numbers	[4, 6, 8]	FALSE	FALSE
The numbers must form a strictly decreasing sequence	[1, 2, 3]	FALSE	FALSE

Table 24: Representative examples from the manual audit of Blicket announcement judgments.

True rule	True ids	Model announcement	Judge	Human
disjunctive	[0,1,2]	relevant=[object 0, object 1, object 2]; rule=contains at least one of object 0, 1, or 2	TRUE	TRUE
conjunctive	[0,3]	relevant=[object 0, object 3]; rule=objects 0 and 3 must be placed on the device together	TRUE	TRUE
xor	[0,7]	relevant=[object 0, object 7]; rule=the device turns on if exactly one of object 0 or object 7 is present	TRUE	TRUE
conjunctive	[0,3]	relevant=[object 0, object 1, object 2, object 3]; rule=the device turns on if all objects are placed on it	FALSE	FALSE
xor	[0,2,3]	relevant=[object 0, object 2, object 3]; rule=the device turns on if any one of them is present	FALSE	FALSE
disjunctive	[0,2]	relevant=[object 0, object 3]; rule=device turns on if object 0 or object 3 is present	FALSE	FALSE

systematic bias, we report re-prompting frequency on a 10% subset of the test set alongside the task success and I:C values reported in Table 2.

As shown in Table 26, non-thinking models require retries on fewer than 1% of turns. Thinking models require retries on 36-47% of turns, most commonly on test turns where the model produces a rule guess instead of a test triple after its reasoning trace. Importantly, we observe no strong correlation between % Retry and either task success or I:C, indicating that re-prompting frequency does not explain performance differences across model-variant pairs.

K.2 Controlled-Prior Check

A potential concern is that task success reflects alignment between a model's prior over rules and the problem set, rather than exploration quality per se. While we note that most evaluation benchmarks

Table 25: Representative examples from the manual audit of Blicket compatibility judgments.

Previous announcement	Tested object ids	Judge	Human
relevant=[object 0, object 2]; rule=must include both object 0 and object 2	[0, 2]	TRUE	TRUE
relevant=[object 1, object 7]; rule=exactly one of object 1 or object 7 is required to turn the device on, but not both	[5, 7]	TRUE	TRUE
relevant=[object 0]; rule=object 0 alone is sufficient to turn the device on	[0]	TRUE	TRUE
relevant=[object 0, object 1, object 3]; rule=The device turns on when at least one of object 0, object 1, or object 3 is placed on it.	[2]	FALSE	FALSE
relevant=[object 0, object 2]; rule=contains object 0 or object 2	[1, 3]	FALSE	FALSE
relevant=[object 0, object 1, object 2, object 3]; rule=all objects must be placed on the device	[0, 1, 3]	FALSE	FALSE

Table 26: Re-prompting frequency alongside task success and I:C for a 10% subset of the test set. High retry rates in thinking models reflect format errors after reasoning traces, not task difficulty. No strong correlation is observed between % Retry and either performance metric.

Model	Variant	Task Suc.	I:C	% Retry
Thinking Models				
Qwen3-8B	Baseline	0.21	0.15	41.07
	TiO	0.36	0.35	45.19
	Dual-Goal	0.40	0.55	46.98
Qwen3-14B	Baseline	0.34	0.19	47.12
	TiO	0.48	0.26	46.15
	Dual-Goal	0.58	0.40	46.15
Qwen3-32B	Baseline	0.41	0.14	45.05
	TiO	0.61	0.54	44.37
	Dual-Goal	0.60	0.77	46.98
DeepSeek-R1-LLaMA-70B	Baseline	0.35	0.32	37.77
	TiO	0.46	0.48	39.29
	Dual-Goal	0.50	0.61	36.40
QwQ-32B	Baseline	0.50	0.36	36.95
	TiO	0.59	0.37	40.80
	Dual-Goal	0.58	0.81	44.51
Non-Thinking Models				
Qwen3-8B	Baseline	0.06	0.35	0.00
	TiO	0.06	0.30	0.00
	Dual-Goal	0.08	0.57	0.69
Qwen3-14B	Baseline	0.23	0.72	0.00
	TiO	0.29	0.90	0.27
	Dual-Goal	0.13	0.41	0.41
Qwen3-32B	Baseline	0.33	0.50	0.14
	TiO	0.36	0.63	0.00
	Dual-Goal	0.10	0.46	0.00
LLaMA-3.3-70B	Baseline	0.21	0.37	0.00
	TiO	0.38	0.50	0.41
	Dual-Goal	0.20	0.53	0.63

are similarly confounded by factors beyond their primary target (e.g., answer formatting in math benchmarks, position bias in multiple-choice settings), we nonetheless design an ablation to control for prior effects directly.

Setup. In addition to an initial triple, we provide models a candidate set of compatible rules and instruct them that each candidate is equally likely, thereby fixing the prior over rules to be uniform. We run 96 episodes (8 initial triples $\times$ 12 compatible rules) using three thinking-mode base models (Qwen3-8B, Qwen3-14B, Qwen3-32B) under Baseline and Think-in-Opposites variants. To reflect

Table 27: Controlled-prior ablation results. Even when rule candidates are provided with uniform probability, I:C positively correlates with task success (Spearman $\rho = 0.341$, $p = 0.021$), and Think-in-Opposites increases disconfirmatory testing.

Model	Variant	Task Suc.	I:C
Qwen3-8B	Baseline	68.75	0.551
	Think-in-Opposites	78.13	0.918
Qwen3-14B	Baseline	79.17	0.871
	Think-in-Opposites	79.17	1.906
Qwen3-32B	Baseline	77.08	1.000
	Think-in-Opposites	79.17	2.047

the reduced difficulty of the setting, we limit each episode to 9 turns (5 guess, 4 test). Results are averaged over all 8 triples and 12 rules per model-variant pair.

I:C correlates with task success under controlled prior. Across model-variant-triple combinations (each point averaged over 12 rules), we observe a statistically significant positive correlation between I:C and task success (Spearman $\rho = 0.341$, $p = 0.021$). This holds even when the prior is constant, supporting the interpretation that I:C captures genuine exploration quality rather than prior alignment.

Effect of intervention under controlled prior. The effects of Think-in-Opposites intervention is less pronounced here compared to the main results (Tables 2). We think this is due to the simplicity of the new task setting, leaving less room for exploration strategies to differentiate.

K.3 Hypothesis Agreement Check

Our I:C metric treats the model's announced rule as its current hypothesis. To test whether a single announcement reliably reflects the model's hypothesis, we sample 100 announcements from Qwen3-8B in thinking mode across interaction settings. For each announcement, we resample the model 10 times from the same conversation history and use Gemini-2.5-Pro to determine whether each resampled hypothesis agrees with the original announcement.

Across the 100 examples, the average agreement rate is 81.6%. Moreover, for 88 of the 100 examples, the original announcement agrees with the majority of the resampled hypotheses. These results suggest that the announced rule generally reflects the model's dominant hypothesis.

L Broader Impact

This paper focuses on identifying and mitigating confirmation bias in LLMs. Confirmation bias may affect LLM decision-making in numerous tasks, ranging from hiring to code debugging to searching for scientific hypotheses. We urge researchers and developers who use our (or other) confirmation bias mitigation strategies to carefully test the impact across the full range of domains where their models may be used. There may be additional potential societal impacts of our work, but none that we feel must be specifically highlighted here.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction accurately state the paper's contributions: (1) a framework and dataset for measuring confirmation bias in LLMs via the Wason 2-4-6 rule-discovery task, (2) evidence that LLMs exhibit confirmation bias that negatively correlates with task success, (3) that human-derived prompting interventions (Dual-Goal, Think-in-Opposites) reduce confirmation bias and improve performance, and (4) that distillation can internalize these behaviors, with generalization to the Blicket test and beyond rule discovery to mathematical self-correction. The claims are supported by the experimental sections on Wason-style rule discovery, intervention results, distillation, Blicket generalization, and mathematical reasoning.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We discuss limitations in the Limitations and Future Work paragraph of the Conclusion. It notes that the framework primarily studies confirmation bias in controlled rule-discovery settings and, while the benefits of distillation extend beyond rule discovery to mathematical self-correction, future work is needed to evaluate whether similar improvements generalize to other real-world settings.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.

- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [N/A]

   Justification: The paper does not include theoretical results or formal proofs. All contributions are empirical.

   Guidelines:

   - The answer [N/A] means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: The paper provides detailed descriptions of the dataset construction process (Appendix A), all prompts used for the Baseline, Think-in-Opposites, and Dual-Goal conditions (Appendices B.1–B.2), LLM-as-a-judge prompts (Appendix C), distillation hyperparameters and learning-rate sweeps (Appendix E), data splits, and statistical testing procedures (Appendix I.1).

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.

(b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
(c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: The repository linked on page 1 contains the datasets, evaluation and distillation code, configuration files, and full model outputs, along with instructions for reproducing the main experimental results.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: Data splits (4 train / 2 validation / 4 test rule groups), evaluation metrics, and the interaction protocol are described in Section 3 and Appendix A. Appendix E provides full distillation hyperparameters (LoRA rank, learning rate sweeps, batch size, epochs, warmup ratio, and checkpoint selection).

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: The paper reports statistical significance for all main comparisons using paired permutation tests (50,000 permutations) with Benjamini–Hochberg correction for multiple comparisons, detailed in Appendices I.1 and I.2. Figures 2 and 5 display 95% confidence intervals.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.
   - It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
   - For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
   - If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Compute details are provided in Appendix E. Fine-tuning was conducted on H200 GPUs (140GB each): 1 GPU for Qwen3-8B students and 2 GPUs for Qwen3-32B students, with training runs taking 2 to 4 days depending on validation loss convergence.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The work evaluates and mitigates confirmation bias in LLM reasoning using synthetic tasks and existing model APIs/checkpoints. To the best of the authors' knowledge, the research conforms to the NeurIPS Code of Ethics.

Guidelines:

- The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: We discuss broader impact in Appendix L. This discusses how our strategies can reduce confirmation bias and affect LLM decision-making in numerous tasks, ranging from hiring to code debugging to searching for scientific hypotheses.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: The paper does not release a high-risk pretrained language model, scraped dataset, or other asset with clear misuse risk. The datasets are controlled synthetic rule-discovery tasks rather than sensitive real-world data.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.

- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: All datasets used are cited: the Large Dataset of Generalization Patterns in the Number Game [4] and Tenenbaum's Number Game [36] are credited for human-derived rules. The Blicket test environment is credited to GX-Chen et al. [16]. AIME 2025 [30] and Beyond AIME [6] are cited to their respective dataset sources. All LLMs are cited with their original papers.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: The new Wason rule-discovery dataset and Blicket evaluation dataset are fully documented in Appendices A and F.2, including generation procedures, filtering criteria, data splits, and all rule definitions and test instances (Tables 4–8). The full datasets and accompanying code are publicly available in the repository linked on page 1.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

Justification: The paper does not involve crowdsourcing or human subjects. All experiments are conducted with LLMs.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [N/A]

Justification: The paper does not involve human subjects; IRB approval is not applicable.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [Yes]

Justification: LLMs are a central component of the methodology. They are used as (1) agents being evaluated for confirmation bias, (2) LLM-as-a-judge for assessing rule correctness and triple compatibility (Appendix C), (3) a data generation tool for synthetic rule sets (Appendix A.1), and (4) teacher models for knowledge distillation (Section 5). All LLM roles are described in detail in the paper.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.