FAILING TO FALSIFY: EVALUATING AND MITIGATING CONFIRMATION BIAS IN LANGUAGE MODELS

Ayush Rajesh Jhaveri, Anthony GX-Chen, Ilia Sucholutsky, Eunsol Choi
New York University
{aj4332,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 slower and less frequent discovery of the hidden rule. We further explore intervention strategies (e.g., encouraging the agent to consider counter examples) developed for humans. We find prompting LLMs with such instruction 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 promising generalization to a new task, the Blicket test. 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 (Nickerson, 1998). For example, a hiring manager who believes that a particular demographic tends to perform better may keep hiring from that group and interpreting positive outcomes as confirmation. Such selective exploration, known as **confirmation bias**, leads to inefficient learning and decision-making. Figure 1 exemplifies such a scenario in a rule-discovery task (Wason, 1960), where consistently using confirmatory examples leads to failed rule discovery.

We aim to understand confirmation bias for large language models (LLMs) acting as agents (Yao et al., 2022; Wang et al., 2024; Shinn et al., 2023). 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 (Wason, 1960). We 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 generate evidence and revise hypotheses. 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 45 interactions). We find recent LLMs with longer reasoning traces (Qwen Team, 2024; Yang et al., 2025) 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.

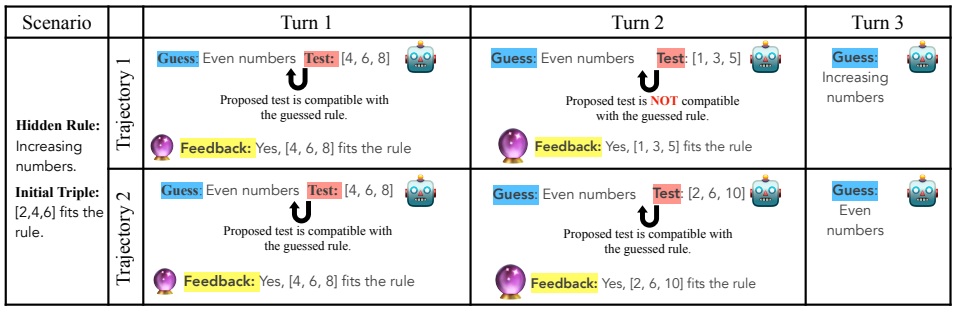


Figure 1: *Confirmation bias leads to narrow exploration.* We show two trajectories for rule discovery task, where an agent aims to infer a hidden numerical rule over multiple turns. Starting from an initial triple, the agent guess a hypothesis and test a new triple, receiving binary feedback on whether the proposed triple satisfies the hidden rule. A **compatible test** is consistent with the agent's current hypothesis, whereas an **incompatible test** contradicts it. Trajectory 2 proposes compatible tests in both turns, showing confirmation bias. In contrast, Trajectory 1 introduces incompatible test in the second turn, allowing elimination of incorrect hypothesis and lead to correct hidden rule discovery.

We then study methods to decrease confirmation bias, using two well-established interventions originally developed for humans: Dual-Goal (Gale & Ball, 2006) and Think-in-Opposites (Branchini et al., 2023). Prompting LLMs with these strategies leads to meaningful performance gains. We then investigate whether fine-tuning LLMs via symbolic knowledge distillation (West et al., 2022) to directly use these intervention strategies helps decrease confirmation bias, alleviating the need for task-specific inference-time interventions. Finally, We study if the fine-tuned models can generalize to a new domain: the Blicket Test (Gopnik & Sobel, 2000). We again confirm that a naive model can exhibit confirmation bias, which also correlates with low performance in this new task. However, a model fine-tuned on the *original task* exhibits *generalization* of its behavior, leading to lower confirmation bias and improved task success without any additional fine-tuning or prompting.

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, our work lays the groundwork for studying how reductions in confirmation bias generalize beyond rule-discovery settings to broader reasoning tasks.

2 Background

2.1 Wason's *2–4–6* Study

Setting Human participants were shown the triple of integers (i.e., `2-4-6`), told it conformed to a hidden rule, 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 session 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. 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.

LLM study Recent relevant work (Banatt et al., 2024) adapted this study 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. They find that the LLMs they evaluated did not perform competitively, at best achieving 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.

2.2 Intervention to Decrease Confirmation Bias

We describe two debiasing interventions from prior human studies that are designed to reduce confirmation bias in Wason's 2-4-6 task.

Think-in-Opposites (Branchini et al., 2023) introduces a metacognitive strategy to reduce confirmation bias. Specifically, participants are instructed to identify a salient feature of the example and then deliberately test an instance that is "opposite" with respect to that feature.

Dual-Goal (Gale & Ball, 2006) rephrases the task to guess two rules: the DAX rule (original rule) and the MED rule, which is the complement of the DAX rule. This encourages humans to test both confirmatory (DAX) and disconfirmatory (MED) triples, reducing confirmation bias.

3 Our Framework to Identify Confirmation Bias in LLM

Notation Summary

- Episode	e	A full sequence of rule discovery game: input followed by 45 guess–test–feedbacks turns
- Input	$(\mathbf{x}_{\text{ini}}, \mathbf{r}^*)$	(Initial triple of integers, Hidden rule)
- 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.[1] Then, LLM takes `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 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.

[1]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	$(79,\ 43,\ 31)$, $(89,\ 67,\ 2)$

(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 (1960), 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 $\mathbf{r}^*$, as well as 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 to generate a rule set that is diverse, while also representing human rules. 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). 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 a set 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 (Bigelow & Piantadosi, 2018; Tenenbaum, 1999). 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 (Wason, 1960) as is. The prompt and example interaction in Appendix B.1. Each episode begins with an initial

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.

	Non-Thinking				Thinking			
	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

`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. (2023) 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 (Gale & Ball, 2006) (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 H.

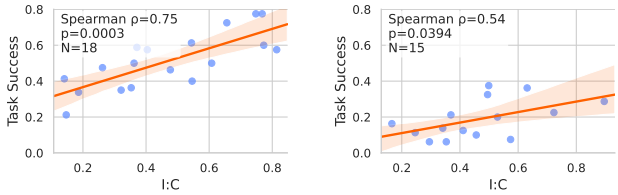
LLMs (standard inference) Qwen3-8B, Qwen3-14B, and Qwen3-32B (Yang et al., 2025); Llama-3.3-70B-Instruct (Meta AI, 2024); and GPT-4o (OpenAI, 2025).

LLMs with Thinking Mode Gemini-2.5-Pro (Comanici et al., 2025); Qwen3-8B, Qwen3-14B, and Qwen3-32B (Yang et al., 2025); QwQ-32B (Qwen Team, 2024); and DeepSeek-R1-Distill-Llama-70B (DeepSeek-AI, 2025). Thinking-mode models generate reasoning traces before an answer.

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. Additional diagnostics, such as first attempt guess rate, are in Appendix D.

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. across all three models and interaction settings. The number of turns to success is not always better with thinking models; potentially, they explore more to identify harder rules, which non-thinking models fail to guess. We also observe that thinking models generate more number of tokens per turn. We also find that the performance improvement from switching non-thinking to thinking mode is not accompanied by a consistent increase in I:C, indicating that these gains are not driven by reduced confirmation bias.



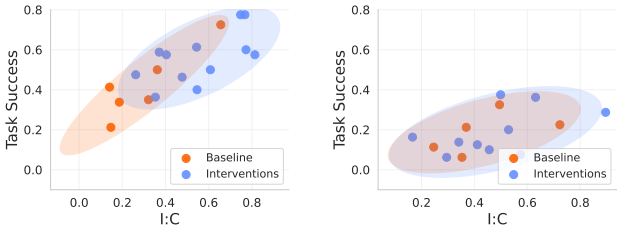
(a) Thinking models (b) Non-thinking models

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.

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.

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). The statistical testing results are in Appendix G.1.



(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.*

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 behavior without providing it at inference time as a prompt. We use a symbolic knowledge distillation approach (West et al., 2022). We generate input-output pairs from the experimental setting with lower confirmation bias and higher task success rate, and 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.

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:

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

We retain the teacher output, including reasoning tokens, since the intervention changes the model's reasoning and hypothesis-testing process. We construct the distillation dataset using the train/validation/test rule. 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 Table 3 reports the performance. 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 also 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. We next investigate whether these distilled models generalize to a new task.

Table 3: Model performance after intervention distillation. We mark distillation performance gap that are statistically significant (from its base model) with superscript * for task success and I:C ratios.

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*

6 Generalization to New Domain: Blicket Test

We further evaluate if models trained to reduce confirmation bias on the Wason task (through TiO distillation) can generalize to a *unseen* task that require hypothesis testing. To this end, we use *the Blicket Test* environment from GX-Chen et al. (2025). Designed by developmental psychologists to test children's ability for causal reasoning (Gopnik & Sobel, 2000), this setting has recently been adapted into text-based games to evaluate LLMs (Piriyakulkij et al., 2024; GX-Chen et al., 2025).

The Blicket Test involves N objects and a "Blicket detector" machine. In the task, a subset of objects (the "Blickets") activates the machine according to an unobserved rule. The *disjunctive* rule describes an OR relationship, whereby the machine turns on when any Blicket object is placed on it. The *conjunctive* rule describes an AND relationship, whereby the machine turns on only when *all* Blicket objects are placed on it. This tests an agent's ability to perform exploration and causal discovery.

Task We adapt the Blicket test to our interactive hypothesis-testing framework (see Appendix F.1 for comparison with the original experiment). Each **episode** begins with an initial object placement and the device being ON or OFF, followed by 45 alternating `Guess` and `Test` turns, mirroring our

Wason setup. On each `Test` turn, the model proposes a subset of objects to place on the device and receives feedback if the device is "ON" or "OFF". On each `Guess` turn, it outputs the guessed set of blickets and the hidden rule; no feedback is provided. A correct guess must identify both the full set of blickets and the correct rule. We evaluate two variants of the initial task prompt: 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.

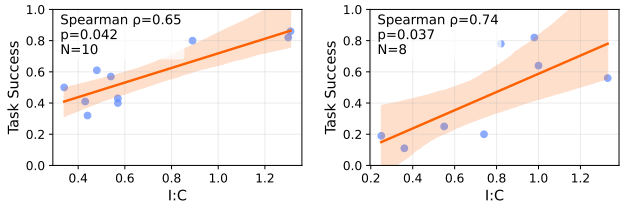
Data We vary the task along three dimensions: (1) number of objects (4 or 8), (2) number of blickets (2 or 3), and (3) hidden rule (AND, OR, or XOR). For each configuration, we generate 16 episodes by varying the blicket assignments and initial object placements. This yields $2 \times 2 \times 3 \times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

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

Table 4: **LLM Task performance in the Blicket Task.** Each metric is aggregated across 192 evaluation episodes.

	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

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 4 reports overall performance across models, and Figure 4 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.



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

Figure 4: **Confirmation bias correlates with task success.** Higher I:C is associated with higher task success. Shaded regions show 95% confidence intervals. *Each point represents a (model, variant) averaged over 192 episodes.*

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 the incompatible-to-compatible test ratio (I:C) on average, and these improvements are statistically significant under paired permutation tests (Appendix G.1). For non-thinking models, the effects are substantially smaller.

Does decreasing confirmation bias in one domain decrease confirmation bias in another domain? In Section 5, we fine-tuned LLMs to show better exploration of the hypothesis space by distilling the intervention prompt. Would the distilled model in the number-based game domain generalize to the new object-based domain?

Table 5 shows Blicket performance for models distilled on the Wason task (bottom), with base model results(top) from Table 4 for comparison. Same-scale intervention distillation (8B TiO → 8B student; 32B TiO → 32B student) yields improvements over the corresponding base models, with statistical test results provided in Appendix G.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.

Table 5: Performance of **distilled models** trained on the Wason task are evaluated on the Blicket Test baseline. We mark distillation performance gap that are statistically significant (from its base model) with superscript * for task success and I:C ratios.

Model	Prompt	Task Suc. (until Suc.)	# Turns	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 with 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*

7 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 (Dasgupta et al., 2022), threshold priming (Chen et al., 2024), overthinking (Liu et al., 2025b), overestimating rationality (Liu et al., 2025a), forming implicit associations (Bai et al., 2025), linguistic illusions (Marjieh et al., 2024), skewed representations of numbers (Marjieh et al., 2025), and causal reasoning (GX-Chen et al., 2025).

Confirmation bias is one of the most well-established cognitive biases in humans, from seminal Wason's 2-4-6 rule-discovery task (Wason, 1960). In the subsequent decades, cognitive scientists have studied identifying and mitigating confirmation bias in humans (Gale & Ball, 2006; Branchini et al., 2023). Recent work has explored confirmation bias in LLMs, either as part of broader bias evaluations (Malberg et al., 2025) or as a standalone phenomenon (Wan et al., 2025). 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 (Romera-Paredes et al., 2024; Yamada et al., 2025; Novikov et al., 2025; Castro et al., 2025; Aygün et al., 2025).

8 Conclusion

We adapt methods from cognitive psychology to evaluate confirmation bias in LLMs, which emerges during interactive hypothesis exploration. Using a controlled rule-discovery framework and a process-level incompatible-to-compatible (I:C) metric, we show that LLMs favor confirmatory tests and that lower confirmation bias strongly correlates with higher task success. Psychologically grounded interventions—Dual-Goal and Think-in-Opposites—consistently reduce confirmation bias and improve rule discovery, particularly for thinking-mode models. We further show that this mitigation can be internalized via distillation, injecting falsification-oriented exploration behavior into base without intervention prompts at inference time. These gains extend beyond scale alone and generalize to a new object-based rule-discovery domain, where distilled models on the number domain show improved task performance and reduced confirmation bias in the new object-based domain. Together, this work provides a cognitively grounded and controlled framework for analyzing confirmation bias during hypothesis exploration in LLMs and demonstrates both prompt-level and parameter-level mechanisms for mitigating it. Future work can evaluate whether such debiasing behavior generalizes to more naturalistic reasoning tasks beyond structured rule-discovery settings.

SOFTWARE AND DATA

Code and data are available at https://github.com/ayushjhaveri/llm-confirmation-bias, including datasets, evaluation setup, distillation training, configuration files, and full model outputs for reproducibility.

ACKNOWLEDGEMENTS

We would like to 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.

REFERENCES

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.

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.

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.

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.

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.

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*, pp. 2025–02, 2025.

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, pp. 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.

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.

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.

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.

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.

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

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.

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*, 2025a. URL https://openreview.net/forum?id=dAeET8gxqg.

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*, 2025b. URL https://openreview.net/forum?id=J3gzdbYZxS.

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*, pp. 578–613, 2025.

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.

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.

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.

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.

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.

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

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.

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

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.

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.

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

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

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.

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

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`.

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.

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.

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.

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 (Bigelow & Piantadosi, 2018), which builds on Tenenbaum's Number Game paradigm (Tenenbaum, 1999).

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 9.

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 9.

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 6–10.

Table 6: 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 7: 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 8: 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 9: 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 10: 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.

Table 11: Results across all rules and episodes for Thinking Models.

Model	Variant	Task Suc. ↑	First Guess	# Turns until Suc. ↓	I:C_{sol} ↑	I:C_{uns} ↑	I:C_{all} ↑	# Tkns per Turn 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	–	–	–

```
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`.

D Full Model Results

Tables 11 and 12 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_{sol}**: Micro-averaged ratio of incompatible to compatible tests *before the first correct hypothesis*, computed over episodes that were eventually solved.
- **I:C_{uns}**: Micro-averaged ratio of incompatible to compatible tests over all test turns in episodes that were not solved.
- **I:C_{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_{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_{uns})**: Average number of tokens generated per model turn, computed over all turns in unsolved episodes.
- **# Tkns per Turn (Avg_{all})**: Average number of tokens generated per model turn, aggregated over both solved and unsolved episodes.

Table 12: Results across all rules and episodes for Non-thinking Models.

Model	Variant	Task Suc. ↑	First Guess	# Turns until Suc. ↓	I:C_{sol} ↑	I:C_{uns} ↑	I:C_{all} ↑	# Tkns per Turn 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

E Distillation Training Details

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

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 13 summarizes the learning-rate sweep ranges explored for each teacher–student pair and the selected checkpoint used in all reported evaluations.

Table 13: 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)

F BLICKET TASK

F.1 ORIGINAL EXPERIMENT VS. OUR ADAPTATION

The original Blicket detector experiment (Gopnik & Sobel, 2000) 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 14, 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.

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.

Table 14: 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

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 STATISTICAL SIGNIFICANCE TESTING

G.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 15 and 16 report the observed mean change (Δ) relative to the baseline prompt and the corresponding p-values.

Table 15: **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).

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

Table 16: **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).

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

G.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 G.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 17: **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 18: **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

H 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

Table 19: Summary of the manual audit for judge validation.

Setting	Sample size	Pos./Neg.	Agreement
Wason announcement	50	25 / 25	50 / 50 (100%)
Wason compatibility	50	25 / 25	50 / 50 (100%)
Blicket announcement	50	25 / 25	50 / 50 (100%)
Blicket compatibility	50	25 / 25	50 / 50 (100%)

Table 20: 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

of these judges, we conduct a manual audit on a stratified sample of evaluation instances from both tasks.

For the Wason and Blicket tasks, we sample 50 announcement judgments each (25 correct and 25 incorrect) and 50 compatibility judgments each (25 compatible and 25 incompatible). Each sample is manually labeled and compared with the decision of the judge.

Table 19 summarizes the audit results. Across all four audited settings, the judge decisions exactly match the human labels on the sampled instances. Tables 20– 23 show a few representative examples.

Table 21: 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 22: 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

Table 23: 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