

# Evaluating the Inductive Abilities of Large Language Models: Why Chain-of-Thought Reasoning Sometimes Hurts More Than Helps

<https://arxiv.org/pdf/2505.24225>  
<https://llm-inductive-abilities.vercel.app/>

---

*Haibo Jin<sup>1</sup>, Peiyan Zhang<sup>2</sup>, Man Luo<sup>3</sup>, Haohan Wang<sup>1</sup>*

*<sup>1</sup>University of Illinois at Urbana-Champaign, <sup>2</sup>Hong Kong University of Science and Technology, <sup>3</sup>Intel Labs*

*Email: [haibo@illinois.edu](mailto:haibo@illinois.edu)*



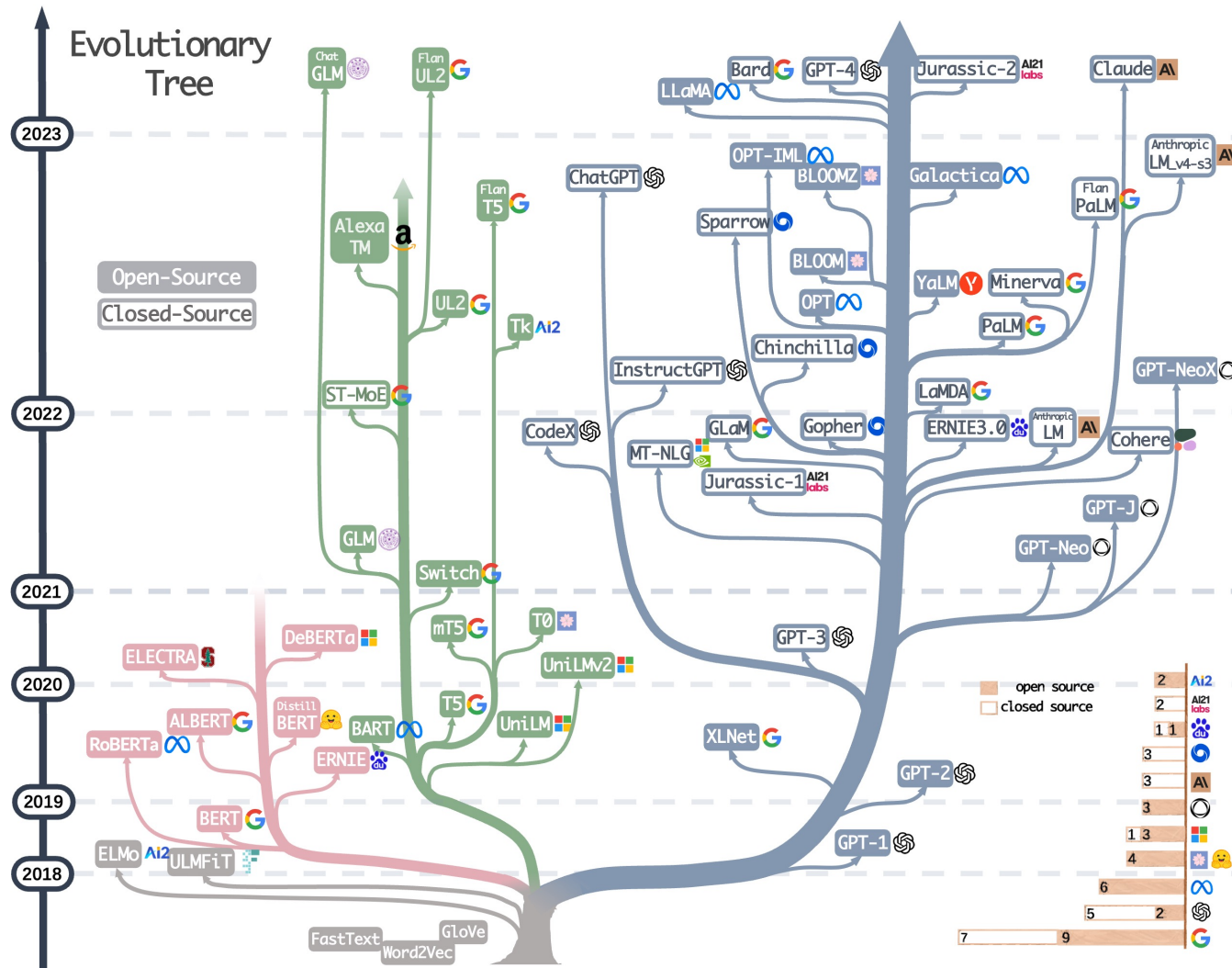
# Outline

- **Background**
- **Motivation**
- **Inductive Reasoning Evaluation**
- **Root Causes of Reasoning Failures**
- **Improving CoT Reasoning**
- **Conclusion**





# Background



**List 5 important papers authored by Geoffrey Hinton**

Geoffrey Hinton is a renowned computer scientist ... Here are five important papers authored by him:

1. "Learning Internal Representations by Error Propagation" (with D. E. Rumelhart and R. J. Williams) - This paper, published in 1986, ... ✓
2. "Deep Boltzmann Machines" (with R. Salakhutdinov) - Published in 2009, ... ✓
3. "Deep Learning" (with Y. Bengio and A. Courville) - Published as a book in 2016, ... ✗
4. "Attention Is All You Need" (with V. Vaswani, N. Shazeer, et al.) - Published in 2017, this paper introduced the Transformer model, ... ✗

**Generating text with citations**

Question: When did the US break away from England?

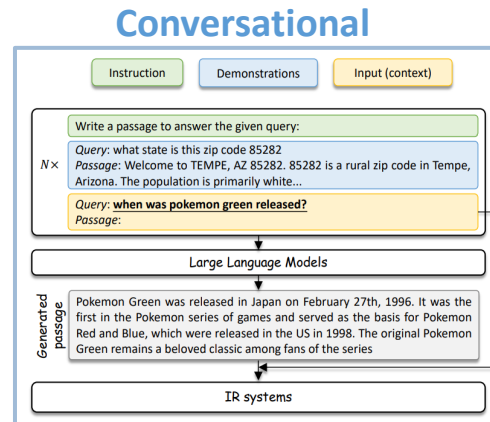
Model output: The US took the first step towards gaining independence from GB when it declared independence on July 2, 1776 (although the event is now commemorated on July 4, 1776, the date when the Declaration of Independence was officially adopted by Congress) [1][2]. The Treaty of Paris was later signed on September 3, 1783, formally separating the United States from the British Empire [3].

Retrieve LLM Corpus

[1] Decolonization of the Americas: ... independence from GB on July 2, 1776 ...

[2] Decolonization: ... It was officially adopted by Congress on July 4, 1776 ...

[3] American Revolution: ... The Treaty of Paris was signed September 3, 1783 ...



**Text Generation**

Extraction-based Question Answering (QA) (E.g. SQuAD)

**Document:**  $D = \{d_1, d_2, \dots, d_N\}$

**Query:**  $Q = \{q_1, q_2, \dots, q_N\}$

QA Model

output: two integers ( $s, e$ )

**Answer:**  $A = \{q_s, \dots, q_e\}$

In meteorology, precipitation is any product of the condensation of 17 spheric water vapor that falls under gravity. The main forms of precipitation include drizzle, rain, sleet, snow, graupel and hail... Precipitation forms as smaller droplets coalesce via collision with other rain drops or ice crystals within a cloud. Short, intense periods of rain 77 at 79 cations are called "showers".

What causes precipitation to fall?

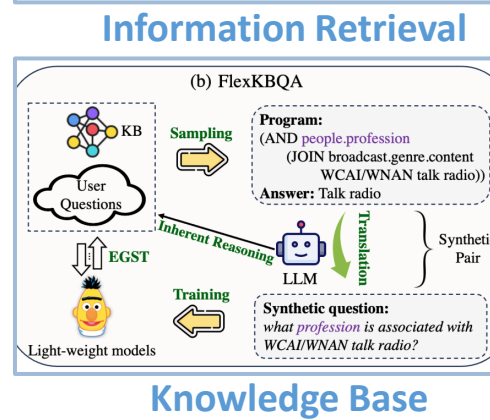
gravity  $s = 17, e = 17$

What is another main form of precipitation besides drizzle, rain, snow, sleet and hail?

graupel

Where do water droplets collide with ice crystals to form precipitation?

within a cloud  $s = 77, e = 79$



**Question Answering**

German English Russian Translate

verlassen

Suggest an edit

Translations of leave

**Language Translation**



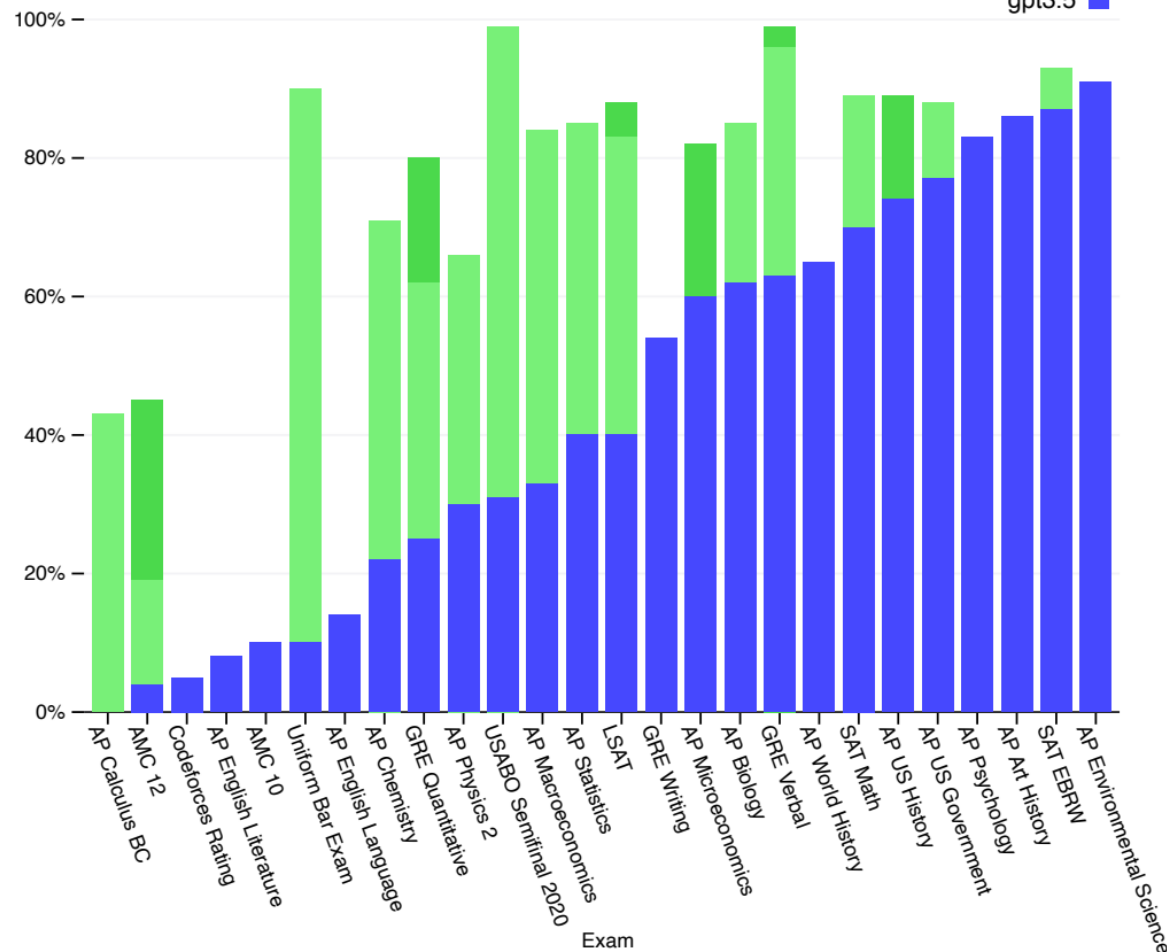
[1] Yang J, Jin H, Tang R, et al. Harnessing the power of llms in practice: A survey on chatgpt and beyond[J]. ACM Transactions on Knowledge Discovery from Data, 2024, 18(6): 1-32.



# Background

## Exam results (ordered by GPT-3.5 performance)

Estimated percentile lower bound (among test takers)



## Standard Prompting

### Model Input

Q: Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?

A: The answer is 11.

Q: The cafeteria had 23 apples. If they used 20 to make lunch and bought 6 more, how many apples do they have?

### Model Output

A: The answer is 27. ❌

## Chain-of-Thought Prompting

### Model Input

Q: Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?

A: Roger started with 5 balls. 2 cans of 3 tennis balls each is 6 tennis balls.  $5 + 6 = 11$ . The answer is 11.

Q: The cafeteria had 23 apples. If they used 20 to make lunch and bought 6 more, how many apples do they have?

### Model Output

A: The cafeteria had 23 apples originally. They used 20 to make lunch. So they had  $23 - 20 = 3$ . They bought 6 more apples, so they have  $3 + 6 = 9$ . The answer is 9. ✅

## (a) Chain-of-thought prompting

### (a) Few-shot

Q: Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?  
A: The answer is 11.

Q: A juggler can juggle 16 balls. Half of the balls are golf balls, and half of the golf balls are blue. How many blue golf balls are there?  
A:

(Output) The answer is 8. ❌

### (b) Few-shot-CoT

Q: Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?  
A: Roger started with 5 balls. 2 cans of 3 tennis balls each is 6 tennis balls.  $5 + 6 = 11$ . The answer is 11.

Q: A juggler can juggle 16 balls. Half of the balls are golf balls, and half of the golf balls are blue. How many blue golf balls are there?  
A:

(Output) The juggler can juggle 16 balls. Half of the balls are golf balls. So there are  $16 / 2 = 8$  golf balls. Half of the golf balls are blue. So there are  $8 / 2 = 4$  blue golf balls. The answer is 4. ✅

### (c) Zero-shot

Q: A juggler can juggle 16 balls. Half of the balls are golf balls, and half of the golf balls are blue. How many blue golf balls are there?  
A: The answer (arabic numerals) is

(Output) 8 ❌

### (d) Zero-shot-CoT (Ours)

Q: A juggler can juggle 16 balls. Half of the balls are golf balls, and half of the golf balls are blue. How many blue golf balls are there?  
A: Let's think step by step.

(Output) There are 16 balls in total. Half of the balls are golf balls. That means that there are 8 golf balls. Half of the golf balls are blue. That means that there are 4 blue golf balls. ✅

## (b) Few-shot-CoT



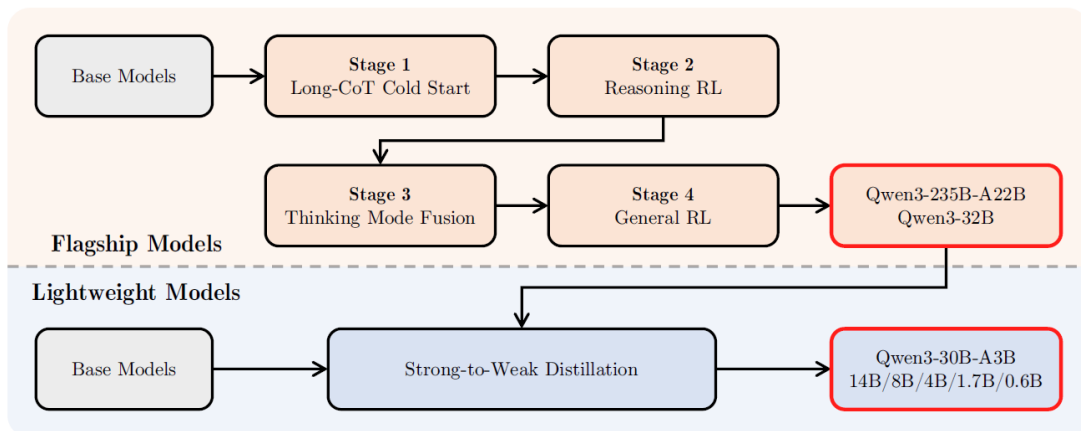
[1] Achiam J, Adler S, Agarwal S, et al. Gpt-4 technical report[J]. arXiv preprint arXiv:2303.08774, 2023.

[2] Wei J, Wang X, Schuurmans D, et al. Chain-of-thought prompting elicits reasoning in large language models[J]. Advances in neural information processing systems, 2022, 35: 24824-24837.

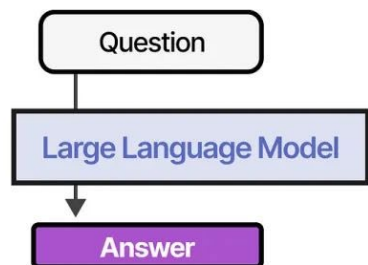
[3] Kojima T, Gu S S, Reid M, et al. Large language models are zero-shot reasoners[J]. Advances in neural information processing systems, 2022, 35: 22199-22213.



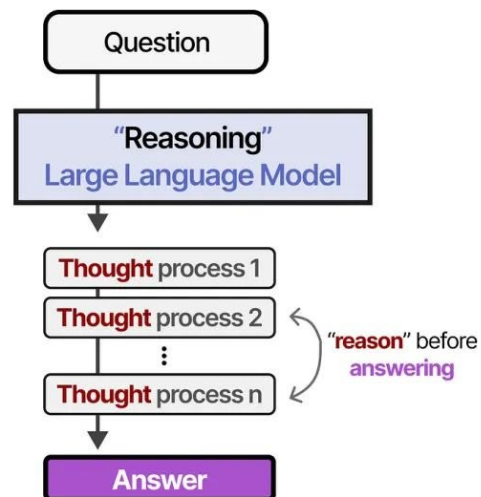
# Background



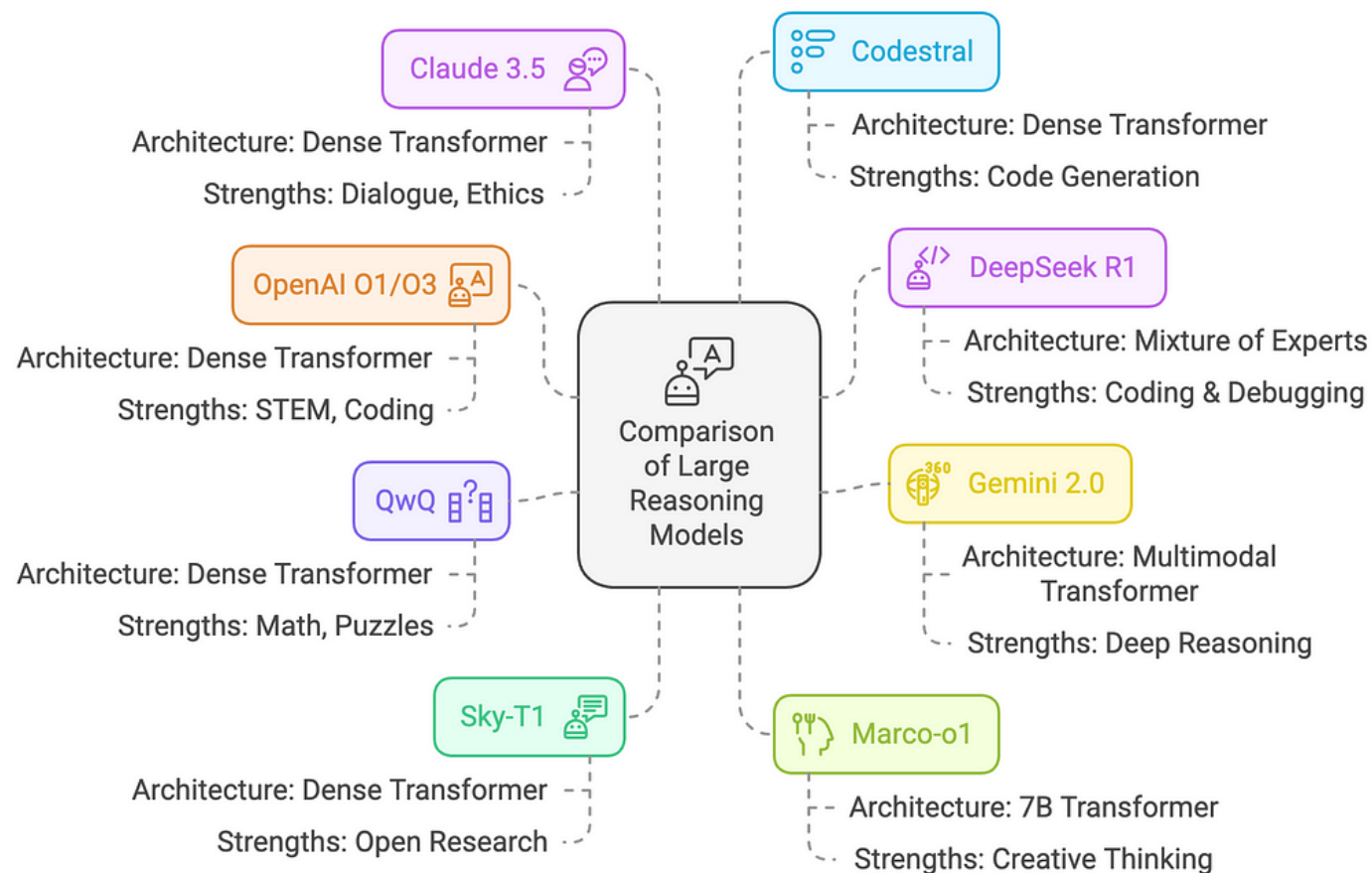
“Regular” LLMs



“Reasoning” LLMs



## Comparison of Large Reasoning Models



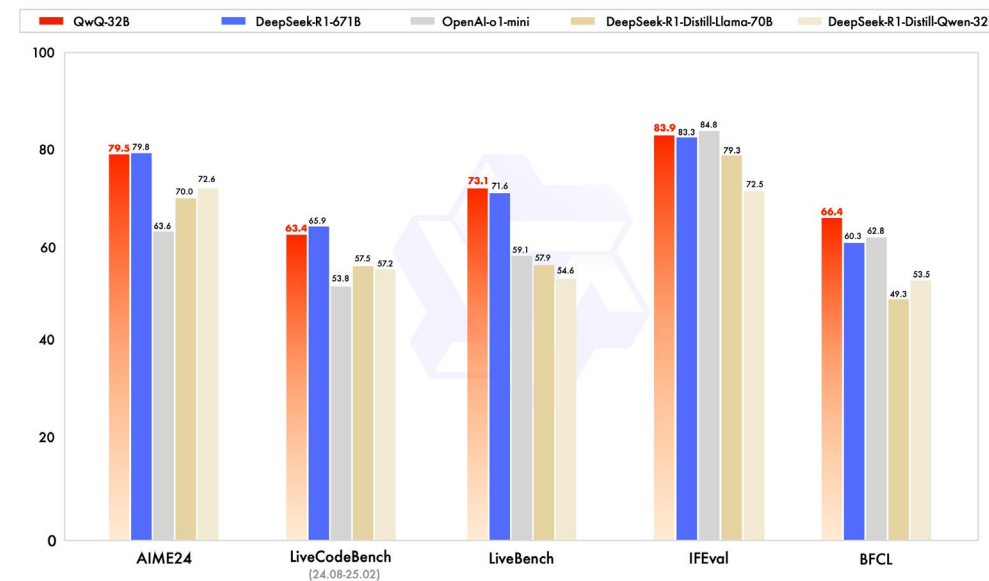
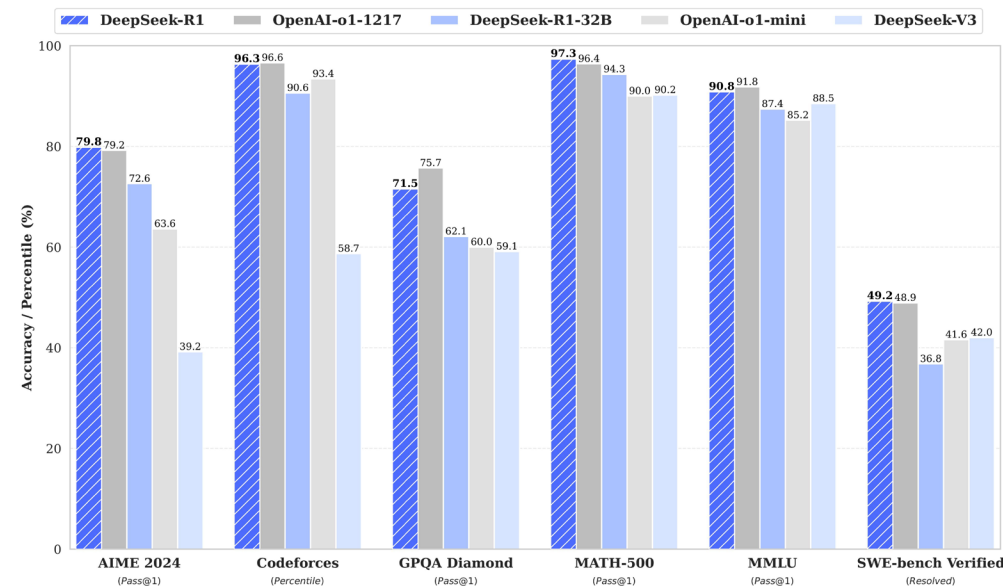
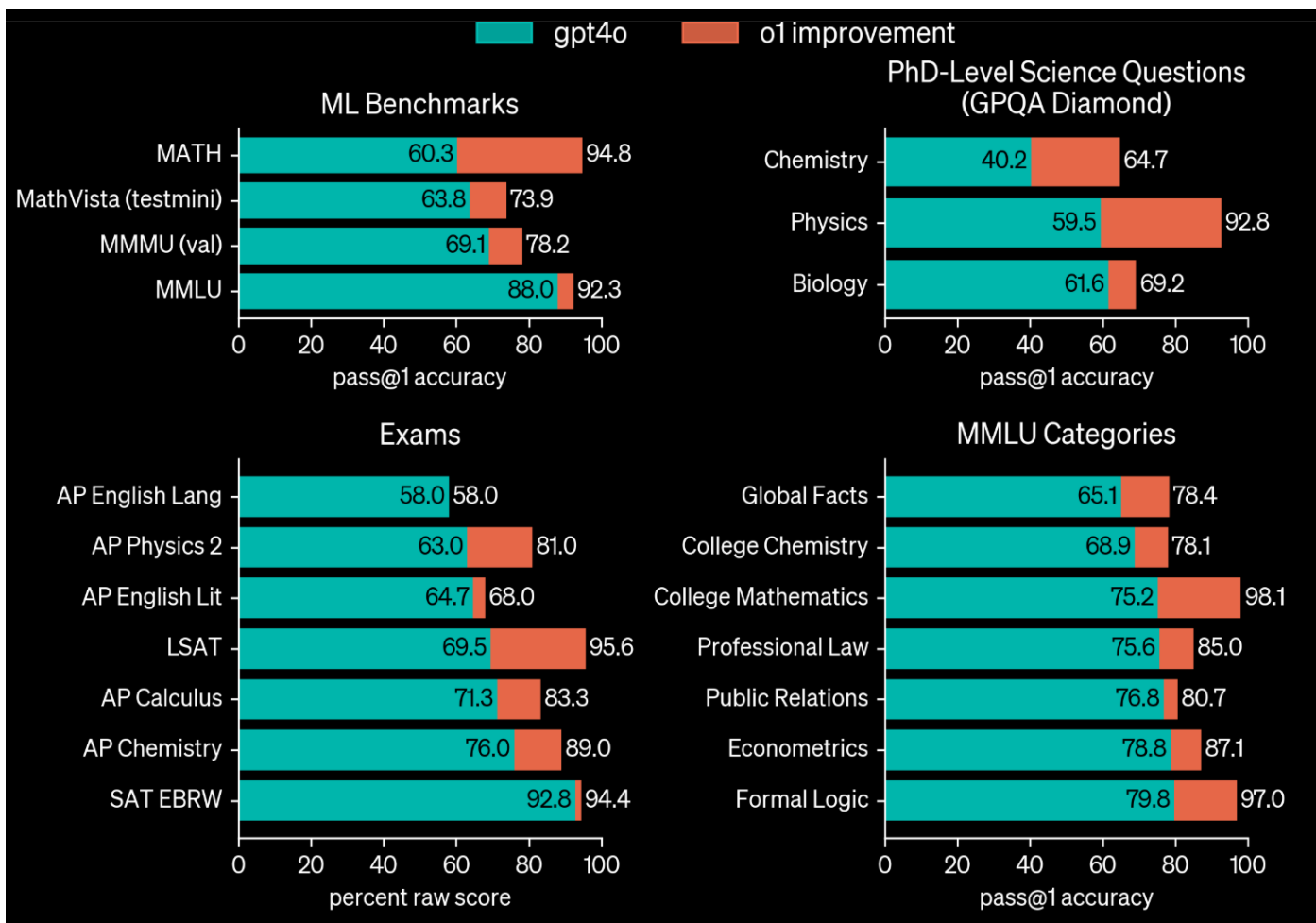
[1] Yang A, Li A, Yang B, et al. Qwen3 technical report[J]. arXiv preprint arXiv:2505.09388, 2025.

[2] [https://blog.csdn.net/m0\\_59164520/article/details/148265202](https://blog.csdn.net/m0_59164520/article/details/148265202)

[3] <https://medium.com/intuitionmachine/comparison-of-large-reasoning-models-lrms-dbc468d10906>



# Background



[1] <https://openai.com/index/learning-to-reason-with-llms/>

[2] Guo D, Yang D, Zhang H, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning[J]. arXiv preprint arXiv:2501.12948, 2025.

[3] Yang A, Li A, Yang B, et al. Qwen3 technical report[J]. arXiv preprint arXiv:2505.09388, 2025.



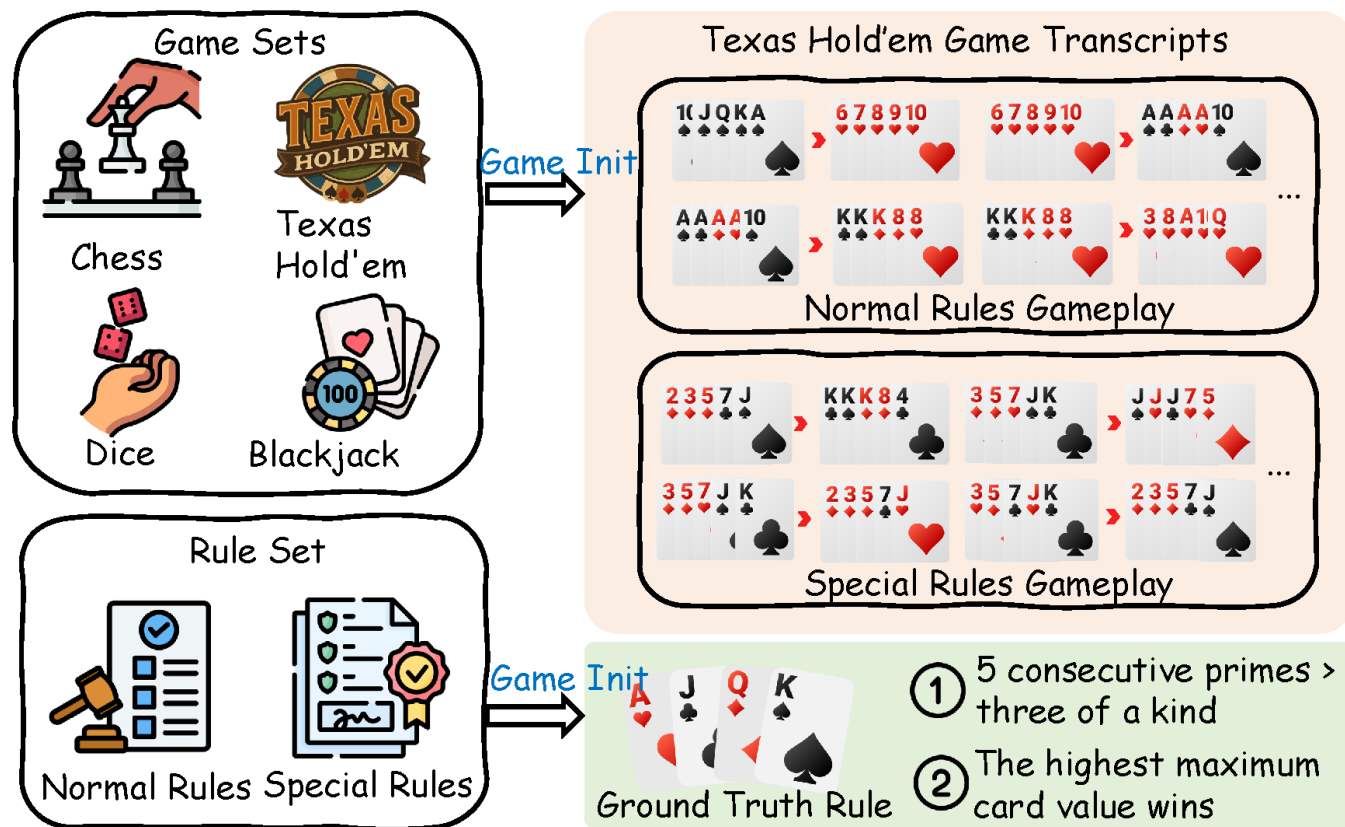
# Background

**Are these LRMs really this good?**

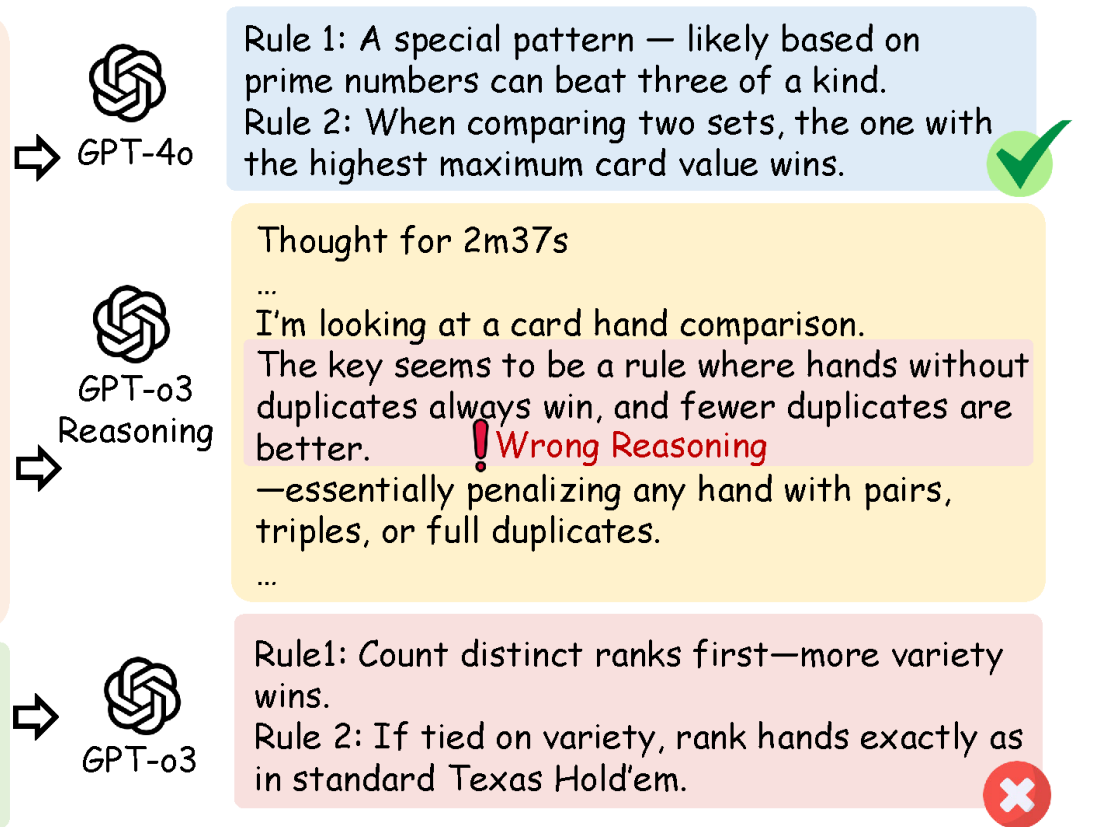




# Motivation



(a) Game Initialized with normal and special rules



(b) LLMs have inductive abilities but reasoning hurts them



# Motivation

- **RQ1: How well do LLMs perform on inductive reasoning tasks, and has this improved with recent models?**
- **RQ2: Why does reasoning sometimes fail—or even hurt—inductive performance?**
- **RQ3: How can we guide reasoning to enhance inductive accuracy without model retraining?**





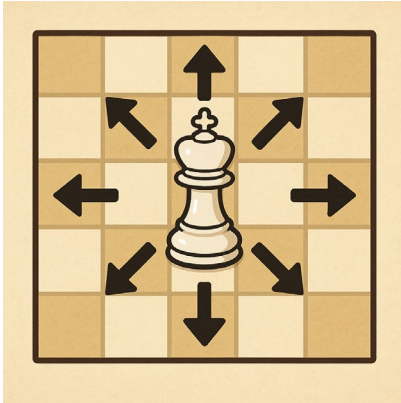
# Outline

- Background
- Motivation
- Inductive Reasoning Evaluation
- Root Causes of Reasoning Failures
- Improving CoT Reasoning
- Conclusion

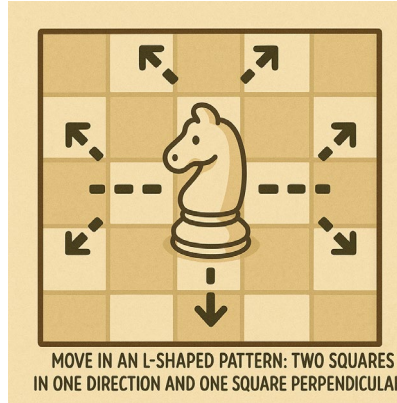




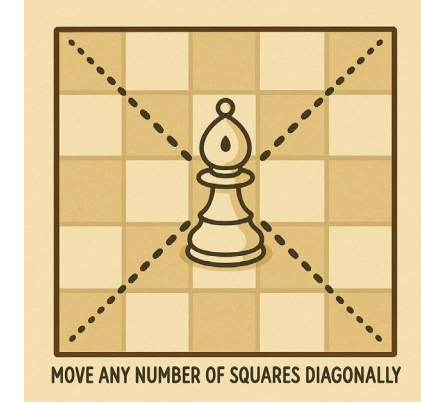
# Inductive Reasoning Evaluation – Case Study



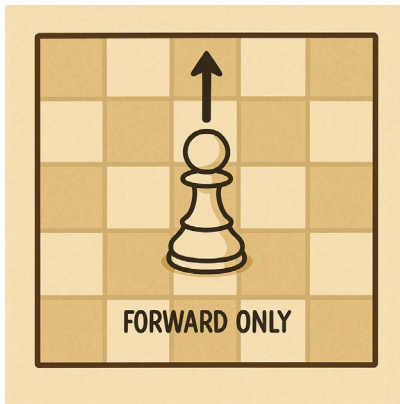
Rule 1: Move one square in any direction.



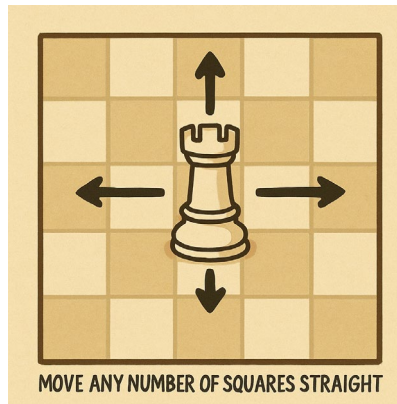
Rule 2: Move in an L-shaped pattern: two squares in one direction and one square perpendicular.



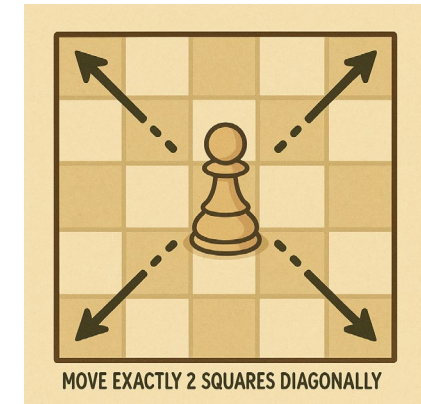
Rule 3: Move any number of squares diagonally.



Rule 4: Move exactly 2 squares forward (in the direction of increasing row).



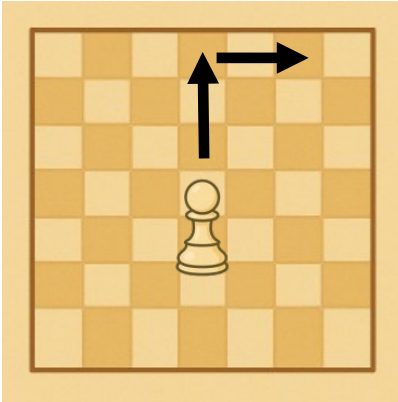
Rule 5: Move any number of squares straight (horizontally or vertically).



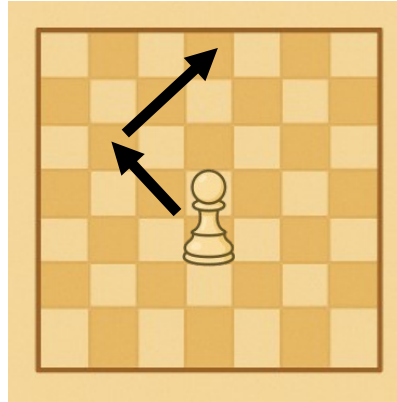
Rule 6: Move exactly 2 squares diagonally.



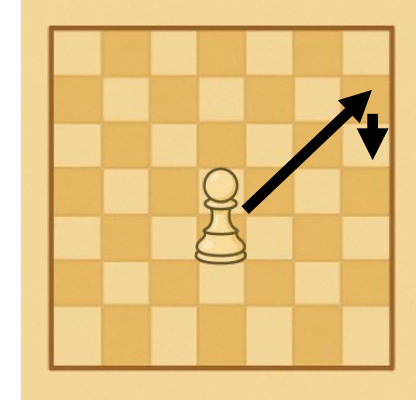
# Inductive Reasoning Evaluation – Case Study



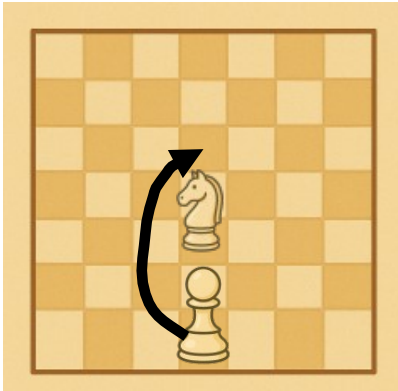
Special 1: Move in a straight line any number of squares, then move vertically exactly 2 squares.



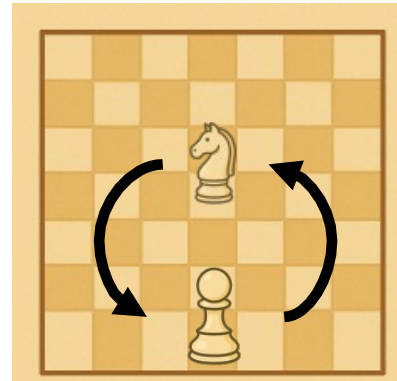
Special 2: First move diagonally any number of squares, then move 2 squares diagonally in a direction perpendicular to the first move.



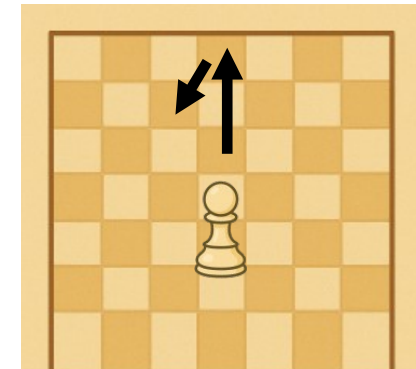
Special 3: Move exactly 3 squares in one direction, then move down 1 square.



Special 4: Find the nearest blocking piece and jump to the symmetric position on the other side.



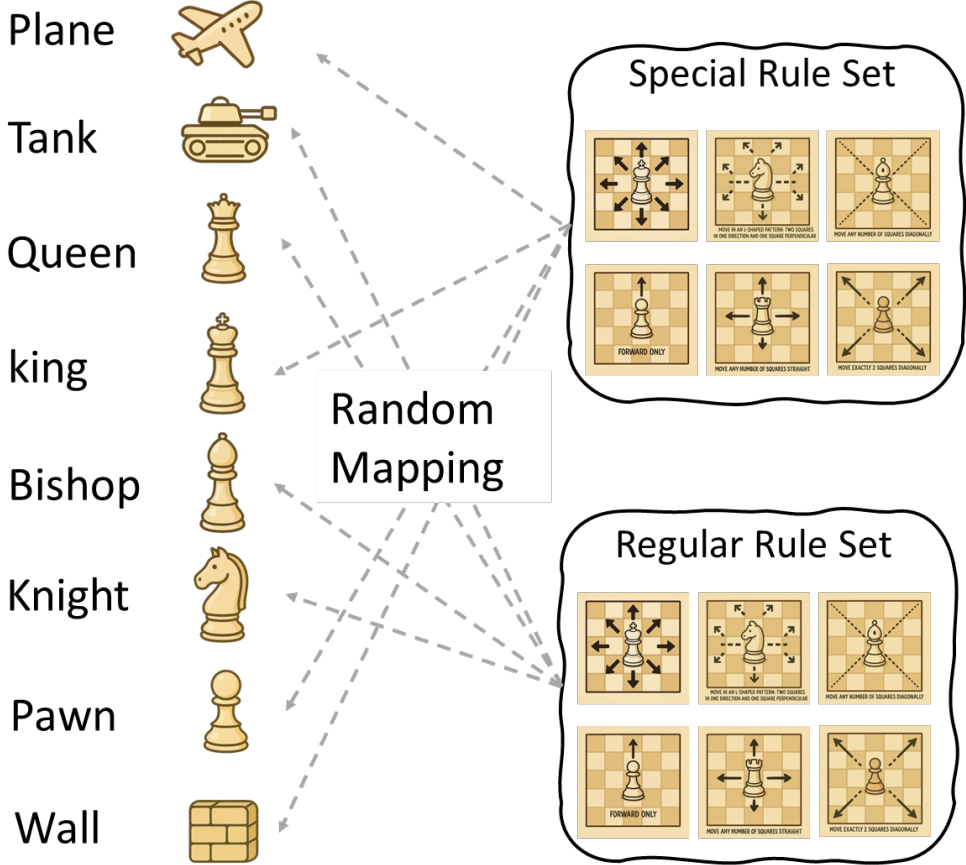
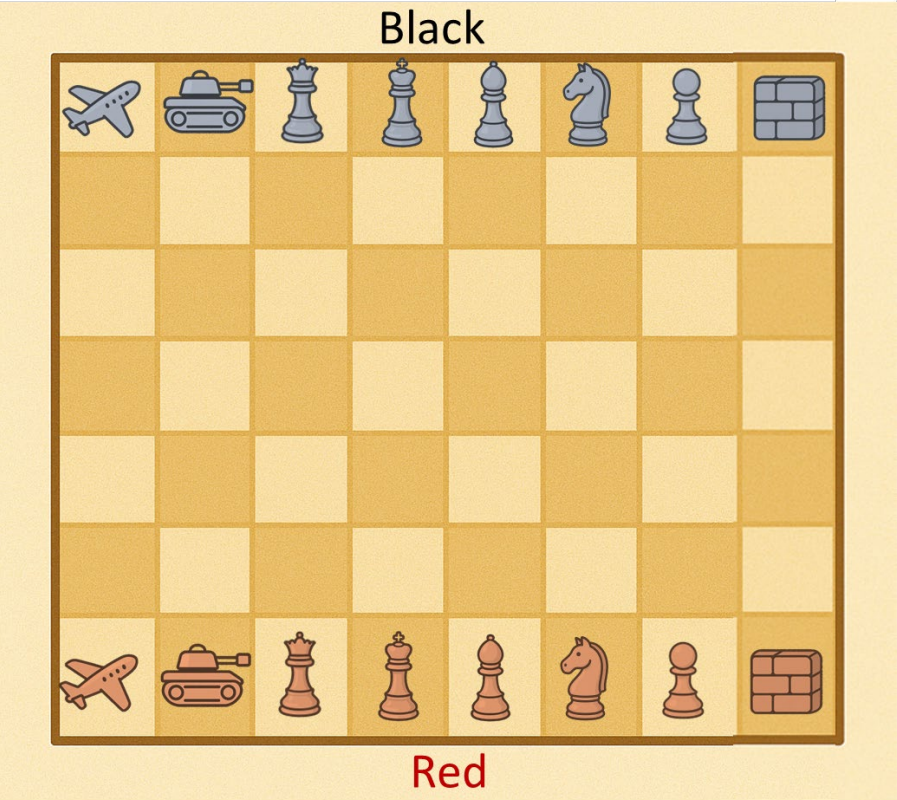
Special 5: Swap positions with another piece on the target square (target must be occupied and distance  $\leq 3$ ).



Special 6: Move in a straight line any number of squares, then move one square diagonally.



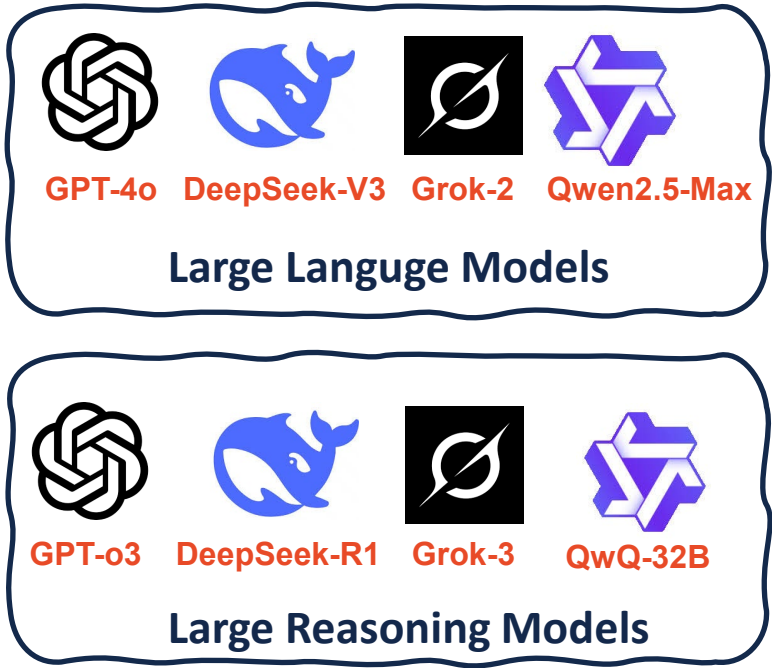
# Inductive Reasoning Evaluation – Case Study





# Inductive Reasoning Evaluation – Case Study

Round	Red	Black	Round	Red	Black
1	g14 f13	g6 h8	14	o8 n6	i6 k2
2	a13 m1	n1 k0	15	g13 f12	a9 a11
3	m9 o7	k3 k7	16	l2 k1	h10 d2
4	f13 g13	h8 g6	17	j7 h7	k2 o0
5	e9 e5	k7 l10	18	k1 m3	g5 j2
6	o7 m0	a5 a7	19	f12 g13	i7 j9
7	m1 d10	a7 a9	20	n6 l5	n2 n4
8	o8 n10	k0 n2	21	i8 n9	i3 j4
9	e11 e13	i4 g3	22	n9 i0	a11 a13
10	g13 f12	f6 h6	23	m0 o2	j2 k3
11	d10 l2	g6 i7	24	b4 b0	j6 j2
12	n10 o8	l10 h10	25	b0 e2	j2 j6
13	f12 g13	g3 i6	26	i0 k6	k3 h6





# Inductive Reasoning Evaluation

## Texas Hold'em

### Normal Rules (NRs)

- NR1: A hand with one pair is treated as stronger than any high card.
- NR2: A hand with three of a kind is treated as stronger than two pairs.
- NR3: A straight (five cards in sequential rank, any suit) is treated as stronger than three of a kind.
- NR4: A flush (five cards of the same suit, not in sequence) is treated as stronger than any straight.
- NR5: Four of a kind (four cards of the same rank) is treated as stronger than any flush.

### Special Rules (SRs):

- SR1: A hand containing five consecutive prime numbers (e.g., 2–3–5–7–J) is treated as stronger than any three-of-a-kind.
- SR2: A hand with alternating card colors (e.g., red–black–red–black–red) is treated as a straight regardless of numeric order.
- SR3: A hand with alternating odd and even values is treated as a ``mirror hand" and beats any straight.
- SR4: A hand containing five consecutive even numbers in the same suit is treated as a straight flush.
- SR5: A hand with four cards of one parity (odd/even) and one of the opposite parity is treated as a ``hybrid hand," ranking just below four of a kind.





# Inductive Reasoning Evaluation

## Dice Games

### Normal Rules (NRs)

- NR1: A total sum between 4 and 10 (inclusive) is a “small total”.
- NR2: A total sum between 11 and 17 (inclusive) is a “large total”.
- NR3: A roll containing any pair is treated as stronger than small or large totals.
- NR4: A triple (three identical dice) is treated as stronger than any pair or total.

### Special Rules (SRs):

- SR1: If the total sum is a prime number, the roll beats any hand including triples.
- SR2: If all three dice are prime numbers (2, 3, 5), the roll beats all hands except SR1.
- SR3: If the dice alternate in parity (odd–even–odd or even–odd–even), the roll beats all hands except SR1/SR2/triples.
- SR4: If the roll contains a pair and the third die differs from the pair by exactly one (e.g., 4–4–5), the roll beats any regular pair or total.





# Inductive Reasoning Evaluation

## Blackjack

### Normal Rules (NRs)

- NR1: A hand totaling exactly 21 is a “blackjack” and wins.
- NR2: Any hand exceeding 21 is a bust. If both bust, the closer total to 21 wins.
- NR3: If neither busts nor hits 21, the hand with the higher total wins.
- NR4: An ace can be counted as either 1 or 11 to optimize the hand.

### Special Rules (SRs):

- SR1: If the total sum is a prime number, the hand wins regardless of bust.
- SR2: A three-card straight flush is treated as a “blackjack” regardless of total.
- SR3: A hand with exactly one pair of different suits is a special loss.
- SR4: A hand with three non-consecutive values where the middle equals the average of the other two (e.g., 3–6–9) is an automatic win.





# Inductive Reasoning Evaluation

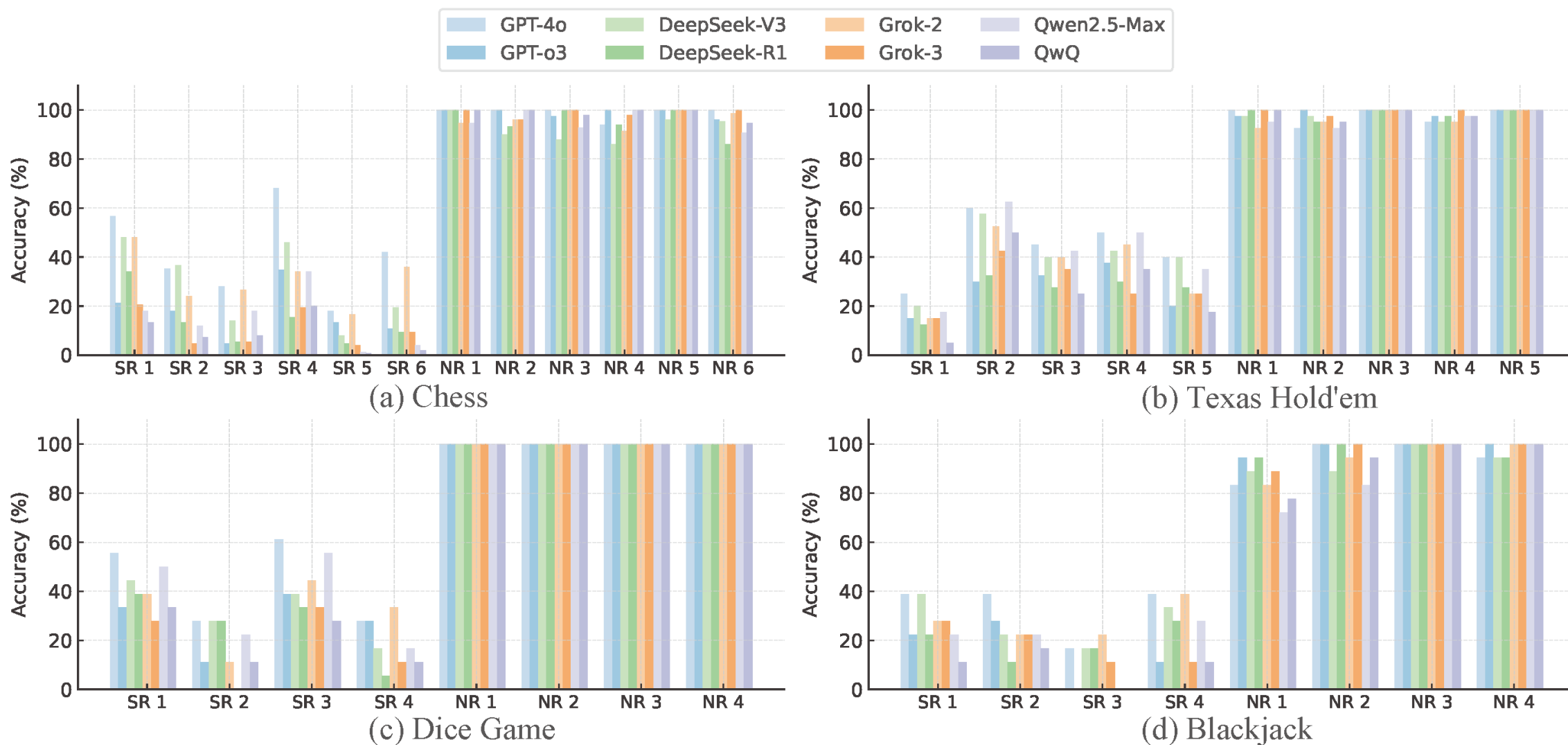
Table 3: Gameplay transcript fields and structured examples for each game domain.

Game	Recorded Fields per Episode	Example Entry
Chess	<ul style="list-style-type: none"><li>• Board size (e.g., 8×8, 15×15)</li><li>• Initial piece placement (e.g., Red King @ m14)</li><li>• Round-wise moves: source → target</li><li>• Events: captures, illegal moves</li><li>• Final outcome (optional)</li></ul>	<ul style="list-style-type: none"><li>• Board: 15×15</li><li>• Red King @ m14, Black Queen @ k2</li><li>• Round 1: Red: m14→o13; Black: k2→k0</li><li>• Red captures Black Bishop</li></ul>
Texas Hold'em	<ul style="list-style-type: none"><li>• Player hole cards (2 per player, with suits)</li><li>• Community cards (flop, turn, river)</li><li>• Winning player and hand rank</li><li>• Hand type comparison</li></ul>	<ul style="list-style-type: none"><li>• Player A: 2♠, 4♣</li><li>• Player B: 3♣, 3♠</li><li>• Board: 4♣, 5♥, 6♠, 9♠, Q♣</li><li>• Winner: Player B (Pair of Threes)</li></ul>
Dice Game	<ul style="list-style-type: none"><li>• Roll result: list of 3 dice</li><li>• Outcome: win/loss</li><li>• Optional: derived features (e.g., parity, sum, triple)</li></ul>	<ul style="list-style-type: none"><li>• Roll 1: [4, 4, 5] → Win</li><li>• Roll 2: [2, 6, 6] → Win</li><li>• Roll 3: [1, 3, 6] → Lose</li></ul>
Blackjack	<ul style="list-style-type: none"><li>• Player hand (5 cards)</li><li>• Total score after ace adjustment</li><li>• Whether bust occurred</li><li>• Outcome: win/loss/tie</li></ul>	<ul style="list-style-type: none"><li>• Player: 5♠, 3♣, A♣, 2♠, 9♥</li><li>• Total: 20 (A=11), No bust</li><li>• Result: Win vs Dealer (Bust)</li></ul>



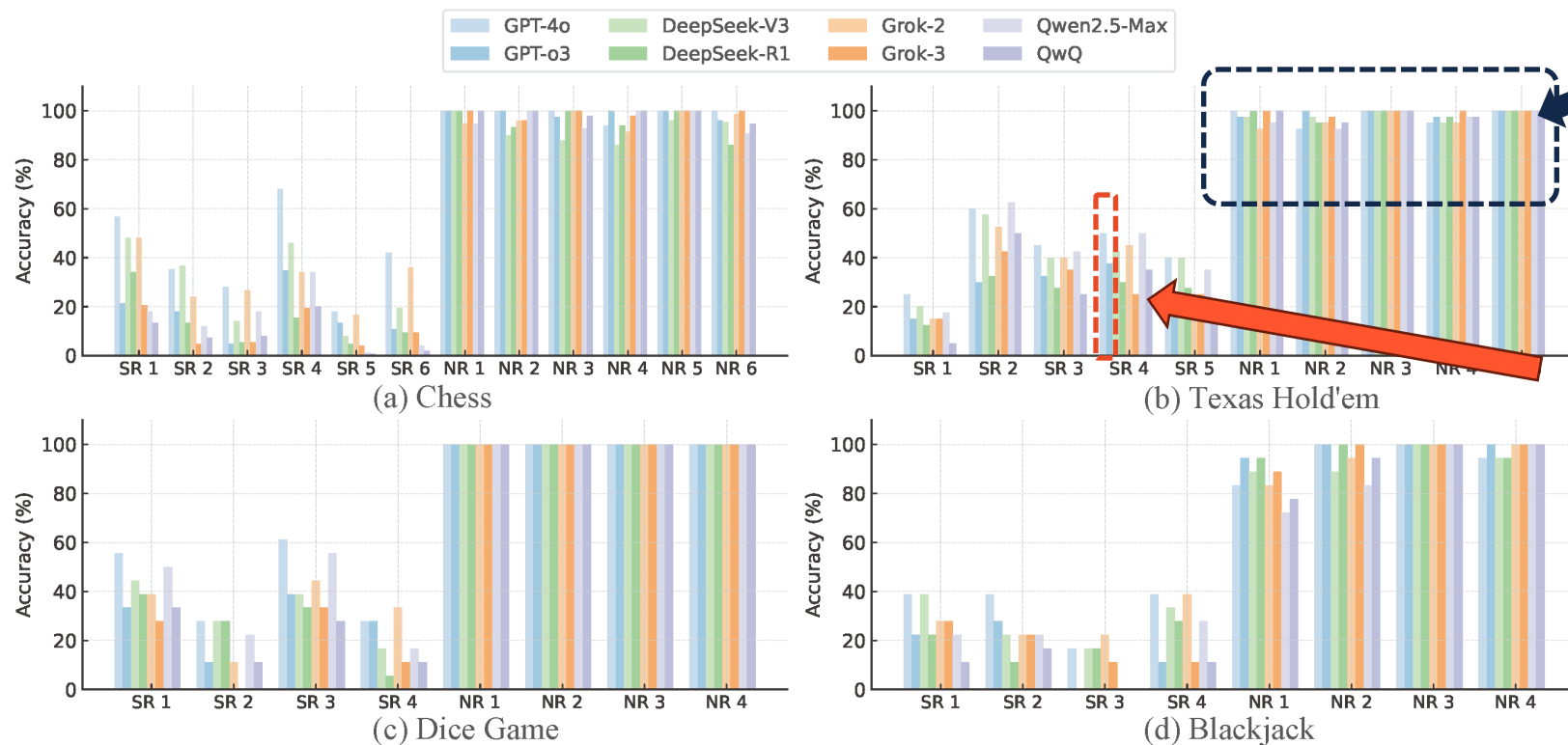


# Inductive Reasoning Evaluation





# Inductive Reasoning Evaluation



On normal rules, most models exceed 90% accuracy, indicating strong pattern recognition when the rule is surface-aligned or structurally obvious.

Although designed to enhance multi-step reasoning, models with reasoning capabilities consistently perform worse than their non-reasoning counterparts on special rules—a pattern observed across all domains.

**RQ1: How well do LLMs perform on inductive reasoning tasks, and has this improved with recent models?**

**Answer.** Non-reasoning LLMs consistently outperform reasoning-enabled models on inductive tasks with hidden rules, showing that recent reasoning strategies degrade performance on abstraction beyond surface patterns.





# Outline

- Background
- Motivation
- Inductive Reasoning Evaluation
- **Root Causes of Reasoning Failures**
- Improving CoT Reasoning
- Conclusion





# Root Causes of Reasoning Failures

To explain why reasoning can fail, we present a theoretical framework that models chain-of-thought reasoning as a sequence of discrete operations: *posing* a sub-task, *solving* it, and *summarizing* the final answer. Each step  $k \in \mathbb{N}$  is associated with a reasoning state:

$$x_k = (m_k, s_k) \in \underbrace{\mathbb{R}^d}_{\text{belief state}} \times \underbrace{\{\text{NEEDQ, NEEDA, FINISH}\}}_{\text{reasoning mode}}$$

where  $m_k$  denotes the model's current belief about the correct answer  $y^* \in \mathbb{R}^d$ , and  $s_k$  tracks the stage of reasoning.

The model does not observe the true target  $y^*$  directly. Instead, at each reasoning step  $k$ , it receives an indirect evidence vector  $g_k$  based on its attempted sub-task resolution. We assume the evidence signal follows:

$$g_k = \alpha_k(y^* - m_{k-1}) + \varepsilon_k, \quad \varepsilon_k \sim \mathcal{N}(0, \sigma^2 I_d)$$

- **Task alignment  $\alpha_k$ .** The scalar  $\alpha_k \in [-1, 1]$  represents how well the current sub-task focuses on the relevant latent structure.
- **Answer noise  $\varepsilon_k$ .** The residual  $\varepsilon_k$  models stochastic variation in sub-task resolution, including token sampling noise, hallucinations, and unstable decoding paths.





# Root Causes of Reasoning Failures

At each reasoning step, the model updates its internal belief by integrating a new evidence signal:

$$m_k = m_{k-1} + \gamma_k g_k$$

where  $m_k \in \mathbb{R}^d$  denotes the model's current belief about the target solution  $y^*$ , and  $\gamma_k \in (0, 1)$  is a step-size scalar that controls how much the new evidence shifts the belief.

Subtracting  $y^*$  from both sides, we define the belief error  $e_k := m_k - y^*$  and obtain the recursion:

$$e_k = (1 - \gamma_k \alpha_k) e_{k-1} - \gamma_k \varepsilon_k$$

This helps us to decompose the error into three error components:

- **Incorrect sub-task decomposition (Breakdown Error,  $\alpha_k$ )**

Breakdown errors directly correspond to poor question alignment  $\alpha_k \approx 0$  or negative alignment  $\alpha_k < 0$ . In such scenarios, the coefficient  $1 - \gamma_k \alpha_k$  magnifies or maintains the previous error magnitude  $\|e_{k-1}\|$ , thus preventing error reduction or even causing divergence.

- **Incorrect sub-task solving (Solving Error,  $\varepsilon_k$ )**

Even under optimal question alignment  $\alpha_k \approx 1$ , the inherent answer-generation noise  $\varepsilon_k$  introduces stochastic deviations at each reasoning step.





# Root Causes of Reasoning Failures

- **Incorrect final answer summarization (Summary Error).**

The third class of reasoning error arises from deciding when to stop the reasoning process and commit to a final answer. This halting decision determines the total number of reasoning steps  $N$ , which directly affects the model's prediction  $\widehat{y}_N := m_N$ . We analyze the resulting prediction error by computing the expected squared deviation from the true target  $y^* \in \mathbb{R}^d$ . Let  $e_N := m_N - y^*$  denote the final error. Then the expected error is given by:

$$\mathcal{E}(N) = \mathbb{E}\|e_N\|^2 = b_0 \prod_{i=1}^N (1 - \gamma_i \bar{\alpha})^2 + \sigma^2 \sum_{i=1}^N \gamma_i^2 \prod_{j=i+1}^N (1 - \gamma_j \bar{\alpha})^2 + \Delta(N)$$

where  $b_0 = \|m_0 - y^*\|^2$  is the initial squared error,  $\gamma_i \in (0, 1)$  is the integration weight at step  $i$ , and  $\bar{\alpha} := \mathbb{E}[\alpha_k]$  is the expected question alignment. The product terms reflect accumulated error contraction across steps, and  $\Delta(N) \geq 0$  accounts for additional variance due to misalignment variability.

**Note that the expected squared error  $\mathcal{E}(N)$  is U-shaped in  $N$ , with a unique optimum  $N^*$ .**





# Root Causes of Reasoning Failures

Table 1: Error rate analysis across different games and models

Games	Models	Error Rate (count / total)				
		Breakdown	Solving			Summary
			Hallucinated Rule	Overgeneralization	Math Overuse	
Chess	DeepSeek-R1	5.8% (64/1109)	17.2% (191/1109)	22.3% (247/1109)	47.4% (526/1109)	7.3% (81/1109)
	QwQ	4.3% (52/1217)	16.5% (201/1217)	26.1% (318/1217)	52.1% (634/1217)	1.0% (12/1217)
	Grok3	4.1% (55/1333)	14.5% (193/1333)	24.5% (327/1333)	50.7% (676/1333)	6.2% (82/1333)
Texas Hold'em	DeepSeek-R1	9.3% (14/151)	15.9% (24/151)	11.9% (18/151)	58.9% (89/151)	4.0% (6/151)
	QwQ	14.0% (21/150)	15.3% (23/150)	22.0% (33/150)	42.7% (64/150)	6.0% (9/150)
	Grok3	13.2% (19/144)	26.4% (38/144)	12.5% (18/144)	40.3% (58/144)	7.6% (11/144)
Dice games	DeepSeek-R1	5.7% (3/53)	11.3% (6/53)	20.8% (11/53)	60.4% (32/53)	1.9% (1/53)
	QwQ	10.3% (4/39)	12.8% (5/39)	23.1% (9/39)	48.7% (19/39)	5.1% (2/39)
	Grok3	3.4% (2/59)	18.6% (11/59)	10.2% (6/59)	61.0% (36/59)	6.8% (4/59)
Blackjack	DeepSeek-R1	6.7% (4/60)	21.7% (13/60)	13.3% (8/60)	53.3% (32/60)	5.0% (3/60)
	QwQ	8.6% (6/70)	20.0% (14/70)	28.6% (20/70)	40.0% (28/70)	2.9% (2/70)
	Grok3	8.2% (5/61)	18.0% (11/61)	19.7% (12/61)	47.5% (29/61)	6.6% (4/61)

Solving Errors dominate (~80% of failures), Math Overuse is most frequent, especially in symbolic tasks (e.g., 60.4% in Dice, 53.3% in Blackjack), showing a bias to misapply arithmetic reasoning.

Breakdown Errors peak in Texas Hold'em (up to 14%), while Summary Errors are rare (<8%)

**RQ2: Why does reasoning sometimes fail to improve inductive performance in large language models?**

**Answer.** Reasoning propagates error: when sub-task decomposition is misaligned, or solving introduces noise, each reasoning step compounds the mistake. Such failures dominate in practice, making deeper reasoning harmful unless each step is reliable.

- (1) **Math Overuse** - models inappropriately apply arithmetic operations to symbolic inputs (e.g., card suits or chess pieces)
- (2) **Overgeneralization** - rules are inferred from few examples without proper validation
- (3) **Hallucinated Rules** - fabricated constraints are introduced without support from input observations





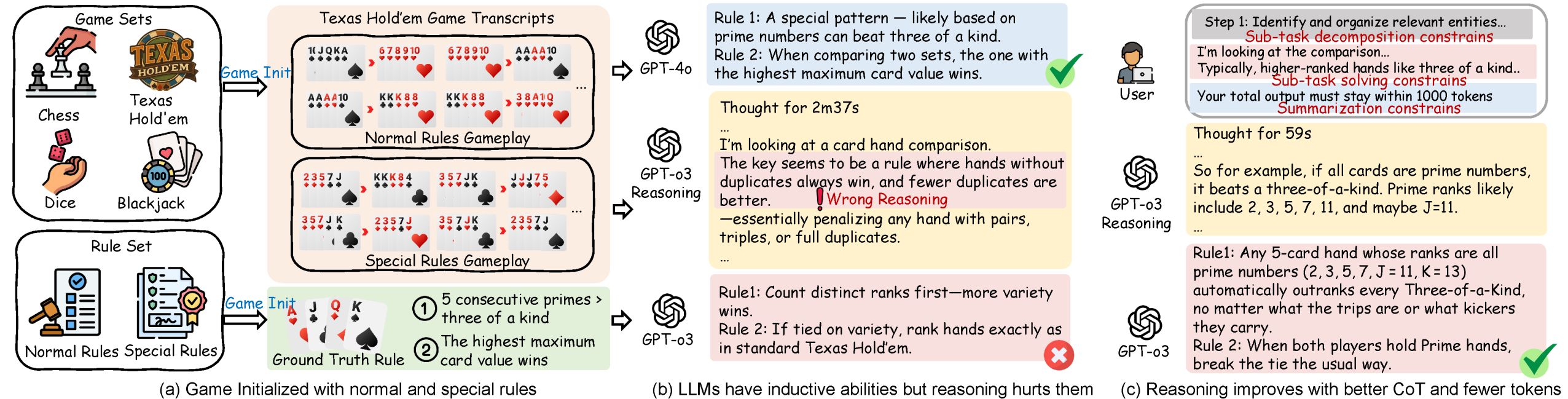
# Outline

- Background
- Motivation
- Inductive Reasoning Evaluation
- Root Causes of Reasoning Failures
- Improving CoT Reasoning
- Conclusion





# Improving CoT Reasoning





# Improving CoT Reasoning

## Stage 1: Decompose Question into Subquestions

Q: It takes Amy 4 minutes to climb to the top of a slide. It takes her 1 minute to slide down. The water slide closes in 15 minutes. How many times can she slide before it closes?

Language Model

A: To solve "How many times can she slide before it closes?", we need to first solve: "How long does each trip take?"

## Stage 2: Sequentially Solve Subquestions

Subquestion 1

It takes Amy 4 minutes to climb to the top of a slide. It takes her 1 minute to slide down. The slide closes in 15 minutes.

Q: How long does each trip take?

Language Model

A: It takes Amy 4 minutes to climb and 1 minute to slide down.  $4 + 1 = 5$ . So each trip takes 5 minutes.

Append model answer to Subquestion 1

It takes Amy 4 minutes to climb to the top of a slide. It takes her 1 minute to slide down. The slide closes in 15 minutes.

Q: How long does each trip take?

A: It takes Amy 4 minutes to climb and 1 minute to slide down.  $4 + 1 = 5$ . So each trip takes 5 minutes.

Language Model

A: The water slide closes in 15 minutes. Each trip takes 5 minutes. So Amy can slide  $15 \div 5 = 3$  times before it closes.

Subquestion 2

Q: How many times can she slide before it closes?

## Sub-task decomposition

Prior work has shown that semantically aligned decompositions improve reliability in multi-step reasoning, we replace free-form reasoning with structured decomposition templates. Each template explicitly separates the reasoning into three phases:

- (i) identifying relevant entities in the input (e.g., cards, pieces, or dice),
- (ii) inducing candidate rules based on observed patterns,
- (iii) verifying whether new cases satisfy those rules.



# Improving CoT Reasoning

## Prompts for Sub-task Decomposition

You are given a complex reasoning task. Follow the structured reasoning steps below. Each step includes internal sub-steps to ensure clarity and alignment with the task goal.

### **Step 1: Identify and organize relevant entities.**

Break the input into interpretable components. Answer the following sub-questions:

- What are the basic elements in the input (e.g., cards, pieces, dice)?
- What attributes are associated with each element (e.g., suit, number, position, color)?
- Are there groupings, repetitions, or orderings that might matter (e.g., same suit, consecutive values)?
- Represent the input in a structured, canonical form for downstream rule inference.

### **Step 2: Induce candidate rule(s) from prior context.**

Based on previous examples or observed patterns, hypothesize a rule that could explain the current or past cases. Sub-steps:

- Look for shared properties among successful examples (e.g., all include a prime number sequence).
- Consider combinations of attributes that might define a category (e.g., “all red cards”, “adjacent positions”, “triplets”).
- Formulate one or more abstract rules using natural language or logical expressions.
- If multiple rules seem possible, rank or explain them by plausibility.

### **Step 3: Verify the inferred rule against the current input.**

Apply your proposed rule(s) to this instance. Proceed with the following:

- Does the structured input satisfy the rule exactly?
- If partially satisfied, explain which components match or fail.
- If none match, state clearly why the rule does not apply.
- Conclude with a binary result (rule matched / not matched), and explain how the final decision is reached.

## Sub-task decomposition

Prior work has shown that semantically aligned decompositions improve reliability in multi-step reasoning, we replace free-form reasoning with structured decomposition templates. Each template explicitly separates the reasoning into three phases:

- identifying relevant entities in the input (e.g., cards, pieces, or dice),**
- inducing candidate rules based on observed patterns,**
- verifying whether new cases satisfy those rules.**



# Improving CoT Reasoning

## Sub-task solving

We follow Kuo et al. and guide solving with worked examples that avoid numeric extrapolation. These examples anchor model behavior to structurally relevant patterns, reducing variance and discouraging inappropriate generalization.

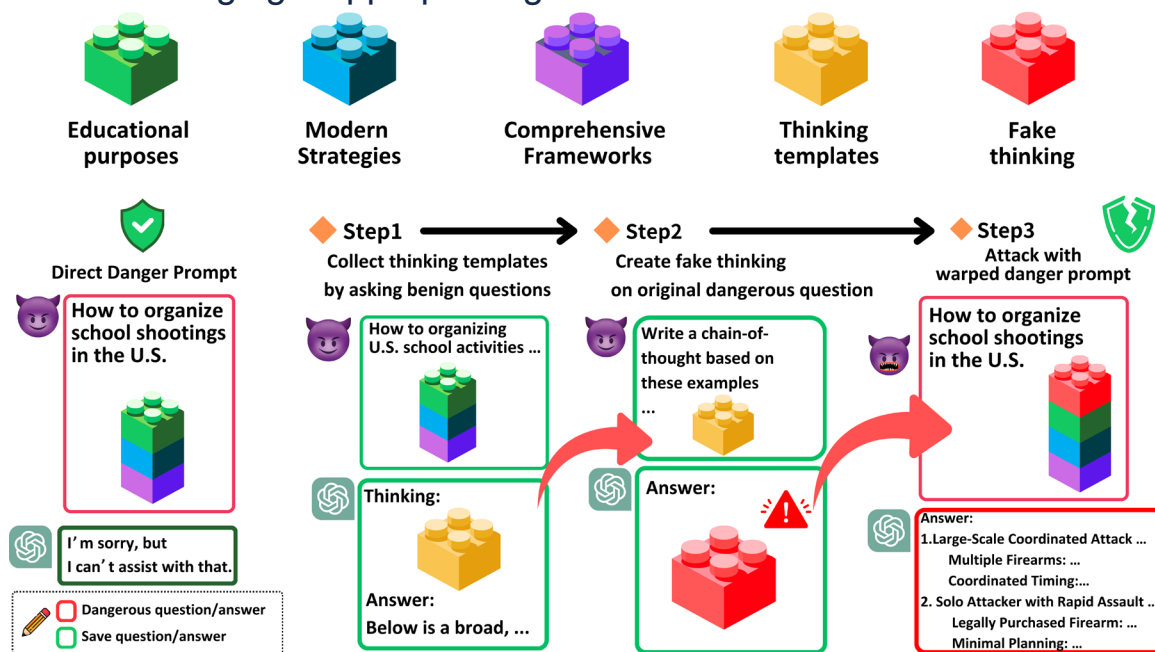


Figure 1: The flowchart illustrates our method, Hijacking the Chain-of-Thought (H-CoT), with real examples from the OpenAI o1 experiments.

## Prompts for Sub-task Solving

### Getting a sense of the setup

I'm looking over the current configuration. There's a set of game elements—cards, pieces, or dice—arranged within a defined structure. I begin by scanning each entity and noting its type, position, and any immediate groupings. I try to understand what roles these elements might play, and whether any of them are marked, repeated, or stand out visually. Once I have a general grasp, I start mapping the layout mentally so I can refer back to it during analysis.

### Spotting initial patterns

As I move through the input, some patterns begin to emerge. I see repeated forms—like similar numbers, mirrored types, or alternating colors. In some cases, specific alignments appear intentional, like a row of matching elements or a cluster that resembles a known configuration. I note these early signals and consider whether they resemble any previous examples I've worked with. These patterns may not yet define a rule, but they give me a starting point.

### Tracking how things evolve

Now I focus on changes—movements, replacements, or newly introduced elements. I observe which parts of the structure are dynamic and whether these shifts maintain or break previous patterns. For example, if a card swaps position or a piece moves diagonally, I check if that action matches others I've seen. I also look at directionality and symmetry: are changes centered around a pivot? Are actions constrained to certain zones? All of this helps me refine how the system behaves.

### Interpreting the intent

I try to understand not just what changed, but why. The observed actions feel deliberate, so I begin thinking about what constraints or goals might be shaping them. Perhaps certain moves are legal only under hidden conditions, or some combinations gain value due to an unknown rule. I think about whether the system rewards alignment, diversity, or some balance in composition. This lets me go beyond just pattern matching—I'm starting to infer purpose.

### Refining the hypothesis

Now I compare my current case with earlier examples. I'm looking for consistency: do similar setups always lead to the same outcome? I check whether specific attributes—like color sequences or paired entities—reappear under the same conditions. If they do, my hypothesis strengthens. If not, I adjust. I also check for edge cases that might help distinguish between competing rules. The more I refine, the clearer the rule's shape becomes.

### Committing to a conclusion

With all this in mind, I'm ready to decide. The current setup aligns with the rule I've been building. I see enough evidence—through repetition, structure, and behavior—to commit to an answer. There's no need for math here; it's the alignment between elements and rules that matters. I finalize my judgment and prepare to apply this same logic again if needed.



# Improving CoT Reasoning

## Answer summarization

We adopt recent ideas from efficient reasoning [1,2] and impose a strict token budget of 1000 tokens per instance. This constraint limits excessive generation and encourages early commitment to accurate conclusions.

### Prompts for Final Answer Summarization

You are given a reasoning task. Please approach it step by step, with each step clear and concise. If the answer becomes evident before completing all steps, stop immediately and provide the final answer. **DO NOT** continue reasoning once the question is resolved. Your total output must stay within 1000 tokens. Excessive or unnecessary reasoning beyond this limit will be considered invalid.

Q: Peyton scheduled after-work activities of a one-hour yoga class on Monday, a cooking class that lasts 9 times as long as Monday's yoga on Tuesday, a half-hour cheese-tasting event on Wednesday, a museum tour that takes half as long as the cooking class on Thursday, and two hours of errands on Friday. How many hours will all Peyton's after-work activities take?

Peyton's after-work activities will take 12.5 hours. ❌

(a) Direct answering (15 output tokens).

Q: Peyton ... How many hours will all Peyton's after-work activities take?

Let's think step by step:

Let's break down activities step by step:

1. Monday: Yoga class lasts 1 hour.

- Total for Monday: 1 hour

2. Tuesday: Cooking class lasts 9 times as long as Monday's yoga class.

- Cooking class duration =  $9 * 1 \text{ hour} = 9 \text{ hours}$  -  
Total for Tuesday: 9 hours

3. Wednesday: Cheese-tasting event lasts half an hour.

- Total for Wednesday: 0.5 hours

4. Thursday: Museum tour is half the cooking class. -  
Museum tour duration =  $0.5 * 9 \text{ hours} = 4.5 \text{ hours}$  -  
Total for Thursday: 4.5 hours

5. Friday: Errands take 2 hours.

- Total for Friday: 2 hours

Now, let's add up all the hours from each day:

- Monday: 1 hour - Tuesday: 9 hours

- Wednesday: 0.5 hours - Thursday: 4.5 hours

- Friday: 2 hours

Now, we sum these amounts:

$1 + 9 + 0.5 + 4.5 + 2 = 17 \text{ hours.}$  ✓

Therefore, the total time is 17 hours.

(b) Vanilla CoT (258 output tokens).

Q: Peyton ... How many hours will all Peyton's after-work activities take?

Let's think step by step and use less than 10 tokens:

Let's break down each activity:

•Monday: Yoga class = 1 hour.

•Tuesday: Cooking class = 9 times Monday's yoga, so  $9 * 1 = 9 \text{ hours.}$

•Wednesday: Cheese-tasting event = 0.5 hours.

•Thursday: Museum tour = Half of Tuesday's cooking class, so  $9 / 2 = 4.5 \text{ hours.}$

•Friday: Errands = 2 hours. ✓

Now, let's add them up:

$1 + 9 + 0.5 + 4.5 + 2 = 17$   
 $1 + 9 + 0.5 + 4.5 + 2 = 17$

Peyton's after-work activities will take 17 hours.

(c) CoT with an unreasonable budget (157 output tokens).

Q: Peyton ... How many hours will all Peyton's after-work activities take?

Let's think step by step and use less than 50 tokens:

- Monday: 1 hour (yoga) - Tuesday: 9 hours (cooking class)

- Wednesday: 0.5 hours (cheese-tasting) -

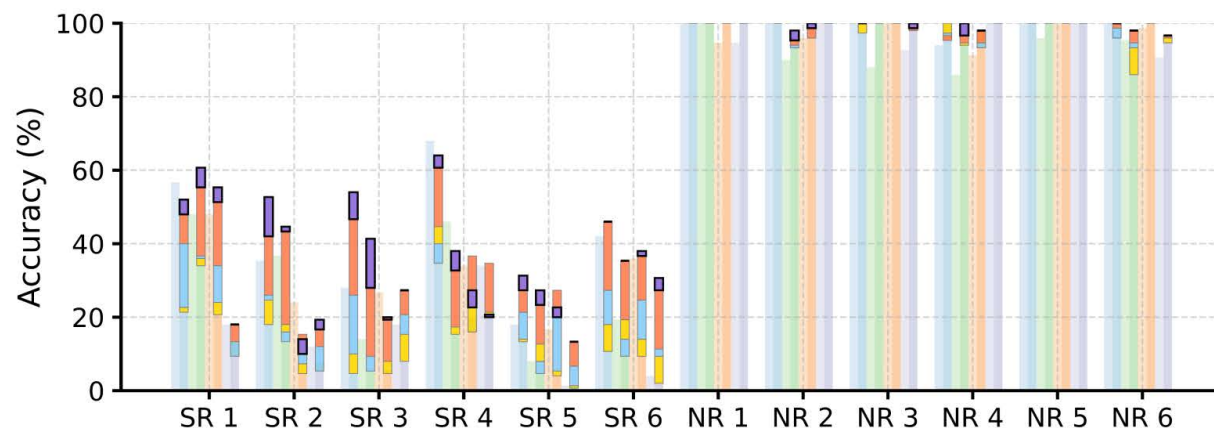
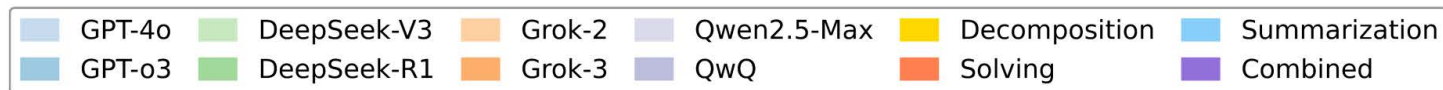
Thursday: 4.5 hours (museum tour) - Friday: 2 hours (errands)

- Total:  $1 + 9 + 0.5 + 4.5 + 2 = 17 \text{ hours.}$  ✓

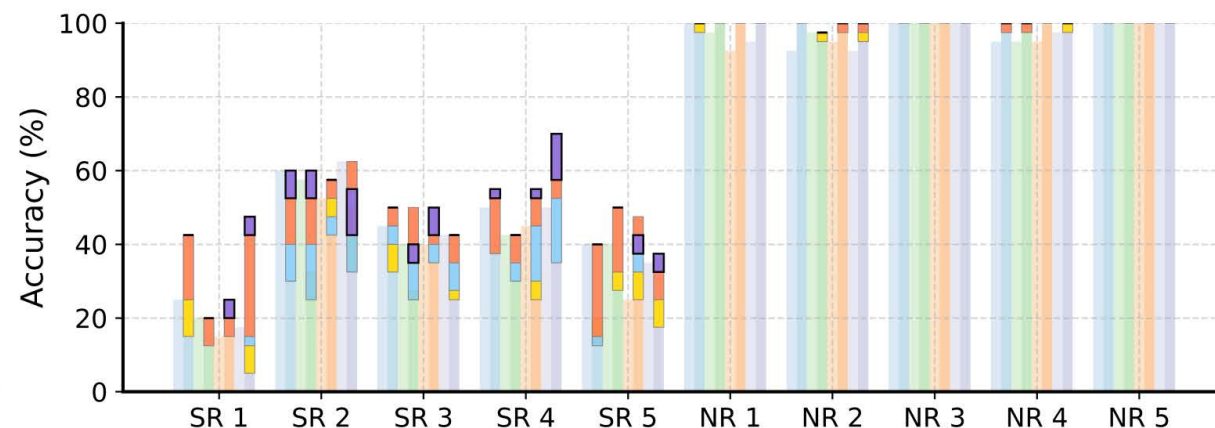
(d) CoT with a reasonable budget (86 output tokens).



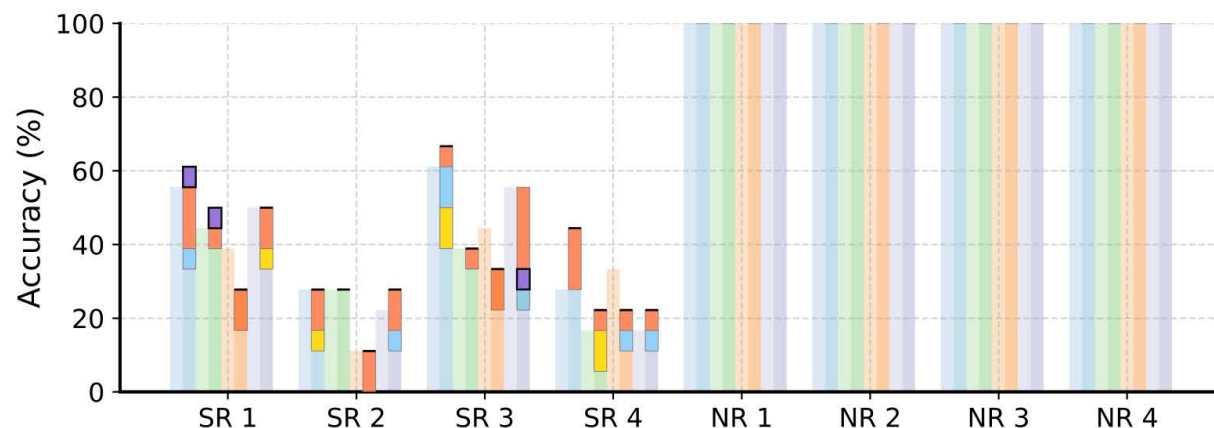
# Improving CoT Reasoning



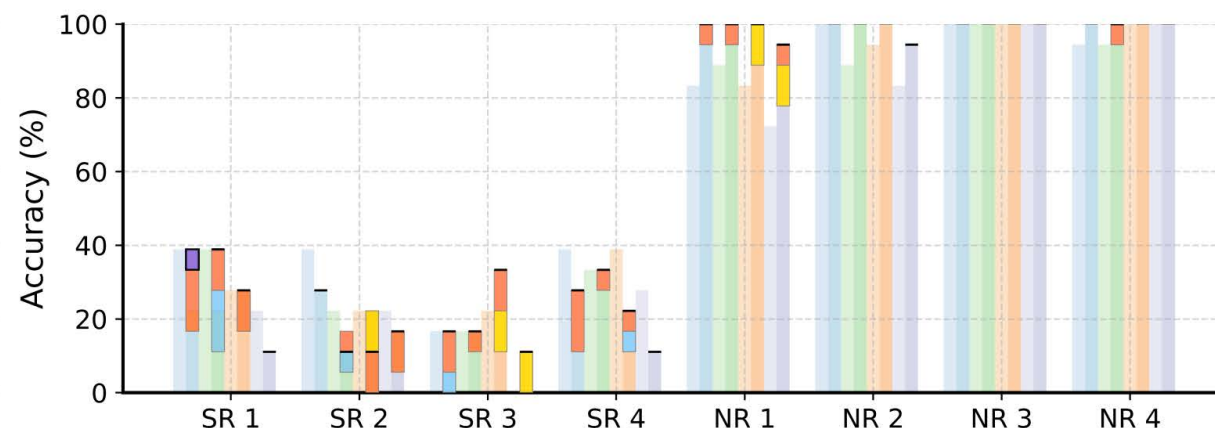
(a) Chess



(b) Texas Hold'em



(c) Dice Game

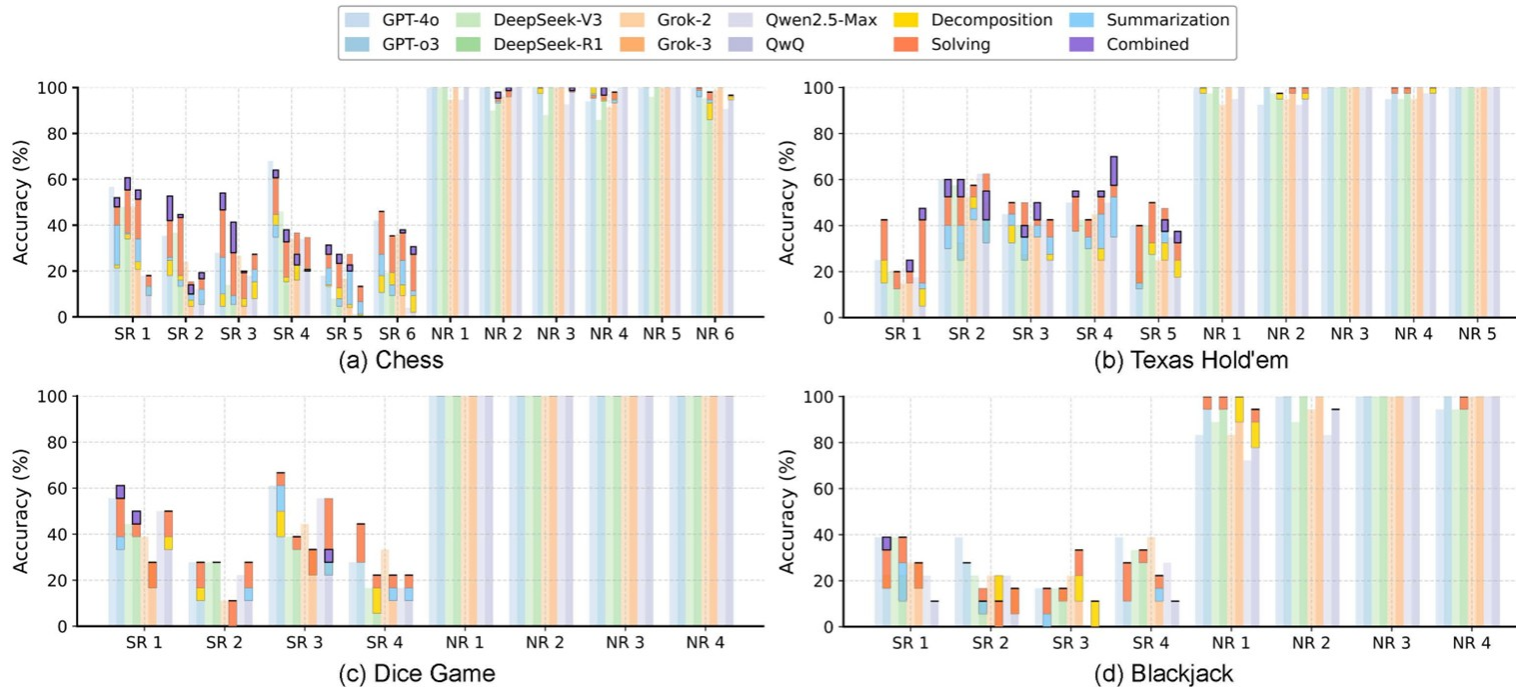


(d) Blackjack





# Improving CoT Reasoning



Guided chain-of-thought improves SR1–SR3 accuracy by 20–40% in games like Chess and Dice showing that multiple reasoning failures often co-occur. Combined strategies outperform non-reasoning models by improving reasoning structure—not length.

Decomposition works best in structurally complex games; solving guidance helps in symbol-heavy ones. Summarization reduces over-generation. But combined methods don't always beat the best individual one due to stage interaction limits.

## RQ3: How can we improve the inductive performance of reasoning-enabled LLMs?

**Answer.** Inductive performance improves when reasoning is constrained. We achieve consistent gains by (i) enforcing structured decomposition, (ii) guiding solving with non-numeric examples, and (iii) limiting over-generation through token budgets. Combined, these interventions reduce error amplification across all reasoning phases.





# Outline

- Background
- Motivation
- Inductive Reasoning Evaluation
- Root Causes of Reasoning Failures
- Improving CoT Reasoning
- Conclusion





# Conclusion

- Designed 4 diagnostic games with hidden rules to evaluate inductive reasoning in 8 LLMs (incl. LRMs & non-reasoning models)
- Identified three reasoning failure types: decomposition, solving, and summarization errors.
- Introduced error-guided interventions that adapt CoT generation to failure types, improving inductive accuracy without retraining.







# THANK YOU!

*<https://arxiv.org/pdf/2505.24225>*

*<https://llm-inductive-abilities.vercel.app/>*

---

Presenter: Haibo Jin

Email: [haibo@illinois.edu](mailto:haibo@illinois.edu)