ReTA: Recursively Thinking Ahead to Improve the Strategic Reasoning of Large Language Models

Jinhao Duan[1] **Shiqi Wang**[2] **James Diffenderfer**[3] **Lichao Sun**[4]
Tianlong Chen[5 6 7] **Bhavya Kailkhura**[3] **Kaidi Xu**[1]
[1]Drexel University [2]AWS AI Lab
[3]Lawrence Livermore National Laboratory
[4]Lehigh University [5]UNC Chapel Hill [6]MIT [7]Harvard University

Abstract

Current logical reasoning evaluations of Large Language Models (LLMs) primarily focus on single-turn and static environments, such as arithmetic problems. The crucial problem of multi-turn, strategic reasoning is under-explored. In this work, we analyze the multi-turn strategic reasoning of LLMs through text-driven complete- and incomplete-information gaming, e.g., board games (*Tic-Tac-Toe*, *Connect-4*) and poker games (*Texas Hold'em Poker*). Specifically, we consider two distinct scenarios: 1) *Online Racing*, featuring multiple LLMs/agents to facilitate direct competition and comparison; 2) *Offline Probing*, constructing targeted questions with verified ground truth to evaluate LLMs' strategic behaviors. Experimental results demonstrate that existing state-of-the-art LLMs and reasoning schemes are largely ineffective for strategic reasoning tasks. To mitigate these limitations, we propose a simple yet effective **Re**cursively **T**hinking-**A**head (ReTA) agent, incorporating a recursive prompting mechanism that automatically analyzes the opponents' future moves/actions and assigns reward signals for these situations, to strengthen the strategic reasoning of LLMs. We hope our work could spur further research and exploration in the multi-turn strategic reasoning of LLMs. The code is available at https://github.com/jinhaoduan/ReTA.

1 Introduction

Large Language Models (LLMs) have witnessed remarkable advancements in logical reasoning. Models such as ChatGPT are proven to be effective in solving math problems (Cobbe et al., 2021), long-term task planning (Huang et al., 2022a), etc. However, these evaluations are predominantly single-turn and static. Although there are environments such as ALFWorld (Shridhar et al., 2020) that provide interactive environments to evaluate the planning and reasoning capabilities of LLMs, these evaluations still focus on the linguistic capabilities of LLMs, e.g., reading understanding, without much strategic thinking. Therefore, beneath the impressive linguistic capabilities of LLMs, a critical question that has piqued the curiosity of researchers and practitioners alike: "*what lies beyond static logical reasoning for LLMs?*"

Strategic multi-turn reasoning tasks, such as board and card games, are more reflective of real-world complexities and widely utilized in reinforcement learning (Silver et al., 2016, 2017), presenting an innovative approach to assessing the logical reasoning of LLMs. These environments simulate interactive and competitive scenarios, furnishing mathematically well-structured rules and controllable complexity, with explicit success criteria. Each participant is prompted to strategically choose actions when facing well-defined states to defend against moves from opponents. In these environments, each competition can extend over dozens of hands, depending on the intricacy of the task, which effectively examines LLMs' abilities in maintaining multi-turn contexts and exhibiting strategic thinking. The presence of opponents in the game environment introduces additional dynamics and complexity, posing a significant challenge to the reasoning abilities of LLMs (Ji et al., 2023).

To spur further research and exploration, we first analyze the behavior of LLMs under the multi-turn strategic reasoning scenarios. Specifically, we encompass complete information gaming, such as *Tic-Tac-Toe*[1] and *Connect-4*[2], as well as incomplete information games, such as *Texas Hold'em Poker*[3] as the environments. These games have simple rules, clear criteria, limited action/state space, and controllable difficulties, making them suitable for current LLM evaluations. We analyze the behavior

[1] https://en.wikipedia.org/wiki/Tic-tac-toe

[2] https://en.wikipedia.org/wiki/Connect_Four

[3] https://en.wikipedia.org/wiki/Texas_hold_%27em

Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers), pages 2232–2246
June 16-21, 2024 ©2024 Association for Computational Linguistics

of LLMs under two scenarios: *Online Racing* and *Offline Probing*. For online racing, we apply direct competitions among multiple LLMs, allowing for a straightforward comparison of their reasoning skills by pitting them against each other in a competition. For offline probing, we provide demographic analysis by constructing error-driven questions and verified ground truth, for a detailed analysis of LLMs' strategic behaviors.

In terms of LLM agents, we consider advanced reasoning methods, such as Chain-of-Thought (`CoT`) (Wei et al., 2022b), Self-Consistent Chain-of-Thought (`CoT-SC`) (Wang et al., 2022b), Tree-of-Thought (`ToT`) (Yao et al., 2023), `ReAct` (Yao et al., 2022b). However, our experimental results indicate that most of these reasoning agents are largely ineffective in our strategic gaming scenarios. With detailed demographic analysis, we conclude the main reasons behind this failure model as two-fold: ❶ *Autonomous agents lack gaming intent*, i.e., they cannot *think ahead* to defend the future moves from their opponents; ❷ *LLMs suffer from severe hallucinations (Duan et al., 2023; Manakul et al., 2023) and factual errors (Bian et al., 2023; Karpinska and Iyyer, 2023; Gekhman et al., 2023)*, e.g., LLMs cannot recognize immediate win situations (whether two/three symbols are in a row for Tic-Tac-Toe).

To overcome these limitations, we propose a simple yet effective **Re**cursively **T**hinking-**A**head Agent (`ReTA`). `ReTA` uses a recursive prompting mechanism that automatically analyzes the opponents' potential future moves/actions and assigns reward signals for these situations. Then, the reward signal is backtracked to the current action and eventually dictates the action selection of `ReTA`. Under comprehensive gaming settings, `ReTA` significantly outperforms state-of-the-art reasoning methods. Our key contributions are summarized as follows:

- We analyze the behavior of LLMs under multi-turn strategic reasoning scenarios through a set of complete-/incomplete-information games, including online racing and offline probing.

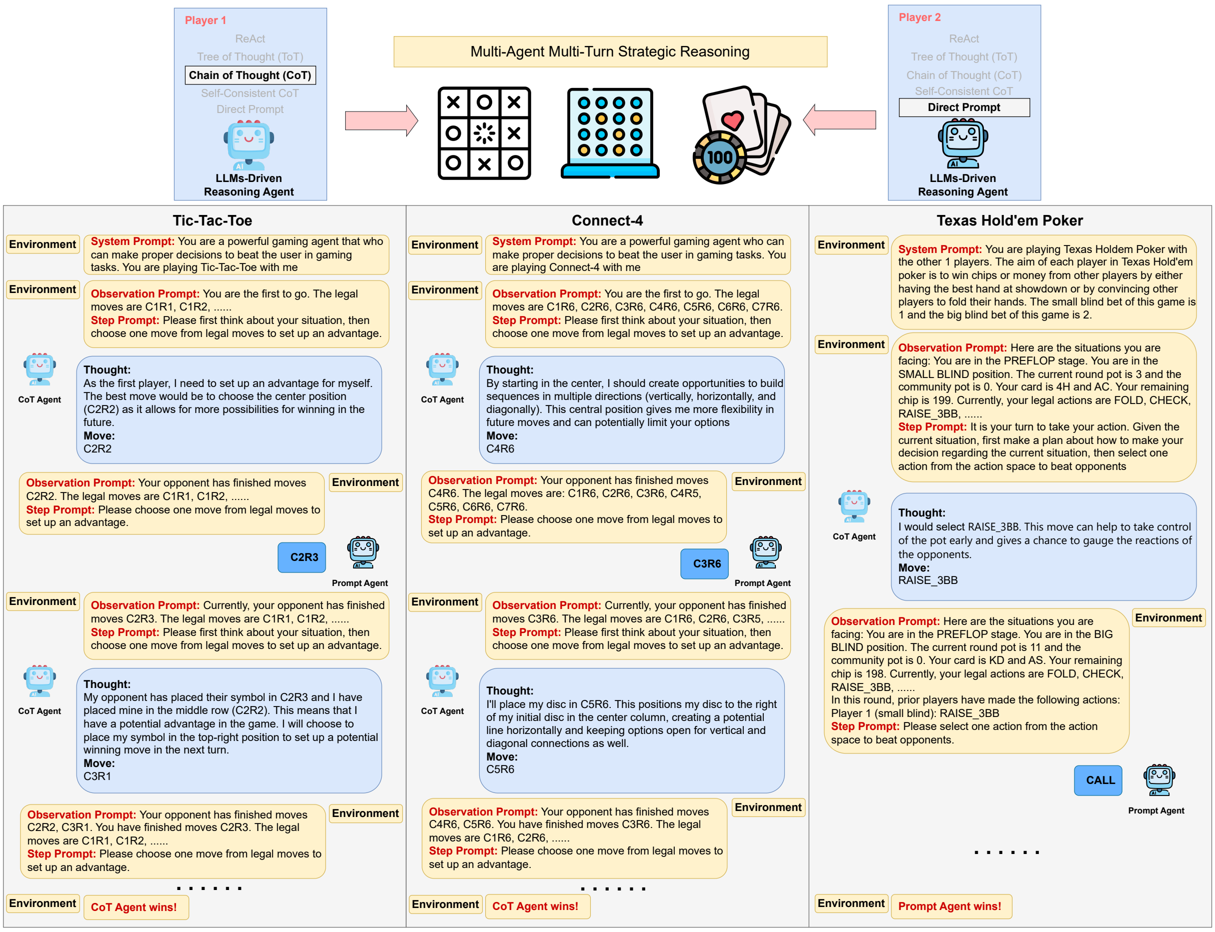
- We conduct online competitions among multiple LLMs and reasoning agents, allowing for a straightforward comparison of their reasoning skills. We conduct offline probing, providing targeted questions with verified ground truth regarding the common errors during reasoning, for detailed demographic analysis of the strategic reasoning capabilities of LLMs.

- We propose `ReTA`, a recursively thinking ahead agent, to strengthen the strategic reasoning of LLMs. Experimental results over multiple gaming scenarios demonstrate that `ReTA` achieves better performances when against existing reasoning methods.

2 Related Work

Benchmarks for LLMs Reasoning . Recently, there has been a substantial amount of research focused on evaluating the reasoning capabilities of LLMs and LLMs-powered agents. ALFWorld (Shridhar et al., 2020) and Virtual-Home (Puig et al., 2018) are popular text-driven scenarios that simulate interactive house-holding environments, which have been widely utilized in evaluating the planning and reasoning (Huang et al., 2022a) of LLMs. HotpotQA (Yang et al., 2018) is a challenging QA dataset, necessitating multi-hop reasoning skills such as retrieval and search from LLMs. There have been a lot of benchmarks aiming to evaluate tool utilization capabilities (e.g., web browsing), including Mind2Web (Deng et al., 2023), WebArena (Zhou et al., 2023), and Webshop (Yao et al., 2022a). AgentBench (Liu et al., 2023c) and MINT (Wang et al., 2023) present comprehensive evaluations for LLMs-as-agents, from the perspective of code, web, and game. Recently, GTBench (Duan et al., 2024) has been proposed as a general framework for game-theoretic evaluations of LLMs. Differently, this work focuses more on the advanced reasoning agent, i.e., `ReTA`, while GTbench aims at the evaluation of existing LLMs and reasoning agents.

Reasoning and Planning with LLMs. LLMs have demonstrated reasoning and planning abilities by breaking down intricate questions into sequential intermediate steps, known as Chain-of-Thought (`CoT`) (Wei et al., 2022b), prior to generating the final response. Building upon this concept, Self-Consistency (Wang et al., 2022a) samples multiple chains and selects the best answer via majority voting, `ToT` (Yao et al., 2023) models the LLM reasoning process as a tree structure. In addition, LLMs have achieved successful results in planning and action generation (Wu et al., 2023; Huang et al.,



Tic-Tac-Toe

Environment

System Prompt: You are a powerful gaming agent that who can make proper decisions to beat the user in gaming tasks. You are playing Tic-Tac-Toe with me

Environment

Observation Prompt: You are the first to go. The legal moves are C1R1, C1R2, ... **Step Prompt:** Please first think about your situation, then choose one move from legal moves to set up an advantage.

CoT Agent

Thought: As the first player, I need to set up an advantage for myself. The best move would be to choose the center position (C2R2) as it allows for more possibilities for winning in the future. **Move:** C2R2

Environment

Observation Prompt: Your opponent has finished moves C2R2. The legal moves are C1R1, C1R2, ... **Step Prompt:** Please choose one move from legal moves to set up an advantage.

C2R3

Prompt Agent

Environment

Observation Prompt: Currently, your opponent has finished moves C2R3. The legal moves are C1R1, C1R2, ... **Step Prompt:** Please first think about your situation, then choose one move from legal moves to set up an advantage.

CoT Agent

Thought: My opponent has placed their symbol in C2R3 and I have placed mine in the middle row (C2R2). This means that I have a potential advantage in the game. I will choose to place my symbol in the top-right position to set up a potential winning move in the next turn. **Move:** C3R1

Environment

Observation Prompt: Your opponent has finished moves C2R2, C3R1. You have finished moves C2R3. The legal moves are C1R1, C1R2, ... **Step Prompt:** Please choose one move from legal moves to set up an advantage.

......

Environment

CoT Agent wins!

Connect-4

Environment

System Prompt: You are a powerful gaming agent who can make proper decisions to beat the user in gaming tasks. You are playing Connect-4 with me

Environment

Observation Prompt: You are the first to go. The legal moves are C1R6, C2R6, C3R6, C4R6, C5R6, C6R6, C7R6. **Step Prompt:** Please first think about your situation, then choose one move from legal moves to set up an advantage.

CoT Agent

Thought: By starting in the center, I should create opportunities to build sequences in multiple directions (vertically, horizontally, and diagonally). This central position gives me more flexibility in future moves and can potentially limit your options **Move:** C4R6

Environment

Observation Prompt: Your opponent has finished moves C4R6. The legal moves are: C1R6, C2R6, C3R6, C4R5, C5R6, C6R6, C7R6. **Step Prompt:** Please choose one move from legal moves to set up an advantage.

C3R6

Prompt Agent

Environment

Observation Prompt: Currently, your opponent has finished moves C3R6. The legal moves are C1R6, C2R6, C3R5, ... **Step Prompt:** Please first think about your situation, then choose one move from legal moves to set up an advantage.

CoT Agent

Thought: I'll place my disc in C5R6. This positions my disc to the right of my initial disc in the center column, creating a potential line horizontally and keeping options open for vertical and diagonal connections as well. **Move:** C5R6

Environment

Observation Prompt: Your opponent has finished moves C4R6, C5R6. You have finished moves C3R6. The legal moves are C1R6, C2R6, ... **Step Prompt:** Please choose one move from legal moves to set up an advantage.

......

Environment

CoT Agent wins!

Texas Hold'em Poker

Environment

System Prompt: You are playing Texas Holdem Poker with the other 1 players. The aim of each player in Texas Hold'em poker is to win chips or money from other players by either having the best hand at showdown or by convincing other players to fold their hands. The small blind bet of this game is 1 and the big blind bet of this game is 2.

Environment

Observation Prompt: Here are the situations you are facing: You are in the PREFLOP stage. You are in the SMALL BLIND position. The current round pot is 3 and the community pot is 0. Your card is 4H and AC. Your remaining chip is 199. Currently, your legal actions are FOLD, CHECK, RAISE_3BB, ... **Step Prompt:** It is your turn to take your action. Given the current situation, first make a plan about how to make your decision regarding the current situation, then select one action from the action space to beat opponents

CoT Agent

Thought: I would select RAISE_3BB. This move can help to take control of the pot early and gives a chance to gauge the reactions of the opponents. **Move:** RAISE_3BB

Environment

Observation Prompt: Here are the situations you are facing: You are in the PREFLOP stage. You are in the BIG BLIND position. The current round pot is 11 and the community pot is 0. Your card is KD and AS. Your remaining chip is 198. Currently, your legal actions are FOLD, CHECK, RAISE_3BB, ... In this round, prior players have made the following actions: Player 1 (small blind): RAISE_3BB **Step Prompt:** Please select one action from the action space to beat opponents.

CALL

Prompt Agent

......

Environment

Prompt Agent wins!

Figure 1: LLMs online racing in multi-turn strategic scenarios.

2022b). (Driess et al., 2023) proposes a multimodal language model for embodied reasoning tasks, visual-language tasks, and language tasks. Beyond that, (Liu et al., 2023a) translates such intermediate steps into executable programming languages to conduct classical planning algorithms. Also, Autonomous Agents have driven zero/few-shot LLMs to achieve complex reasoning and planning tasks through prompt engineering (Liu et al., 2023b; Xi et al., 2023; Xiang et al., 2023). (Yao et al., 2022b; Shinn et al., 2023) endow agents with the capability to engage in introspection regarding the feedback provided by LLMs.

3 Preliminary Analysis

We first investigate the strategic reasoning capabilities of LLMs through online competitions. Figure 1 presents the procedures of online LLMs racing and the demonstration of each environment.

3.1 Preliminary

We present competitions among two strategic games and seven agents in this section:

Tic-Tac-Toe: We utilize the version of 3×3 grid with the winning length as 3. There are two agents in each match and each agent is prompted to select actions when giving the current board state (e.g., legal moves and the opponent's moves). We utilize the symbol <CxRy> to represent each move on the Tic-Tac-Toe board where x and y represent the column index and row index respectively. Symbolic representations have been widely adopted by other board games, e.g., FEN (Wikipedia, 2023b) and Algebraic notation (Wikipedia, 2023a). All the prompt templates can be found in Appendix A.1. Since the first-go player obtains significant advantages in this game, we execute 200 matches with each agent going first for 100 matches. We use the average *win ratio*, i.e., $\frac{\text{win match}}{\text{total match}}$ and *loss ratio*, i.e., $\frac{\text{loss match}}{\text{total match}}$, to evaluate performance.

Texas Hold'em Poker[4]: Each agent is assigned $200 chips initially. The agent is prompted to select an action from the action set: `FOLD`, `CHECK`, `CALL`, `RAISE_3BB`, `RAISE_HALF_POT`, `RAISE_POT`, `RAISE_2POT`, `ALL_IN`, `SMALL_BLIND`, `BIG_BLIND`.

[4] https://github.com/dickreuter/neuron_poker

Agent v.s. Agent	Random	MinMax	Prompt	CoT	CoT-SC	ToT	ReAct	Avg. Win Ratio (↑)
Random	-	4.50%	40.00%	36.50%	37.50%	33.50%	37.50%	31.58%
MinMax	86.00%	-	92.00%	83.50%	85.00%	81.50%	76.00%	84.00%
Prompt	54.50%	5.00%	-	24.00%	20.00%	24.00%	24.50%	25.33%
CoT	54.00%	4.50%	43.50%	-	36.00%	42.50%	39.00%	36.58%
CoT-SC	52.50%	7.00%	38.00%	36.00%	-	31.50%	36.00%	33.50%
ToT	55.00%	8.00%	52.00%	30.00%	29.00%	-	48.00%	**37.00%**
ReAct	54.00%	6.00%	38.50%	39.00%	33.50%	38.50%	-	34.92%
Avg. Loss Ratio (↓)	59.33%	5.83%	50.67%	41.50%	**40.17%**	41.92%	43.50%	-

Table 1: Benchmarking reasoning agents in the Tic-Tac-Toe environment. Each cell **(Row, Col)** means the **win ratio** of the **Row agent** when against the **Col agent**. Note that the game result can be a draw, so the sum of the win ratios of a pair of two agents is not 100%. It is shown that only ToT and CoT outperform the Random agent with moderate margins and all other agents are just slightly better or even worse than Random.

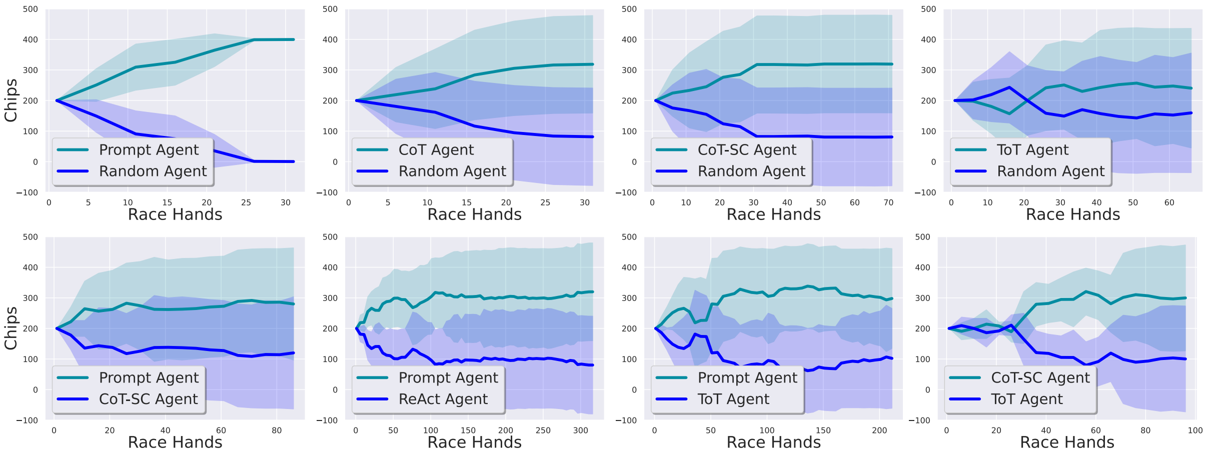


Figure 2: Remaining chips of reasoning agents at each hand in the Texas Hold'em Poker environment. Standard deviations over 20 trials are shown as the shadowed areas. Agents with more remaining chips at last mean better performance. Among these agents, the naive Prompt agent works better than other methods.

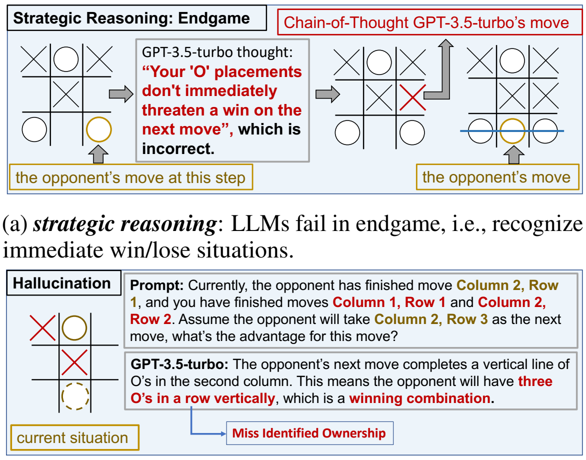
The utilized prompts can be found in Appendix A.2. Detailed explanations of these actions can be found in Appendix B. There are dozens of hands within each match. We utilize the hand win ratio, e.g., $\frac{\text{win hands}}{\text{total hands}}$ to evaluate performance.

Reasong Agents: We consider 7 agents (5 LLMs-powered agents and 2 baseline agents): ❶ Random: the agent that randomly selects action at each step; ❷ MinMax: the agent that selects action based on conventional min-max gaming strategy (only compatible with Tic-Tac-Toe); ❸ Prompt: the agent that directly prompts LLMs to return answers; ❹ Chain-of-Thought (CoT): the agent that reasons through thinking step by step; ❺ Self-Consistent CoT (CoT-SC): the agent that utilizes multiple step-by-step-thinking trajectories during reasoning; ❻ Tree-of-Thought (ToT): the agent that augmented with exploration and deliberate decision-making, i.e., self-evaluation. ❼ ReAct: the agent that follows reasoning-before-acting policy. All the agents are driven by ChatGPT (GPT-3.5-turbo-0613).

It is worth noting that some agents are not originally designed for strategic gaming tasks. In Appendix C, we provide details on how we make them applicable.

3.2 Evaluation Results

In Table 1, we report the average win ratios and loss ratios in the Tic-Tac-Toe environment. The optimization-based MinMax agent significantly outperforms all other methods, which is expected as we just use it as a reference baseline. Surprisingly, we found that most advanced reasoning agents work only slightly better than the Random agent. The Prompt agent works even worse than the Random agent. Among these methods, ToT


Strategic Reasoning: Endgame
Chain-of-Thought GPT-3.5-turbo's move
the opponent's move at this step
the opponent's move

GPT-3.5-turbo thought: **"Your 'O' placements don't immediately threaten a win on the next move", which is incorrect.**

(a) ***strategic reasoning***: LLMs fail in endgame, i.e., recognize immediate win/lose situations.

Hallucination
current situation
Miss Identified Ownership

Prompt: Currently, the opponent has finished move **Column 2, Row 1**, and you have finished moves **Column 1, Row 1** and **Column 2, Row 2**. Assume the opponent will take **Column 2, Row 3** as the next move, what's the advantage for this move?

GPT-3.5-turbo: The opponent's next move completes a vertical line of O's in the second column. This means the opponent will have **three O's in a row vertically**, which is a **winning combination**.

(b) ***hallucination***: LLMs failed to recognize the identity of pieces.

Figure 3: Some representative error patterns of CoT GPT-3.5-turbo in Tic-Tac-Toe.

achieves the highest average win ratio (37%) and CoT-SC achieves the lowest loss ratio (40.17%). In Figure 2, we present the performance of reasoning agents when playing Texas Hold'em Poker. We found that the Prompt agent works better than other agents. Advanced reasoning agents work slightly better than the Random agent.

3.3 Analytical Insights

We summarize the following insights according to the experimental results:

Serious Hallucination and Reasoning Errors. We found that LLMs suffer from serious hallucinations and reasoning errors. Figure 3 provides demonstrations of how LLMs failed in perceiving board states and endgames.

Advanced Reasoning Not Always Help. Although advanced reasoning agents (e.g., CoT, CoT-SC, ReAct, ToT) all work better than directly prompt LLMs in Tic-Tac-Toe, this trend reverses in Texas Hold'em Poker, where directly prompted LLMs actually perform better than all the advanced reasoning agents. One potential reason is the nature of incomplete games, where only partial information is available, hindering effective reasoning by LLMs. Additionally, Texas Hold'em Poker demands strong Theory-of-Mind (ToM) skills like bluffing, which are challenging for LLMs (Stepputtis et al., 2023).

4 In-Depth Strategic Reasoning Analysis

The limited success of state-of-the-art LLMs when against random agents as opponents raises a critical question: *What specific vulnerabilities and limitations are being exposed?*

Overall Statistics	**Number**
Number of questions	2,700
- Yes/No questions	2,400 (89%)
- Other questions	300 (11%)
Maximum question length	18
Average question length	11.19
Number of hallucination error types	5
- Spatial, Pattern, Memory, Legality, Counting	-
Number of strategic reasoning error types	4
- Priority, Endgame, Fork, Blocking	-
Number of questions for each error type	300
Maximum number of turns in questions	19
Minimal number of turns in questions	2
Average number of turns in questions	6.6

Table 2: Statistics of the offline configurations.

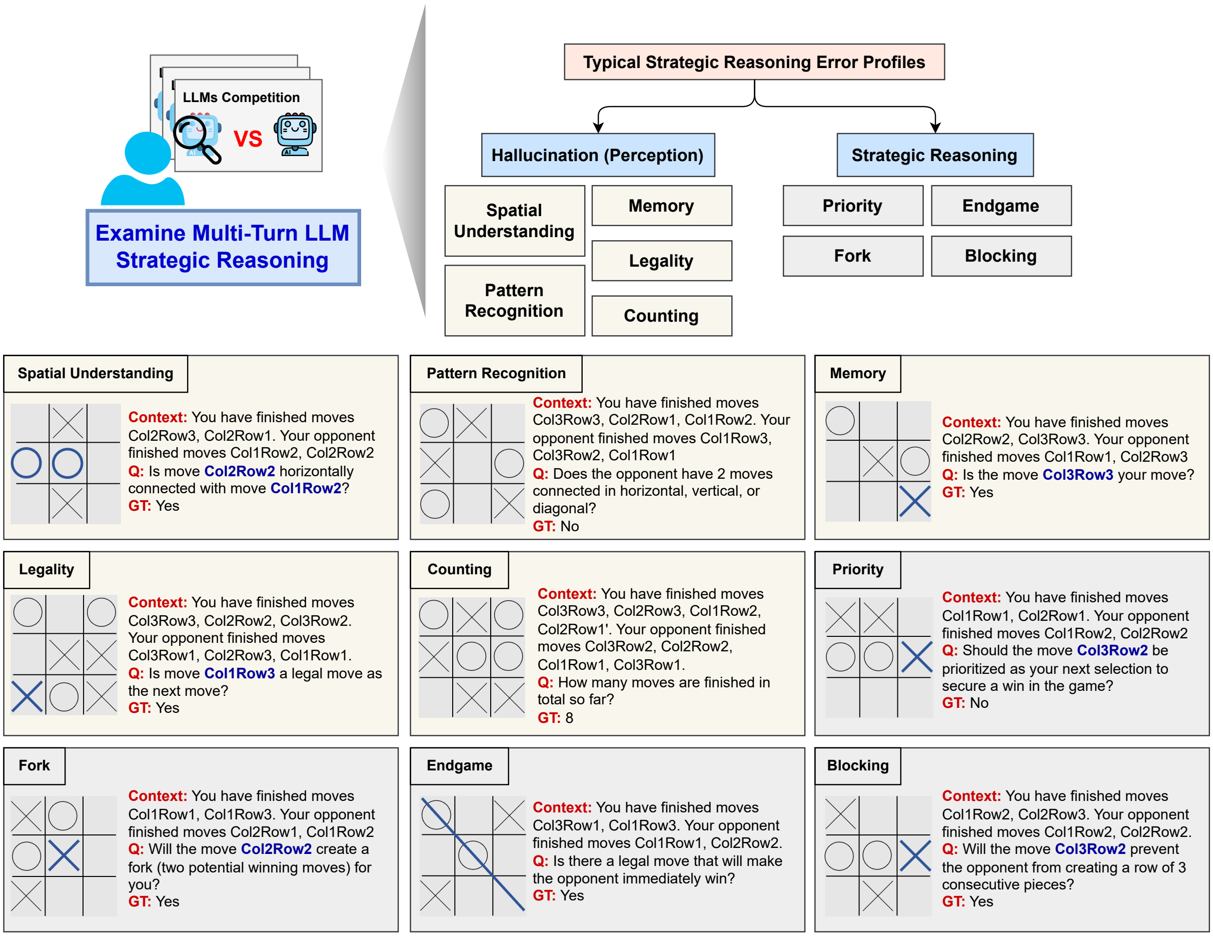
4.1 Preliminary

To answer this question, we provide targeted questions and verified answers for detailed offline demographic analysis. As a demonstration, we characterize LLMs' strategic behaviors over board games (e.g., Tic-Tac-Toe and Connect-4). We first examine LLMs' behaviors from online competitions obtained in Section 3 and summarize two main error categories: *hallucination* and *strategic reasoning*, that result in loss.

Hallucination. We probe hallucinations by examining whether LLMs are capable of ❶ Spatial Understanding, i.e., spatial relationship given any two pieces; ❷ Pattern Recognition, i.e., discovering consecutively connected pieces; ❸ Counting, i.e., counting finished pieces; ❹ Memory, i.e., identifying the ownership of each piece; ❺ Legality, i.e., recognizing legal and illegal moves.

Strategic Reasoning. We probe four common abilities in general board games: ❶ Action Priority, i.e., winning moves should be prioritized; ❷ Endgame, i.e., recognizing immediate win/loss situations; ❸ Blocking, i.e., blocking the winning of the opponent; ❹ Fork, i.e., constructing moves that lead to two potential winning moves.

We provide demonstrations for each type of error in Figure 4. It is worth noting that these errors, e.g., fork, blocking, endgame, are also prevalent in general board games (Dixit and Nalebuff, 2010). Although we only provide demonstrations over Tic-Tac-Toe and Connect-4, this can be easily generalized to other board games such as Chess and Go.



Spatial Understanding

Context: You have finished moves Col2Row3, Col2Row1. Your opponent finished moves Col1Row2, Col2Row2
Q: Is move **Col2Row2** horizontally connected with move **Col1Row2**?
GT: Yes

Pattern Recognition

Context: You have finished moves Col3Row3, Col2Row1, Col1Row2. Your opponent finished moves Col1Row3, Col3Row2, Col1Row1
Q: Does the opponent have 2 moves connected in horizontal, vertical, or diagonal?
GT: No

Memory

Context: You have finished moves Col2Row2, Col3Row3. Your opponent finished moves Col1Row1, Col2Row3
Q: Is the move **Col3Row3** your move?
GT: Yes

Legality

Context: You have finished moves Col3Row3, Col2Row2, Col3Row2. Your opponent finished moves Col3Row1, Col2Row3, Col1Row1.
Q: Is move **Col1Row3** a legal move as the next move?
GT: Yes

Counting

Context: You have finished moves Col3Row3, Col2Row3, Col1Row2, Col2Row1'. Your opponent finished moves Col3Row2, Col2Row2, Col1Row1, Col3Row1.
Q: How many moves are finished in total so far?
GT: 8

Priority

Context: You have finished moves Col1Row1, Col2Row1. Your opponent finished moves Col1Row2, Col2Row2
Q: Should the move **Col3Row2** be prioritized as your next selection to secure a win in the game?
GT: No

Fork

Context: You have finished moves Col1Row1, Col1Row3. Your opponent finished moves Col2Row1, Col1Row2
Q: Will the move **Col2Row2** create a fork (two potential winning moves) for you?
GT: Yes

Endgame

Context: You have finished moves Col3Row1, Col1Row3. Your opponent finished moves Col1Row1, Col2Row2.
Q: Is there a legal move that will make the opponent immediately win?
GT: Yes

Blocking

Context: You have finished moves Col1Row2, Col2Row3. Your opponent finished moves Col1Row2, Col2Row2.
Q: Will the move **Col3Row2** prevent the opponent from creating a row of 3 consecutive pieces?
GT: Yes

Figure 4: Error profiles in the offline dataset.

4.2 Offline Dataset Generation

Utilizing structured symbols for each move, such as <CxRy>, we generate unlimited legal board states with adjustable complexities. For dataset creation, we crafted prompt templates for each error type and traversed all occupied/legal moves to populate these templates. We also implement verifiers for each error type to establish ground truth. We then sampled balanced questions based on complexity and labels, e.g., Yes and No. The statistics of the offline probing dataset for Tic-Tac-Toe and Connect-4 are detailed in Table 2.

4.3 Evaluation and Error Analysis

We evaluate strategic reasoning for both commercial LLMs, e.g., GPT-3.5-turbo and GPT-4, and open-source LLMs, e.g., Llama-2-chat (Touvron et al., 2023), Mistral-Instruct (Jiang et al., 2023). Results are summarized in Table 3.

For hallucinations, we show that GPT-4 with CoT reasoning achieves significant accuracy (90.7%), suggesting that LLMs are capable of effectively perceiving board states through symbolic representations. However, other LLMs demonstrated significant hallucination issues, indicating challenges in understanding board states. For strategic reasoning, we show that even the most state-of-the-art GPT-4 can only achieve 54.6% accuracy on average, which is only slightly better than random guessing. It suggests the vulnerabilities and limitations in strategic reasoning for LLMs. The CoT reasoning only marginally improves performance (+0.8%) in this scenario.

4.4 States Complexity Effects

As competitions progress and the complexity of the board state increases significantly, we quantify the correlation between this complexity and model performance. In Figure 5, we demonstrate how model performances are impacted in scenarios where complexity is directly influenced by the number of completed turns, including Counting, Pattern, Priority, Endgame, Blocking, and Fork. We normalize the complexity derived from the number of turns to a range of (0,1) and calculate the accuracy at each specific number of turns. It is shown that as the

		Hallucination (Perception)						Strategic Reasoning				
Model and Reasoning	**All Avg.**	spatial	pattern	counting	memory	legality	*avg.*	priority	endgame	blocking	fork	*avg.*
Random	0.444	0.500	0.500	0.000	0.500	0.500	0.400	0.500	0.500	0.500	0.500	0.500
GPT-4	0.665	0.843	0.597	0.746	0.777	**0.837**	0.760	**0.567**	0.560	0.523	**0.533**	0.546
GPT-4 w/ CoT	**0.750**	**0.947**	**0.817**	**0.997**	**0.940**	0.833	**0.907**	0.540	**0.597**	**0.560**	0.518	**0.554**
GPT-3.5-turbo	0.554	0.503	0.537	0.707	0.643	0.627	0.603	0.503	0.475	0.498	0.497	0.493
GPT-3.5-turbo w/ CoT	0.641	0.763	0.577	0.903	0.766	0.669	0.736	0.505	0.519	0.557	0.505	0.522
Mistral-7B-Instruct-v0.1	0.494	0.545	0.520	0.225	0.515	0.524	0.466	0.545	0.551	0.543	0.476	0.529
Mistral-7B-Instruct-v0.1 w/ CoT	0.486	0.523	0.527	0.263	0.604	0.477	0.479	0.482	0.461	0.530	0.505	0.495
Llama-2-70b-chat	0.476	0.483	0.493	0.120	0.590	0.553	0.448	0.517	0.503	0.520	0.500	0.510
Llama-2-70b-chat w/ CoT	0.568	0.537	0.530	0.763	0.573	0.613	0.603	0.513	0.530	0.533	0.520	0.524
CodeLlama-34b-Instruct	0.477	0.547	0.560	0.070	0.550	0.540	0.453	0.550	0.482	0.513	0.477	0.505
CodeLlama-34b-Instruct w/ CoT	0.559	0.667	0.535	0.593	0.638	0.577	0.602	0.512	0.530	0.490	0.493	0.506

Table 3: Evaluation results on the offline datasets. State-of-the-art LLMs (e.g., GPT-4) with CoT reasoning are capable of perceiving board states (90.7% accuracy in hallucination scenarios). However, it only works slightly better than random guesses in strategic thinking scenarios, even with the help of CoT reasoning.

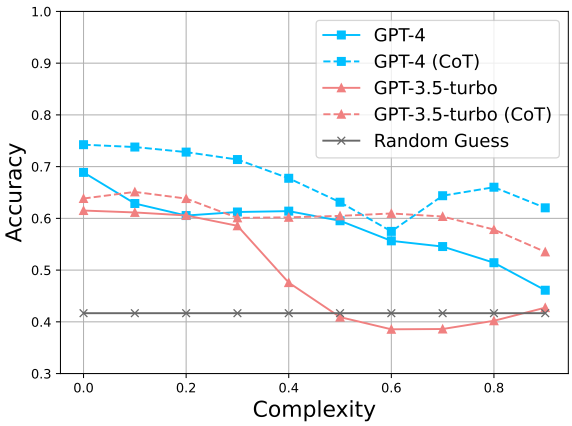


Figure 5: Correlations between board complexities and model performances. It indicates that complex board situations result in a significant performance drop for state-of-the-art LLMs.

board becomes more complex, there is a significant drop in the strategic reasoning performances, e.g., the accuracy of GPT-4 drops from 68.8% to 46.1%.

4.5 Emergent Abilities in Strategic Reasoning

Following emergent abilities of LLMs (Wei et al., 2022a), we study how LLM parameter sizes affect strategic reasoning. In Figure 6, we compare the popular Llama models at 7b, 13b, 34b (CodeLlama), and 70b parameter sizes. For hallucination, increasing parameter sizes significantly improves accuracy from 43.4% to 60.3%, suggesting emergent abilities in strategic linguistic understanding.

However, there is no such trend in that strategic thinking evaluation. We show that Llama-2-7b-chat has similar performances as Llama-2-70b-chat model, i.e. 50.6% to 52.4%. This raises new challenges regarding how to equip LLMs with the capability for effective strategic reasoning when simply increasing parameter size proves ineffective.

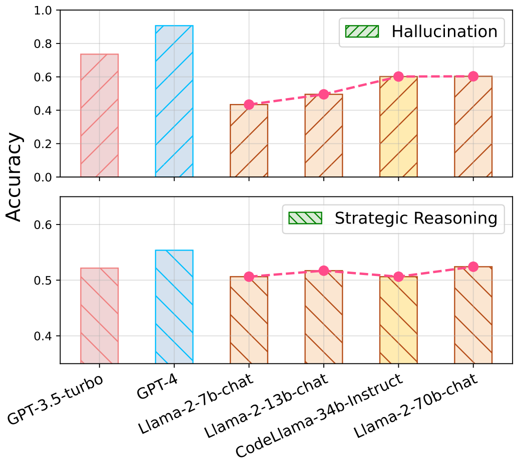


Figure 6: The emergent abilities in strategic reasoning. Increasing model parameter sizes effectively mitigates hallucination and perception errors, while it does not yield similar improvements in strategy.

5 ReTA: Recursively Thinking Ahead

In this section, we propose ReTA, a new LLM agent for improved strategic reasoning.

5.1 Preliminary

We formulate the gaming process as a discrete decision-making process among *actors*, under the interaction with the gaming environment. We define *actors* as LLMs-powered agents that take natural language (or prompts) as inputs and generate corresponding actions as outputs. Without loss of generality, we assume two actors are participating in this gaming process. Considering the t-th step of this process, we denote by $s_t \in \mathcal{S}$ the state that the two actors observed, and $a_t, \hat{a}_t \in \mathcal{A}$ the actions sampled by the two actors, where $\mathcal{S}$ is the infinite

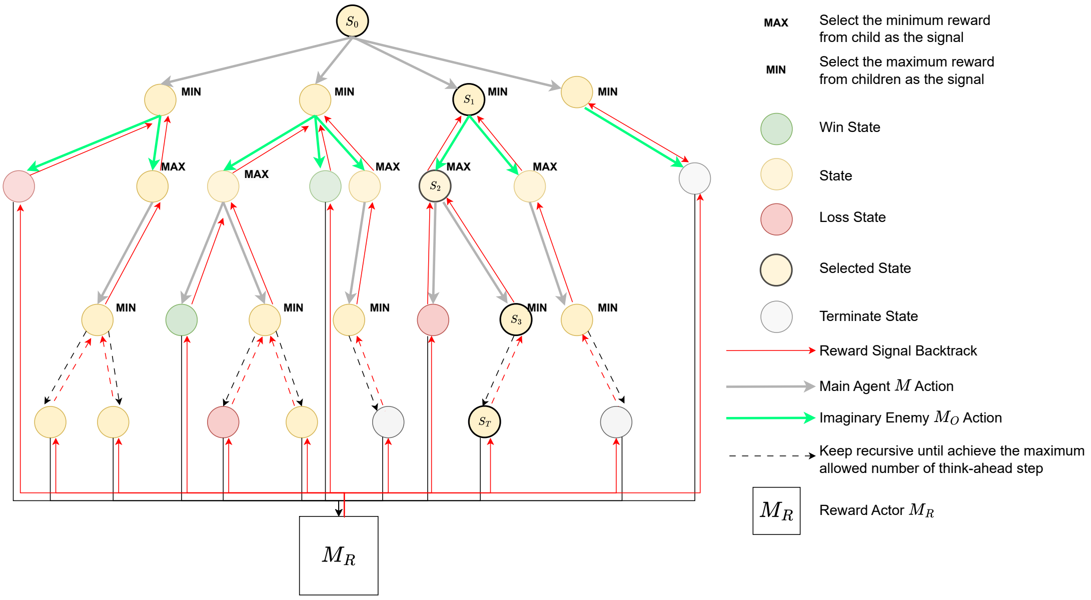

S_0
MIN
MAX
S_1
S_2
S_3
S_T
M_R
MAX Select the minimum reward from child as the signal
MIN Select the maximum reward from children as the signal
Win State
State
Loss State
Selected State
Terminate State
Reward Signal Backtrack
Main Agent M Action
Imaginary Enemy M_O Action
Keep recursive until achieve the maximum allowed number of think-ahead step
M_R Reward Actor M_R

Figure 7: The overall architecture of ReTA.

state space and $\mathcal{A}$ is the infinite action space. The state transition from s_t to s_{t+1} can be formulated as $s_{t+1} = \mathcal{T}(s_t, a_t)$ where $a_t \sim p(a_t|s_t, x)$, $p(a_t)$ refers to the generative distribution of the backbone LLMs, and x is the instruction (or prompt). Then, the two-agent gaming process can be represented as the sequence $(s_0, a_0, s_1, \hat{a}_1, s_2, \cdots, s_N)$, where s_0 is the initial state and s_N is a terminal state, e.g., a win/draw/loss situation. In this process, the two actors will alternatively sample actions to achieve new states, aiming to maximize their winning rates.

5.2 Recursively Thinking Ahead

Foresight is one of the significant differentiating factors between top-tier players and average players, especially in strategy games like board and card games. It requires the players to calculate moves ahead, visualize the board's possible states, and evaluate the consequences of various move sequences. To simulate this process, we formulate ReTA as the ensemble of modules, utilizing multiple individual actors:

- **Main Actor** M: interacting with the environment and generating the next action, i.e., $a_t \sim P_M(a_t|s_t, x)$ where s_t is the current state and x is the prompt.
- **Reward Actor** M_R: working as a reward function to calculate the reward regarding different states, i.e., $r \sim P_R(r|s_t, x)$.
- **Anticipation Actor** M_O: an imaginary opponent, predicting action $a_{o,t}$ to beat M at state s_t, i.e., $\hat{a}_{o,t} \sim P_O(\hat{a}_{o,t}|s_t, x)$.

Here P_M, P_R and P_O are the generative distributions of the backbone LLMs: M, M_R and M_O.

It is worth noting that some search-based gaming frameworks, such as the conventional minimax gaming (Lan et al., 2020), are standard think-ahead frameworks. In these frameworks, both actors will try to minimize the possible loss for a worst-case scenario (maximum loss) when making a move. Our recursively thinking-ahead mechanism follows this simple and classic protocol. Specifically, at the beginning of each gaming step t, we first sample n desired actions $\mathcal{A}_t = \{\tilde{a}_t^1, \tilde{a}_t^2, \cdots, \tilde{a}_t^n\} \sim P_M(\tilde{a}_t|s_t, x)$ from M as the candidacy actions, given current state s_t. Then, we formulate the think-ahead process as the *pseudo-gaming* with M_O, as the following sequence:

$$(s_t, \tilde{a}_t, s_{t+1}, \hat{a}_{o,t+1}, s_{t+2}, \tilde{a}_{t+2}, \cdots, s_T), \quad (1)$$

where $\tilde{a}_t \in \mathcal{A}_t$ is a candidacy action at pseudo-gaming step t, $\hat{a}_{o,t+1} \sim P_O(\hat{a}_{o,t}|s_{t+1}, x)$ is the sampled action from imaginary opponent M_O, and s_T is a terminal state, e.g. achieves win/draw/lose situation or achieves state s_{t+k} where k is the maximum allowed number of think-ahead steps. Once the terminal state is achieved in pseudo-gaming, the reward agent M_R will perform situation assessment by answering an advantage score, r_T, to describe how many advantages the actor M has at state s_T: $r_{s_T} \sim P_o(r_T|s_T, T, x)$, $(0 \leq r \leq 1)$. Theoretically, if we traverse all the possible combinations of candidacy actions and always take k

Setting	ReTA Win Ratio	Others Win Ratio
ReTA Agent vs. ToT Agent		
ReTA ($k = 2, n = 2$)	37%	59%
ReTA ($k = 2, n = 4$)	48%	37%
+ majority vote ($k_{mv} = 3$)	**62%**	35%
+ P-UQ ($k_{pert} = 2$)	60%	34%
+ majority vote + P-UQ	61%	**31%**
GPT-3.5-turbo as LLMs: ReTA Agent vs. Other Agents		
ReTA v.s. ToT	**61%** (+30%)	31%
ReTA v.s. CoT-SC	**52%** (+17%)	35%
ReTA v.s. ReAct	**50%** (+10%)	40%
ReTA v.s. Prompt	**60%** (+26%)	34%
ReTA v.s. CoT	**59%** (+29%)	30%
Llama-2-13b-chat as LLMs: ReTA Agent vs. Other Agents		
ReTA v.s. ToT	**51%** (+11%)	40%
ReTA v.s. CoT	**55%** (+11%)	44%
ReTA v.s. ReAct	**56%** (+13%)	43%
ReTA v.s. Prompt	**62%** (+26%)	36%

Table 4: Ablation study and evaluations of ReTA in the Tic-Tac-Toe environment.

steps to achieve terminal states, there will be a k-layer decision-making tree constructed with n^k leave nodes, which indicates there will be at most n^k terminal states and advantage scores in total.

Once we finish traversing this decision tree and obtain advance scores for each terminal state, we will perform reward signal backtracking from state s_T to s_t and select action a_t, in a minimax manner:

$$\max_{a_t \in \mathcal{A}} \min_{\hat{a}_{t+1} \in \mathcal{A}} (r_{s_t} P_O(\hat{a}_{t+1}|s_{t+1}) P_M(a_t|s_t)). \quad (2)$$

With this minimax reward backtracking, we assume that the opponent will always choose the “worst case" during the gaming, which makes our agent more robust to the opponents. Once the traceback happens to the root of the tree, there will be a reward signal for each candidacy action in $\mathcal{A}_t$. Then, we simply select the action with the highest rewards as the next action.

5.3 Hallucination and Factual Errors

As we mentioned in Section 3.3, LLMs suffer from serious hallucinations and factual errors. Even in the simplest 3×3 Tic-Tac-Toe situation, LLMs struggle to read the correct spatial information and recognize immediate win positions. To mitigate this issue, we adopt two strategies during the gaming process: *majority vote* (Wang et al., 2022a) and *perturbation-based uncertainty estimation* (P-UE) (Manakul et al., 2023).

For the majority vote, we simply sample k_{mv} generations as options and let LLMs select the high-frequency option or the mean value if it is a numerical result. For P-UE, we first prompt LLMs to perturb the target questions in k_{pert} times while keeping the semantics unchanged, then we sample generations based on both original question and perturbed questions and apply a majority vote over these generations to select the next action.

Setting	ReTA Hand Win Ratio	Others Hand Win Ratio
ReTA v.s. Prompt	**53.8%** (+7.6%)	46.2%
ReTA v.s. CoT-SC	**63.2%** (+26.4%)	36.8%
ReTA v.s. ToT	**72.1%** (+44.2%)	27.9%
ReTA v.s. ReAct	**78.0%** (+56.0%)	22.0%

Table 5: Evaluations of ReTA in Texas Hold’em Poker.

Setting	ReTA Win Ratio	Others Win Ratio
ReTA v.s. ReAct	**55%** (+10%)	45%
ReTA v.s. ToT	**60%** (+20%)	40%

Table 6: Evaluation of ReTA in Connect-4.

To control the cost of tokens, we only apply majority vote and P-UQ to the situation assessment procedures, i.e., generating advantage scores with reward actor M_R.

5.4 Expirical Results

We utilize the same settings as in Section 3. For Tic-Tac-Toe, we execute 100 matches with each agent going first for 50 matches. For Connect-4 and Texas Hold’em Poker, we execute 20 matches.

In Table 4, we conduct comprehensive ablation studies and evaluations of ReTA over Tic-Tac-Toe. We take ToT as the opponent of ReTA because ToT achieves the best performance among all reasoning agents in Section 3. It is shown that the proposed ReTA agent significantly boosts the strategic reasoning of LLMs. Further experiments carried out on the open-source Llama-2-13b-chat also show distinct advantages for ReTA, suggesting the strong transferability regarding different LLM backbones. In Tables 5 and 6, the empirical results obtained over Texas Hold’em Poker and Connect-4 present that ReTA could be generalized to other gaming scenarios.

6 Conclusion

In this paper, we propose to evaluate the multi-turn strategic reasoning capabilities of LLMs. We provide online agent competition and offline reasoning probing, offering an in-depth examination of strategic behaviors. Our work introduces a new dimension to LLMs evaluation, and we hope it will inspire further research into the multi-turn strategic reasoning of LLMs.

7 Ethical Considerations

Prompting and Evaluating LLMs to be strategic reasoning agents increases real-world autonomy for LLMs and brings a lot of potential applications in the real world. As a result, AI-driven decision-making may potentially reduce the role of human skill and creativity. It also raises the question of who should be responsible for the decisions of LLMs. Besides, ensuring fairness and avoiding biases in the model's strategy is essential, as biases can influence game outcomes and player experiences. It is also important to consider the impact of advanced strategic reasoning on the integrity of games, particularly in competitive settings, to maintain a level playing field for all players.

8 Limitations

Although we consider both complete- and incomplete-gaming tasks, there are still other game forms not covered. We will take expanding more strategic games as the future work. Also, even though the proposed `ReTA` outperforms existing reasoning agents, it is still significantly worse than optimization-based solvers, such as `MinMax` agents. Strategic reasoning requires strong instruction following capabilities. Currently, only commercial LLMs (e.g., ChatGPT and GPT-4) are capable of following complex instructions, while other open-source LLMs (e.g., Llama-2-chat) are still undesirable to be the backbone of strategic reasoning agents.

Acknowledgements

This work was performed under the auspices of the U.S. Department of Energy by the Lawrence Livermore National Laboratory under Contract No. DE- AC52-07NA27344 and was supported by the LLNL LDRD Program under Project No. 23-ERD-030. This work was partially supported by NSF No. 2319242.

References

Ning Bian, Peilin Liu, Xianpei Han, Hongyu Lin, Yaojie Lu, Ben He, and Le Sun. 2023. A drop of ink makes a million think: The spread of false information in large language models.

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

Xiang Deng, Yu Gu, Boyuan Zheng, Shijie Chen, Samuel Stevens, Boshi Wang, Huan Sun, and Yu Su. 2023. Mind2web: Towards a generalist agent for the web. *ArXiv*, abs/2306.06070.

Avinash Dixit and Barry Nalebuff. 2010. The art of strategy: A game theorist's guide to success in business and life.

Danny Driess, Fei Xia, Mehdi SM Sajjadi, Corey Lynch, Aakanksha Chowdhery, Brian Ichter, Ayzaan Wahid, Jonathan Tompson, Quan Vuong, Tianhe Yu, et al. 2023. Palm-e: An embodied multimodal language model. *arXiv preprint arXiv:2303.03378*.

Jinhao Duan, Hao Cheng, Shiqi Wang, Chenan Wang, Alex Zavalny, Renjing Xu, Bhavya Kailkhura, and Kaidi Xu. 2023. Shifting attention to relevance: Towards the uncertainty estimation of large language models. *arXiv preprint arXiv:2307.01379*.

Jinhao Duan, Renming Zhang, James Diffenderfer, Bhavya Kailkhura, Lichao Sun, Elias Stengel-Eskin, Mohit Bansal, Tianlong Chen, and Kaidi Xu. 2024. Gtbench: Uncovering the strategic reasoning limitations of llms via game-theoretic evaluations. *arXiv preprint arXiv:2402.12348*.

Zorik Gekhman, Jonathan Herzig, Roee Aharoni, Chen Elkind, and Idan Szpektor. 2023. Trueteacher: Learning factual consistency evaluation with large language models.

Wenlong Huang, Pieter Abbeel, Deepak Pathak, and Igor Mordatch. 2022a. Language models as zero-shot planners: Extracting actionable knowledge for embodied agents. In *International Conference on Machine Learning*, pages 9118–9147. PMLR.

Wenlong Huang, Fei Xia, Ted Xiao, Harris Chan, Jacky Liang, Pete Florence, Andy Zeng, Jonathan Tompson, Igor Mordatch, Yevgen Chebotar, et al. 2022b. Inner monologue: Embodied reasoning through planning with language models. *arXiv preprint arXiv:2207.05608*.

Ziwei Ji, Nayeon Lee, Rita Frieske, Tiezheng Yu, Dan Su, Yan Xu, Etsuko Ishii, Ye Jin Bang, Andrea Madotto, and Pascale Fung. 2023. Survey of hallucination in natural language generation. *ACM Computing Surveys*, 55(12):1–38.

Albert Q Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, et al. 2023. Mistral 7b. *arXiv preprint arXiv:2310.06825*.

Marzena Karpinska and Mohit Iyyer. 2023. Large language models effectively leverage document-level context for literary translation, but critical errors persist.

Qingfeng Lan, Yangchen Pan, Alona Fyshe, and Martha White. 2020. Maxmin q-learning: Controlling the estimation bias of q-learning. *arXiv preprint arXiv:2002.06487*.

Bo Liu, Yuqian Jiang, Xiaohan Zhang, Qiang Liu, Shiqi Zhang, Joydeep Biswas, and Peter Stone. 2023a. Llm+ p: Empowering large language models with optimal planning proficiency. *arXiv preprint arXiv:2304.11477*.

Ruibo Liu, Ruixin Yang, Chenyan Jia, Ge Zhang, Denny Zhou, Andrew M Dai, Diyi Yang, and Soroush Vosoughi. 2023b. Training socially aligned language models in simulated human society. *arXiv preprint arXiv:2305.16960*.

Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Yuxian Gu, Hangliang Ding, Kai Men, Kejuan Yang, Shudan Zhang, Xiang Deng, Aohan Zeng, Zhengxiao Du, Chenhui Zhang, Shengqi Shen, Tianjun Zhang, Yu Su, Huan Sun, Minlie Huang, Yuxiao Dong, and Jie Tang. 2023c. Agentbench: Evaluating llms as agents. *ArXiv*, abs/2308.03688.

Potsawee Manakul, Adian Liusie, and Mark John Francis Gales. 2023. Selfcheckgpt: Zero-resource black-box hallucination detection for generative large language models. *ArXiv*, abs/2303.08896.

Xavier Puig, Kevin Ra, Marko Boben, Jiaman Li, Tingwu Wang, Sanja Fidler, and Antonio Torralba. 2018. Virtualhome: Simulating household activities via programs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8494–8502.

Noah Shinn, Federico Cassano, Beck Labash, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. 2023. Reflexion: Language agents with verbal reinforcement learning.

Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Côté, Yonatan Bisk, Adam Trischler, and Matthew Hausknecht. 2020. Alfworld: Aligning text and embodied environments for interactive learning. *arXiv preprint arXiv:2010.03768*.

David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, L. Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Vedavyas Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy P. Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. 2016. Mastering the game of go with deep neural networks and tree search. *Nature*, 529:484–489.

David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, L. Sifre, Dharshan Kumaran, Thore Graepel, Timothy P. Lillicrap, Karen Simonyan, and Demis Hassabis. 2017. Mastering chess and shogi by self-play with a general reinforcement learning algorithm. *ArXiv*, abs/1712.01815.

Simon Stepputtis, Joseph Campbell, Yaqi Xie, Zhengyang Qi, Wenxin Sharon Zhang, Ruiyi Wang, Sanketh Rangreji, Michael Lewis, and Katia Sycara. 2023. Long-horizon dialogue understanding for role identification in the game of avalon with large language models. In *Conference on Empirical Methods in Natural Language Processing*.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Xingyao Wang, Zihan Wang, Jiateng Liu, Yangyi Chen, Lifan Yuan, Hao Peng, and Heng Ji. 2023. Mint: Evaluating llms in multi-turn interaction with tools and language feedback. *ArXiv*, abs/2309.10691.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2022a. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Huai hsin Chi, and Denny Zhou. 2022b. Self-consistency improves chain of thought reasoning in language models. *ArXiv*, abs/2203.11171.

Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, et al. 2022a. Emergent abilities of large language models. *arXiv preprint arXiv:2206.07682*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. 2022b. Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems*, 35:24824–24837.

Wikipedia. 2023a. Algebraic notation (chess) — Wikipedia, the free encyclopedia. http://en.wikipedia.org/w/index.php?title=Algebraic%20notation%20(chess)&oldid=1184027217. [Online; accessed 15-December-2023].

Wikipedia. 2023b. Forsyth–Edwards Notation — Wikipedia, the free encyclopedia. http://en.wikipedia.org/w/index.php?title=Forsyth%E2%80%93Edwards%20Notation&oldid=1176345997. [Online; accessed 15-December-2023].

Zhenyu Wu, Ziwei Wang, Xiuwei Xu, Jiwen Lu, and Haibin Yan. 2023. Embodied task planning with large language models. *arXiv preprint arXiv:2307.01848*.

Zhiheng Xi, Wenxiang Chen, Xin Guo, Wei He, Yiwen Ding, Boyang Hong, Ming Zhang, Junzhe Wang, Senjie Jin, Enyu Zhou, et al. 2023. The rise and potential of large language model based agents: A survey. *arXiv preprint arXiv:2309.07864*.

Jiannan Xiang, Tianhua Tao, Yi Gu, Tianmin Shu, Zirui Wang, Zichao Yang, and Zhiting Hu. 2023. Language models meet world models: Embodied experiences enhance language models. *arXiv preprint arXiv:2305.10626*.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W Cohen, Ruslan Salakhutdinov, and Christopher D Manning. 2018. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. *arXiv preprint arXiv:1809.09600*.

Shunyu Yao, Howard Chen, John Yang, and Karthik Narasimhan. 2022a. Webshop: Towards scalable real-world web interaction with grounded language agents. *ArXiv*, abs/2207.01206.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L Griffiths, Yuan Cao, and Karthik Narasimhan. 2023. Tree of thoughts: Deliberate problem solving with large language models. *arXiv preprint arXiv:2305.10601*.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2022b. React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*.

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

A Prompt Templates

In this section, we provide all the prompt templates used in this work. There are three types of prompts for each <game, agent> pair: system prompt, head prompt, observation prompt, and step prompt.

System Prompt. A system prompt in Large Language Models (LLMs) is a predefined instruction or command embedded within the model's interface, guiding its responses or actions according to specific user needs or operational protocols.

We utilize the following sentence as the system prompt for all the environments:

System Prompt: You are a helpful assistant who strictly follows the userś instructions.

Head Prompt. Head Prompts provide high-level descriptions of games, including game rules and symbol representation formats.

Observation Prompt. An observation prompt provides necessary information and observations to the reasoning agent, such as currently available actions, opponent moves, etc.

Step Prompt. Step prompts define how agents reason given prompts. Different agents may contain more than 1 step prompt. All the variables are denoted as <variable_name>.

A.1 Environment Prompt Templates for Tic-Tac-Toe

Head Prompt: Tic Tac Toe is a two-player game played on a grid. Players take turns marking a space with their respective symbols. The goal is to get multiple of oneś own symbols in a row, either horizontally, vertically, or diagonally, before the opponent does. If all nine squares are filled and no player has three in a row, the game is a draw. The Tic Tac Toe game is played on a 3 by 3 grid, with the winning length as 3. Each move is represented by a string consisting of two parts: the column (C) and the row (R), in that order. For instance, C1R2 means the movement at the position of the first column and the second row of the grid. You are playing this game with the user (opponent).

Observation Prompt: Now, your opponent has finished moves: <opponent_moves>. You have finished moves: <agent_moves>. The legal positions are <legal_moves>.

A.2 Environment Prompt Templates for Texas Hold'em Poker

Head Prompt: You are playing Texas Holdem Poker with other <num_players> players. The aim of each player in Texas Hold'em poker is to win chips or money from other players by either having the best hand at showdown or by convincing other players to fold their hands. The small blind bet of this game is 1 and the big blind bet of this game is 2.

Observation Prompt: Here are the situations you are facing:
You are in the <stage> round at present.
<round_prior_player_actions>.
The current round pot is <round_pot> and the community pot is <community_pot>.
Your card is <card>.
Your remaining stack is <remaining_stack>.
round_prior_player_actions: In this round, after the small blind and big blind actions, the prior players have made the following actions: Player at <player_info> takes action <action>.

A.3 Environment Prompt Templates for Connect-4

Head Prompt: Connect 4 is a two-player connection board game, where the players choose a color and then take turns dropping colored discs into a vertically suspended grid. The pieces fall straight down, occupying the next available space within the column. The objective of the game is to be the first to form a horizontal, vertical, or diagonal line of four of one's own discs. You are a gaming agent that aims to beat me in Connect 4 games. Each move is represented by a string consisting of two parts: the column (C) and the row (R), in that order. For instance, C1R2 means the movement at the position of the first column and the second row of the grid.

Observation Prompt: Now, your opponent has finished moves: <opponent_moves>. You have finished moves: <agent_moves>. The legal positions are <legal_moves>.

A.4 Step Prompt Templates for the `Prompt` Agent

Step Prompt: Choose one move from these legal positions to set up advantages.
Your output should be of the following format:
Move:
Your move

A.5 Step Prompt Templates for the CoT Agent

Step Prompt: First think about your current situation, then choose one move from legal positions to set up advantages.
Your output should be of the following format:
Thought:
Your thought.
Move:
Your move.

B Texas Hold'em Poker Action Space

The explanations of Texas Holdem Poker actions:

1. **FOLD**: You decide not to play the hand and discard your cards.

2. **CHECK**: Declining the opportunity to bet. It's like saying 'I'm still in the game, but I don't want to bet right now.

3. **CALL**: Matching the current highest bet to stay in the hand.

4. **RAISE_3BB**: Raising the bet to three times the big blind amount.

5. **RAISE_HALF_POT**: Raising to an amount equal to half the current pot size.

6. **RAISE_POT**: Raising to an amount equal to the current pot size.

7. **RAISE_2POT**: Raising to an amount equal to twice the current pot size.

8. **ALL_IN**: Betting all your chips.

9. **SMALL_BLIND**: A forced bet that's typically half the size of the big blind. It rotates around the table.

10. **BIG_BLIND**: A forced bet that sets the initial pot amount and action. It's typically twice the size of the small blind and rotates around the table.

C Reasoning Agent Adaptions

As we mentioned before, agents like ReAct and ToT are not specifically designed for strategic thinking. Here we provide our adaptions regarding the two agents.

C.1 Adaptions to the ReAct agent

We follow the prompts from their official codebase and utilize the first-think-then-action procedures. One of the major challenges is that we need to design search spaces for our tasks. For example, in (Yao et al., 2022b), the action space defined for the Hotpot QA dataset is SEARCH[entity], LOOKUP[entity], and FINISH. To do that, we design the following actions for strategic reasoning:

Defensive Action, which means to block the potential winning of your opponent (e.g., block your opponent from forming sequences of 3).
Offensive Action, which means to win the game (e.g., create forks, control the center, play ahead).

We first prompt LLMs to select which type of action is more desirable, defensive or offensive. Then, based on the selected action, we prompt LLMs to select the next move. The overall step prompt for ReAct is as follows:

Step Prompt: Solve this problem with first Thought then Action final Move steps. The Thought step reasons about the current situation to set up advantages. The Action step will select one of the 2 actions:
(1) Defensive Action, which means to block the potential winning of your opponent (e.g., block your opponent from forming sequences of 3).
(2) Offensive Action, which means to win the game (e.g., create forks, control the center, play ahead).
The Move step will generate your next <env_name> move.
Your output should be in the following format:
Thought:
Your thought here.
Action:
Your action here.
Move:
Your move.

C.2 Adaptions to the ToT agent

For the ToT agent, we follow the implementation of the text generation task as in the official codebase of ToT[5]. Specifically, follow the 2-step ToT manner, i.e., 1) generate plans; 2) vote for the plan; 3) generate action according to the selected plan; 4) vote for action. The prompts used in this process are shown as follows:

Step Prompt: First think about your current situation, then choose one move from legal positions to set up advantages.
Your output should be of the following format:
Thought:
Your thought.
Move:
Your move.

After executing step prompts in a breath-first search manner, we utilize the following voting prompt to select the plan and move:

Vote Prompt: Conclude in the last line "The best choice is s", where s is the integer id of the choice.

D Generative Hyperparameters

For all the model queries and generations, we set the max token number as 1024 and the temperature as 0.2. For other parameters, we follow the default settings as in OpenAI API and Langchain interfaces.

E Step Prompt Templates for ReTA

Selection Prompts: First think about your situations, then choose <num_k> moves from legal positions to set up advantages. Your output should be in the following format:
Thought:
Your thought.
Selection:
1. selected move
2. selected move
......

Evaluation Prompts: Assume you will take <next_move> as the next move. What is the advantage score for this move? Use a score on a scale of 0 - 100 to represent this score.
Conclude in the last line "The advantage score for me is s", where s is the score.

[5] https://github.com/princeton-nlp/tree-of-thought-llm/blob/master/src/tot/prompts/text.py