MARSBench: Evaluating Multi-Agent Multi-Turn Strategic Reasoning of Large Language Models and Beyond

Anonymous ACL submission

Abstract

Current logical reasoning benchmarks 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. We introduce *MARSBench*, a novel framework to evaluate 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*). *MARSBench* offers two distinct scenarios: 1) *Online Racing*, featuring multiple LLMs/agents to facilitate direct competition and comparison; 2) *Offline Probing*, constructing targeted questions and verified ground truth to evaluate LLMs' strategic behaviors. We show that existing state-of-the-art LLMs and reasoning schemes are largely ineffective for strategic reasoning tasks. For instance, GPT-3.5-turbo with advanced Tree-of-Thought (`ToT`) is only slightly better than a `Random` agent in the naive Tic-Tac-Toe. Offline probing indicates that these LLMs suffer from serious hallucinations (e.g., spatial understanding) and weak strategic thinking (e.g., endgame). A recursively thinking-ahead agent is proposed to strengthen the strategic reasoning of LLMs. We hope *MARSBench* could spur further research and exploration in the multi-turn strategic reasoning of LLMs.

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 race 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 introduce *MARSBench*, a comprehensive benchmark for evaluating multi-turn strategic online and offline reasoning of LLMs, encompassing 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]. These games have simple rules, clear criteria, limited action/state space, and controllable difficulties, which make them suitable for current LLM evaluations.

[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

Benchmarks	Multi-Turn	Multi-Agent	Strategic	Opponent	Reasoning Agents
HotpotQA (Yang et al., 2018)	✗	✗	✗	✗	✔
ALFWorld (Shridhar et al., 2020)	✔	✗	✗	✗	✔
VirtualHome (Puig et al., 2018)	✔	✗	✗	✗	✗
TextWorld (Côté et al., 2018)	✗	✗	✔	✗	✗
MINT (Wang et al., 2023)	✔	✗	✔	✗	✔
AgentBench (Liu et al., 2023c)	✔	✗	✔	✗	✔
WebArena (Zhou et al., 2023)	✔	✗	✔	✗	✔
MARSBench (ours)	✔	✔	✔	✔	✔

Table 1: Comparisons between *MARSBench* and existing reasoning benchmarks.

MARSBench introduces two different evaluation paradigms: *Online Racing* and *Offline Probing*. For online racing, *MARSBench* facilitates direct competitions among multiple LLMs, allowing for a straightforward comparison of their reasoning skills by pitting them against each other in a race. Apart from LLMs, *MARSBench* also evaluates advanced reasoning paradigms, e.g., 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).

In terms of offline probing, *MARSBench* supports demographic analysis by constructing error-driven questions and verified ground truth, for a detailed analysis of LLMs' strategic behaviors. As a demonstration, we examine LLM races and summarize 9 common strategic reasoning errors for general board games: 5 for hallucination (e.g., *spatial understanding*, *pattern recognition*) and 4 for strategic thinking (e.g., *endgame*). Then we evaluate LLMs over 2.7k questions specifically designed for each error type and quantify their performances. Our experimental results suggest that existing LLMs suffer from both hallucination (Duan et al., 2023; Manakul et al., 2023) and reasoning errors (Bian et al., 2023; Karpinska and Iyyer, 2023; Gekhman et al., 2023), such as judging *fork* (two-win moves) and assessing *priority*. At last, we propose a Recursively Thinking Ahead (`ReTA`) agent equipped with uncertainty quantification mechanisms, to strengthen the strategic reasoning of LLMs, which also serve as a competitive baseline in our benchmark. Our contributions can be summarized as the following:

- ***MARSBench*:** We propose *MARSBench* for evaluating the under-explored multi-turn strategic reasoning capabilities of LLMs, through a set of complete-/incomplete-information board and card games. HotpotQA (Yang et al., 2018) is a challenging QA dataset, necessitating multi-hop reasoning skills such as retrieval and search from LLMs.

- **Online LLMs Racing:** *MARSBench* offers online competitions among multiple LLMs and reasoning agents, allowing for a straightforward comparison of their reasoning skills

- **Offline Reasoning Probing:** *MARSBench* provides targeted questions and verified ground truth, regarding the common errors during reasoning, for detailed demographic analysis of the strategic reasoning capabilities of LLMs.

- **Improved Strategic Reasoning:** We propose a recursively thinking ahead agent, with mechanisms such as majority vote and uncertainty quantification for hallucination control, to further strengthen the strategic reasoning of LLMs.

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 VirtualHome (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. There have been a lot of benchmarks aiming to evaluate tool utilization capabilities (e.g., web browsering), 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

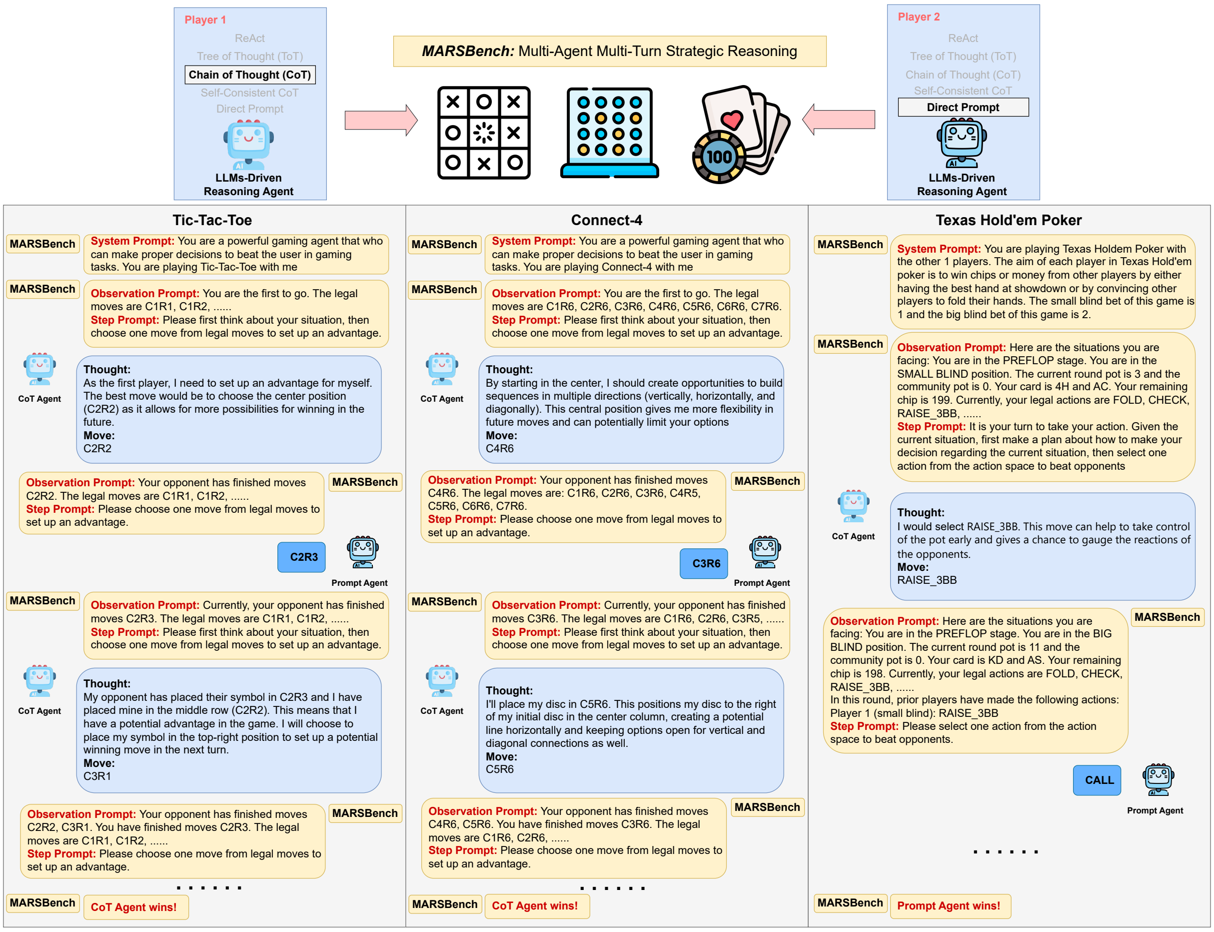


Figure 1: LLMs online racing in multi-turn strategic scenarios.

for LLMs-as-agents, from the perspective of code, web, and game.

The differences between *MARSBench* and existing benchmarks are summarized in Table 1.

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., 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 *MARSBench*: Online Agents Racing

MARSBench facilitates online competition among LLMs and agents, providing a versatile platform for assessing strategic reasoning capabilities. Figure 1 presents the procedures of online LLMs racing and the demonstration of each environment.

3.1 Preliminary

We present online LLMs racing 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 uti-

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 2: 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.

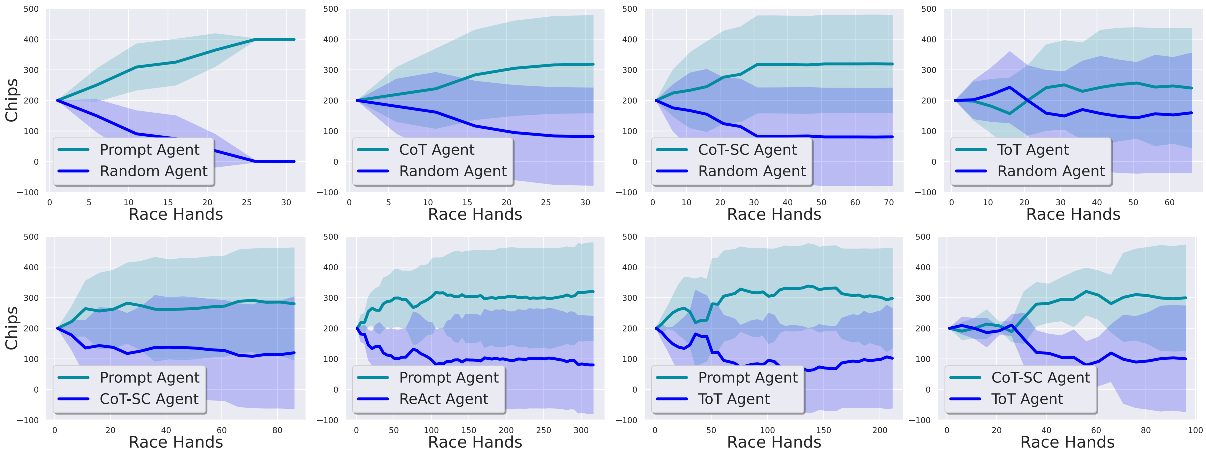


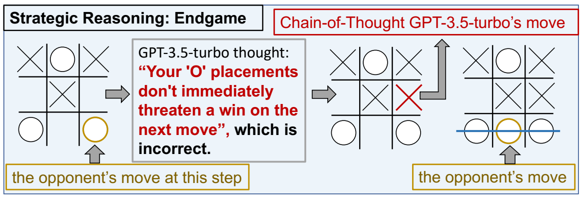
Figure 2: Remaining chips of reasoning agents at each hand in the Texas Hold'em Poker environment. Standard deviations over 20 trials are shown as the shadowed areas. Agents with more remaining chips at last mean better performance. Among these agents, the naive Prompt agent works better than other methods.

lize 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.

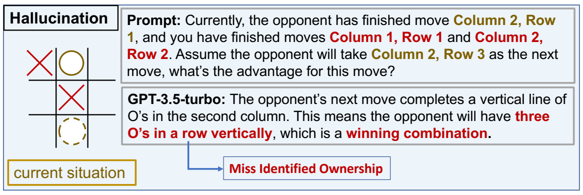
Texas Hold'em Poker[4]: Each agent is assigned \$200 chips initially. The agent is prompted to select an action from the action set: FOLD, CHECK, CALL, RAISE_3BB, RAISE_HALF_POT, RAISE_POT, RAISE_2POT, ALL_IN, SMALL_BLIND, BIG_BLIND. The utilized prompts can be found in Appendix A.2. Detailed explanations 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

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



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



(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.

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 to *MARSBench*.

3.2 Evaluation Results

In Table 2, we report the average win ratios and loss ratios in the *MARSBench* 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` 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 obtained experimental results in *MARSBench*:

Serious Hallucination and Reasoning Errors. We found that LLMs suffer from serious hallucinations and reasoning errors. Figure 3 provides

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 3: Statistics of the *MARSBench* offline datasets.

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 *MARSBench*: Offline In-Depth Probing

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 by MARSBench?*

4.1 Preliminary

To answer this question, *MARSBench* provides targeted questions and verified answers for detailed offline demographic analysis. As a demonstration, we show how *MARSBench* characterizes LLMs' strategic behaviors over board games (e.g., Tic-Tac-Toe and Connect-4). We first examine LLMs' behaviors from online races 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

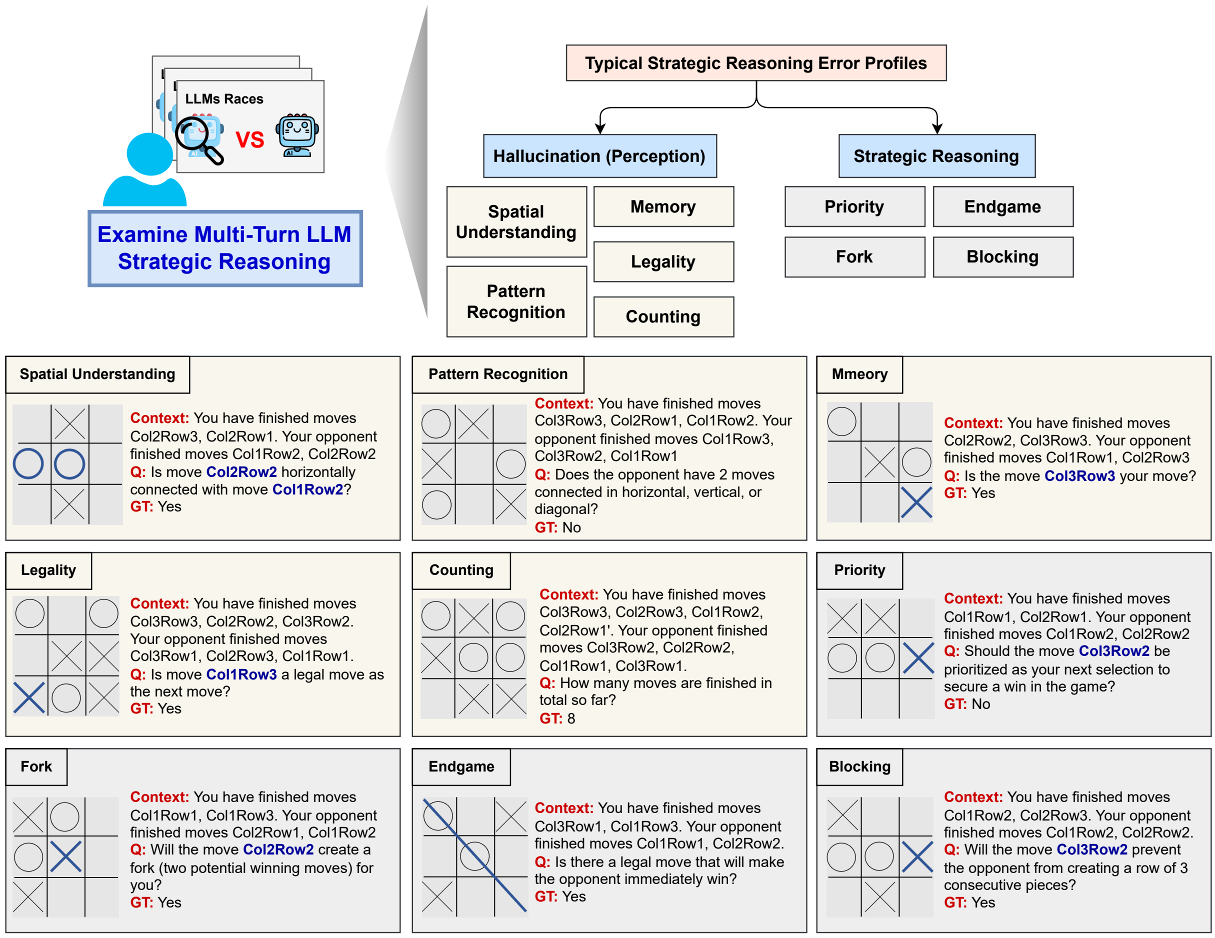


Figure 4: Error profiles in *MARSBench* offline dataset.

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.

4.2 Offline Dataset Generation

Utilizing structured symbols for each move, such as <CxRy>, *MARSBench* can 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 3.

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 4.

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-

Model and Reasoning	All Avg.	Hallucination (Perception) spatial	pattern	counting	memory	legalty	*avg.*	Strategic Reasoning 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 4: Evaluation results of *MARSBench* 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.

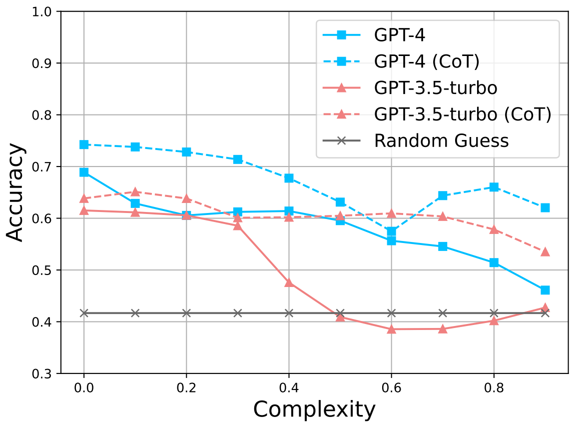


Figure 5: Correlations between board complexities and model performances. It indicates that complex board situations result in a significant performance drop for state-of-the-art LLMs.

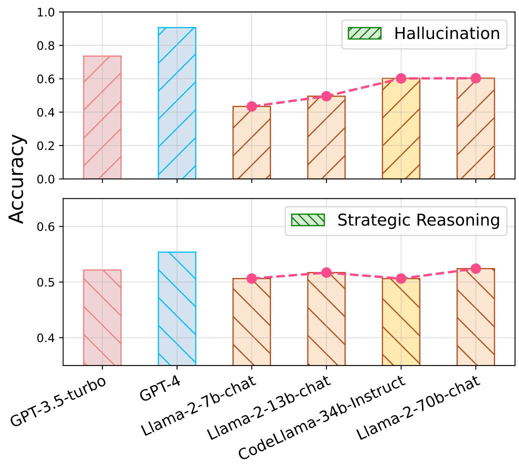


Figure 6: The emergent abilities in strategic reasoning. Increasing model parameter sizes effectively mitigate hallucination and perception errors, while it does not yield similar improvements in strategic.

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 races 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 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 the emergent abilities in strategic linguistic understanding.

However, there is no such trend in those strategic thinking evaluation. We show that Llama-2-7b-

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

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.

5 Improved Multi-Turn Reasoning

We propose preliminary mechanisms for improved strategic reasoning agents in this section, which serve as competitive baselines for *MARSBench*.

5.1 Improved Reasoning

We introduce three mechanisms: Recursively Think Ahead (ReTA), Majority Vote, and Perturbation-based Uncertainty Quantification to improve the strategic reasoning:

Recursively Thinking Ahead (ReTA). It is inspired by the conventional min-max gaming theory where the MinMax agent aims to maximize its own advantage while minimizing the opponent's potential gains. Following that, we introduce an imaginary enemy concept during reasoning and prompt LLMs to play ahead against this imaginary opponent for up to k steps and select n moves for each step. This process involves the agent systematically analyzing potential moves and outcomes, both of its own and of the imaginary enemy. After this thinking-ahead process, the agent is prompted to choose the best move considering the actions of the imaginary enemy as the next move. Detailed architecture of ReTA can be found in Appendix E.

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 6: 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 7: Evaluation of ReTA in Connect-4.

Majority Vote. For a given question x, we simply sample k_{mv} generations from LLMs and select the high-frequency option or the mean value (if it is a numerical situation) as the next move.

Perturbation-based Uncertainty Quantification. We first prompt LLMs to perturb the original question x for k_{pert} times while keeping the semantics unchanged, then we sample generations based on both original question x and perturbed questions $\tilde{x}$ and apply a majority vote over these generations.

5.2 Experimental Settings and Evaluations

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 5, 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 6 and 7, the empirical results obtained over Texas Hold'em Poker and Connect-4 present that ReTA could be generalized to other scenarios.

6 Conclusion

In this paper, we propose *MARSBench*, a comprehensive benchmark for multi-turn strategic reasoning of LLMs. *MARSBench* provides online agent racing 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 *MARSBench* considers 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.

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.

Marc-Alexandre Côté, Ákos Kádár, Xingdi Yuan, Ben A. Kybartas, Tavian Barnes, Emery Fine, James Moore, Matthew J. Hausknecht, Layla El Asri, Mahmoud Adada, Wendy Tay, and Adam Trischler. 2018. Textworld: A learning environment for text-based games. *ArXiv*, abs/1806.11532.

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*.

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.

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. In *MARSBench*, 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 Recursively Thinking Ahead (ReTA)

As we mentioned in Section 5.1, to make LLMs think ahead, we incorporate an imaginary enemy before the LLMs makes the final decision.

E.1 Implementation

To simulate this process, we formulate ReTA as the ensemble of modules, utilizing multiple individual actors:

- **Main Actor** M: interacting with the environment, gathering feedback from other actors, 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 external instructions/feedback.
- **Reward Actor** M_R: working as a signal function to evaluate the reward signals of different actions, 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 for M, M_R and M_O, respectively.

Assume at the beginning of gaming step t, e.g., the t-th turn, 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, the think-ahead process is formulated as the *pseudo-gaming* between M and 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 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 select the action with the highest rewards as the next move. The key design of recursively thinking ahead is the imaginary opponent M_O that tries to block the winning of M and the minimax reward signal backtracking.

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

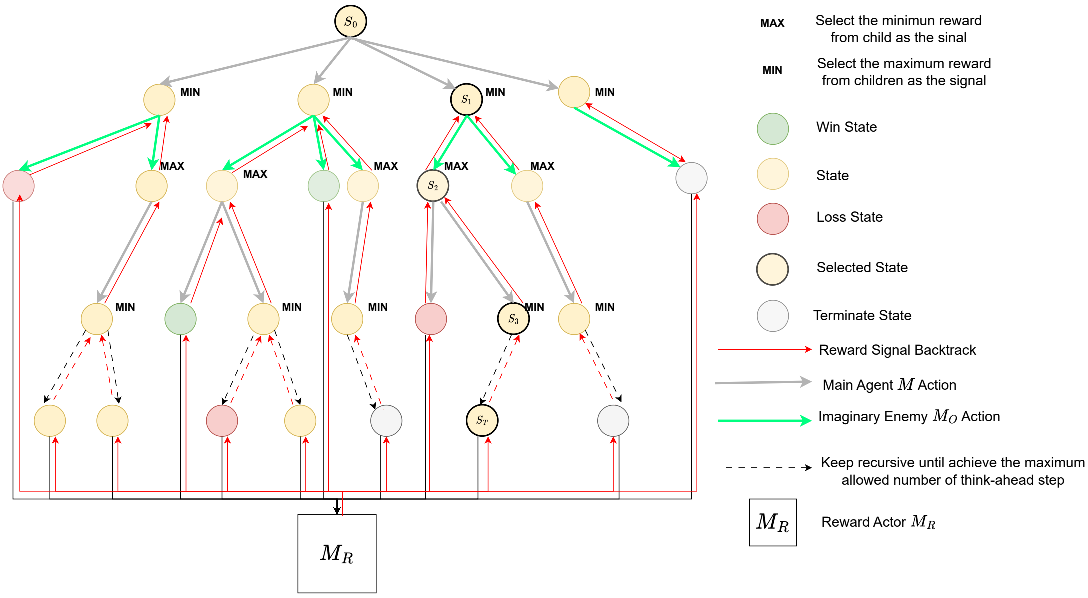


Figure 7: The tree representation of the proposed recursively think ahead in ReTA.

E.2 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.