Tracing LLM Reasoning Processes with Strategic Games: A Framework for Planning, Revision, and Resource-Constrained Decision Making

Anonymous Author(s)
Affiliation
Address
email

Abstract

Large language models (LLMs) are increasingly applied to tasks that require complex reasoning. While most benchmarks focus on evaluating final reasoning outcomes, they overlook the internal processes that lead to those outcomes—such as how a model plans, revises, and makes decisions under constraints. We argue that evaluating these internal reasoning steps is essential for understanding model behavior and improving reliability in real-world applications. To make these processes observable and measurable, we propose using strategic games as a natural and effective environment. These games operate within closed, rule-based systems and provide interpretable states, limited resources, and automatic feedback.

Therefore, we propose a framework to evaluate LLMs along three core process dimensions: planning, revision, and resource-constrained decision making. To support this, we introduce a set of evaluation metrics that extend beyond traditional win rates, incorporating measures such as Over-correction Risk Rate, correction success rate, improvement slope, and over-budget ratio.

In a set of 4320 adversarial rounds across 12 state-of-the-art models, we find that ChatGPT-o3-mini, which demonstrates strong planning capabilities, achieves the highest composite process score (74.7%win rate, 78.6% correction success, and a +0.041 improvement slope). In contrast, Qwen-Plus, despite a high Overcorrection Risk Score of 81.6%, wins only 25.6% of its matches, primarily due to excessive resource use. We also observe a negative correlation between Over-correction Risk Rate and correction success rate (Pearson r = –0.51, p = 0.093), suggesting that more frequent corrections do not always improve outcomes. This pattern may reflect impulsive revision strategies, where premature edits reduce overall effectiveness, while more selective approaches lead to greater accuracy. We hope this work offers a new direction for LLM evaluation—focusing not just on what models decide, but on how they decide it.

1 Introduction

Large language models (LLMs) are now capable of solving increasingly complex reasoning tasks [14, 34]. As their performance on traditional benchmarks improves, it has become clear that measuring *outcome accuracy* alone is no longer sufficient. In many real-world scenarios, the quality of an LLM's reasoning depends not only on the final answer, but also on the internal processes it uses to arrive there: how it plans, how it revises mistakes, and how it makes decisions under resource constraints.

We argue that understanding these *reasoning processes* is a necessary next step in LLM evaluation. Current benchmarks—such as GSM8K [6] or MMLU—offer single-turn questions and measure

Submitted to 39th Conference on Neural Information Processing Systems (NeurIPS 2025). Do not distribute.

correctness in isolation. They provide limited visibility into how a model generates hypotheses, updates them in response to feedback, or adjusts its strategy over time. Automatically generated questions have been proposed to avoid memorization [32], but these bring their own issues, such as variable difficulty and occasional invalidity [18, 33].

To address this, we propose shifting the evaluation paradigm: from static, outcome-based tests toward dynamic, process-aware environments. We identify **strategic games** as a particularly well-suited testbed for this purpose. Games provide closed, rule-based environments with clear feedback signals, bounded resources, and interpretable decision traces. Their structure allows us to directly observe and quantify multi-step reasoning behaviors—without requiring human annotations or handcrafted evaluation rubrics.

In this work, we introduce **AdvGameBench**, a process-based evaluation framework that embeds LLMs in interactive, resource-constrained strategy games. Rather than judging success solely by win rates, our framework traces how models form plans, revise them when needed, and operate under strict resource budgets. We define a set of core evaluation dimensions—**planning**, **revision**, and **resource-constrained decision making**—and propose concrete metrics that capture each of them.

To support broad and interpretable analysis, AdvGameBench spans three classic game genres—tower defense, auto battler, and turn-based combat—each chosen to expose different cognitive and strategic demands. The framework logs full model outputs and action traces, enabling detailed inspection of decision quality, revision behavior, and adherence to constraints.

Our key contributions are:

- A formalization of reasoning process dimensions: planning, revision, and resource-constrained decision making.
- A game-based evaluation framework that instantiates these dimensions using closed, interpretable, and reproducible environments.
- A suite of evaluation metrics that measure not only whether models succeed, but how they reason through the task.

2 Related work

LLMs in Gaming Applications. LLMs have rapidly evolved and demonstrated significant capabilities across various complex tasks, including gaming scenarios[22][28][12][19][15]. Early studies mainly investigated their performance in text-based adventure games. For example, Tsai et al. [24] examined the capabilities of ChatGPT within interactive fiction. Subsequent research expanded to strategic and multi-agent scenarios. Notably, Akata et al. [1] explored repeated two-player interactions such as the Prisoner's Dilemma, highlighting the models' strengths in cooperative scenarios and coordination challenges.

Recent attention has increasingly shifted toward multiplayer and complex card games. Yim et al. [31] studied Guandan, a sophisticated Chinese card game characterized by imperfect information and team cooperation. Their research demonstrated that prompting LLMs with Theory of Mind-like strategies significantly improved collaborative performance, but also revealed critical gaps in managing long-horizon states. Similarly, Hu et al. [13] proposed GameArena, a benchmark designed to evaluate fine-grained reasoning skills of LLMs through specialized interactive games.

Existing Benchmarks for LLM Evaluation Despite these advancements, current benchmarks rely predominantly on simplified textual or stylized environments.AppWorld [23] , GTBench [9], GAMEBENCH [7], MINT [27], and AgentBench [16] illustrate established efforts focusing on puzzle, multi-turn interactions, or agent-oriented tasks. Furthermore, Yang et al. [29] provided benchmarks specifically for StarCraft II, showcasing sophisticated summarization techniques in strategic gaming contexts. Another research direction evaluates strategic reasoning using game-theoretic frameworks, demonstrating how sophisticated models like GPT-4 [20] approximate human decisions, but often fail to achieve a true rational equilibrium in adversarial or coordination-focused scenarios [17, 10].

Multimodal and embodied approaches have also emerged as significant subfields, exemplified by works such as Voyager in Minecraft [25] and evaluations of the use of the LLM tool [30]. However, these approaches primarily tackle open-world exploration or general-purpose tasks rather than structured competitive scenarios common in mainstream gaming genres like tower defense or auto battlers. In addition, they often require frequent API interactions or repeated prompts, raising practical cost and latency concerns [26, 5].

In contrast to previous benchmarks that rely on tool-calling or open-ended exploration, **AdvGameBench** evaluates LLMs within strategic, rule-based environments where decision-making processes are directly observable. The framework eliminates external dependencies, imposes explicit budget constraints, and embeds models in turn-based adversarial settings. This design enables systematic analysis of not only final outcomes but also intermediate behaviors.

3 Method

3.1 Multi model adversarial structure

We introduce a structured adversarial framework for evaluating LLMs' **process-level reasoning behaviors**—specifically, how models **plan**, **revision**, and **resource-constrained decision making** in rule-constrained environments.

Figure 1: This figure illustrates the AdvGameBench evaluation pipeline. Three strategic game genres—tower defense, auto-battler, and turn-based combat—form the core environments for model evaluation. In each round, the model generates a strategy based on explicit rules; the simulator executes the strategy and returns an outcome; the model optionally revises its plan; and this process iterates over multiple rounds. These interactions yield behavioral trajectories from which process-level metrics are computed, including win rate, over-correction risk rate, correction success rate, and over-budget rate.

Game-based, closed-loop setting. Each evaluation match embeds two LLMs in a *closed, deterministic game simulator* governed by explicit rules and resource constraints. Models receive identical prompts and independently generate strategies. The simulator executes both strategies and returns a rule-verifiable win/loss outcome.

Role alternation across diverse games. We construct three adversarial games—*tower defense, auto-battler, and turn-based combat*—each targeting a distinct reasoning capability. In every round, models alternate between attacker and defender roles, exposing both offensive and defensive strategy formation.

Feedback-driven revision. After each round, models receive outcome-based feedback. They may optionally revise their strategy. These revision sequences are logged and scored using **process-aware metrics** including correction success rate, over-correction risk, and improvement slope.

Control for asymmetry. To eliminate bias, we evaluate each model pair under both move orders. This ensures symmetry and isolates model-specific behaviors from structural advantages.

Adversarial matrix evaluation. The complete setup yields a dense match matrix, covering all model pairs, roles, and move orders. This enables *systematic comparison* of revision dynamics, constraint adherence, and planning robustness under matched conditions.

Original game reimplementations. All game environments are re-implemented with shifted design from popular games to avoid strategy leakage from popular games. This ensures that models cannot rely on memorized heuristics or latent familiarity with existing game patterns, preserving the objectivity of the evaluation. All these code will be public available.

3.2 Game suites: Tower Defense, Auto-battler, Turn-based

To evaluate how LLMs revise and adapt across varied reasoning contexts, we design three game environments that span distinct forms of strategic complexity. Each environment imposes different constraints and interaction patterns: **Tower Defense** emphasizes spatial planning under sequential threats, **Battle Card** requires resource allocation and composition under outcome uncertainty, and **Turn-based Game** tests decision consistency across multi-step attribute interactions. This diversity ensures that our evaluation covers a broad spectrum of process-level reasoning behaviors.

Tower Defense Game. In this environment, models alternate between attacker and defender roles. Defenders place units on an 11-column battlefield to block attackers advancing from the right. Attackers aim to reach the left boundary, while defenders strive to destroy all attackers. Success criteria and rule violations provide clear feedback for iterative strategy refinement (see A.1).

Battle Card Game. Models control units with distinct attributes: attackers prioritize damage, defenders emphasize protection and recovery. Units engage in automated battles, with combat sequence determined by the number of units each side possesses. The side that eliminates all opposing units first wins, offering explicit outcome-based feedback for model improvement (see A.2).

Turn-based Attribute Game. Each side controls three characters with assigned elemental attributes (Fire, Wood, Water, Earth, Light, Dark), featuring strategic interactions based on attribute strengths and weaknesses. Characters choose three skills within a budget constraint, cycling through them in combat. Duels continue until one side remains, clearly indicating the strategic effectiveness and compliance of each model's choices (see A.3).

3.3 Evaluation metrics

To evaluate LLMs beyond raw outcomes, we define a set of metrics tracing how models revise strategies, manage constraints, and adapt over time. We categorize our metrics into three groups:

- **Outcome metric**: measures overall performance.
- **Revision behavior metrics**: assess how models respond to failure.
- **Constraint adherence metrics**: quantify rule compliance under resource limits.

Win Rate (WR). Win Rate measures the proportion of matches a model wins out of all played games, with rule violations resulting in immediate forfeiture. This metric captures the final outcome of the reasoning process and provides a baseline for comparison. It reflects how well a model integrates planning, revision, and constraint handling into an executable solution.

Over-Correction Risk Rate (ORR). ORR captures how frequently a model reacts to negative feedback with a revised proposal. This metric targets a critical behavior: over-adjustment in response to failure signals. In practical settings, excessive self-editing can reduce decision stability and degrade coherence over long horizons. High ORR indicates a lack of strategic confidence or an overly reactive revision policy. The need to track this behavior is grounded in the observation that models can degrade their own solutions through unnecessary changes, even when initial plans are viable.

Correction Success Rate (CSR). CSR measures how often a revision leads to an improved result—either by eliminating a rule violation or by turning a loss into a win. This metric isolates the effectiveness of the model's internal feedback loop: can it not only detect failure but also recover from it? A model that frequently edits without reliably improving demonstrates superficial adaptivity rather than meaningful self-correction.

Improvement Slope (β). Improvement Slope captures whether a model improves over repeated interactions in matched environments. This measures whether the model can adapt its planning based on prior failures against a fixed opponent type. Unlike static metrics, β traces whether a model learns or degrades over time. A flat or negative slope suggests overfitting or myopic adjustment; a positive slope reflects effective cumulative reasoning.

Over-Budget Rate (OBR). OBR measures how often a model generates proposals that exceed explicit resource constraints. This metric directly evaluates a model's ability to integrate symbolic or numerical limits into its reasoning process. Many LLMs can optimize performance under unconstrained conditions, but OBR reveals whether they can internalize hard boundaries and behave accordingly. This behavior is essential for real-world deployment, where compliance with external rules is not optional but required for safe execution.

Together, these metrics provide complementary views into different layers of model behavior: WR evaluates final success; ORR and CSR analyze revision dynamics; β measures adaptation over time; and OBR enforces structural discipline. Further detailed metrics are discussed in Appendix B.

4 Results

We evaluate 12 leading LLMs, including DeepSeek-R1/V3 [8], Qwen-Plus/Max [4], Claude-3.5-Sonnet [3], ChatGPT-4.1/4o/o3/o3-mini [20], Gemini-2/2.5-Flash [2], and LLaMA-3-70B [11]. All models use the same decoding settings: temperature 0.3 and top-p 1, allowing for controlled but non-deterministic generation [21].

To assess robustness, each model was tested against three diverse opponents: ChatGPT-4o, Claude-3.5-Sonnet, and DeepSeek-V3. This setup avoids evaluation bias caused by shared architectures or training data. In each round, models play against all opponents in turn-based games, with the platform logging win/loss results and correction behaviors for downstream analysis.

4.1 Revision behavior: correction rate & success

Table 1: Benchmark Metrics (WR = win rate, ORR = over-correction risk, CSR = correction success)

Model	TDG			BCG			TAG			avg		
	WR	ORR	CSR	WR	ORR	CSR	WR	ORR	CSR	WR	ORR	CSR
ChatGPT-4.1	45.0	85.7	40.0	52.5	69.7	65.2	57.5	82.4	67.9	51.7	79.4	56.8
ChatGPT-4o	65.8	81.8	55.6	60.8	44.0	63.6	59.1	82.4	46.4	58.6	70.4	52.6
ChatGPT-o3	75.8	41.1	57.1	76.7	50.0	88.9	70.0	30.0	66.7	74.2	40.0	73.7
ChatGPT-o3-mini	63.3	25.9	57.1	74.2	31.6	100.0	86.7	9.0	100.0	74.7	24.5	78.6
Claude-3-5-Sonnet	56.7	89.3	56.0	45.8	70.0	64.3	55.0	76.9	65.0	52.5	77.7	61.6
DeepSeek-R1	70.8	53.6	80.0	49.2	32.2	40.0	80.0	83.3	70.0	66.7	48.4	63.3
DeepSeek-V3	43.3	84.6	45.5	23.3	75.5	24.3	56.7	75.0	38.1	41.1	78.4	35.2
Gemini-2-Flash	15.8	90.6	10.4	49.2	65.7	60.9	38.3	67.5	28.0	34.4	76.8	27.1
Gemini-2.5-Flash	60.0	40.0	60.0	59.0	79.2	68.4	58.1	76.2	56.3	62.5	65.7	63.0
LLaMA-3-70B	33.3	90.2	29.7	42.5	76.3	51.7	65.0	69.2	66.7	46.9	80.0	45.2
Qwen-Max	39.2	44.7	5.8	10.8	50.0	10.3	41.7	51.3	36.9	30.5	48.9	16.9
Qwen-Plus	19.2	78.4	20.0	16.7	81.5	13.6	40.8	86.1	45.2	25.6	81.6	24.3

Table 1 shows the win rate, over-correction risk rate, and correction success rate for evaluated models.

Win Rate. ChatGPT-o3-mini and ChatGPT-o3 achieved the highest win rates at 74.7% and 74.2%, respectively, substantially outperforming all other models. These results suggest strong capabilities in planning and decision-making under adversarial conditions. In contrast, models such as the Qwen series and Gemini-2-Flash exhibited significantly lower win rates, indicating weaker performance in high-pressure strategic settings.

Over-correction Risk Score. This metric reflects a model's tendency to overreact to feedback through frequent revisions. Qwen-Plus, DeepSeek-V3, and Claude-3-5-Sonnet exhibited high Over-correction Risk Rates (ORR), suggesting an unstable decision-making process characterized by impulsive or excessive adjustments. In contrast, ChatGPT-o3-mini maintained a relatively low ORR of 49.3%, indicating a more disciplined and stable strategy that avoids unnecessary revisions unless a confident improvement is identified.

Correction Success Rate. This measures the effectiveness of the attempted revisions. ChatGPT-o3-mini achieved the highest success rate at 78.6%, indicating that most of its corrections were accurate. Conversely, Qwen-Max and Qwen-Plus had success rates around 20% despite frequent corrections, reflecting a tendency toward uninformed or premature changes—what we refer to as "blind correction."

These findings highlight an important distinction: frequent correction behavior does not necessarily imply improved performance. High-performing models engaged in fewer revisions, but these were more targeted and successful. In contrast, models that frequently attempted corrections without sufficient understanding failed to translate effort into meaningful gains. Effective revision thus requires not just responsiveness, but discernment in identifying when and how to intervene.

4.2 Planning ability: Init-win & Improvement Slope

We evaluate planning capabilities using two complementary metrics: initial win rate (init-win), which reflects first-round performance without feedback, and improvement slope, which measures a model's ability to enhance its strategy over time. Together, they capture a model's capacity to start strong and adapt through interaction. **Figure 2** shows win rate trajectories across five rounds, while **Figure 3** reports improvement slopes.

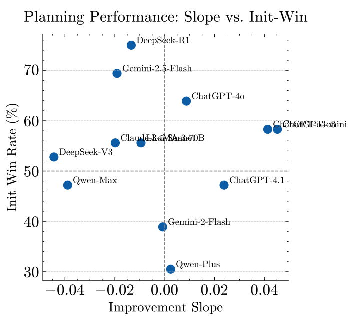


Figure 2: Planning performance: slope vs. initial win rate.

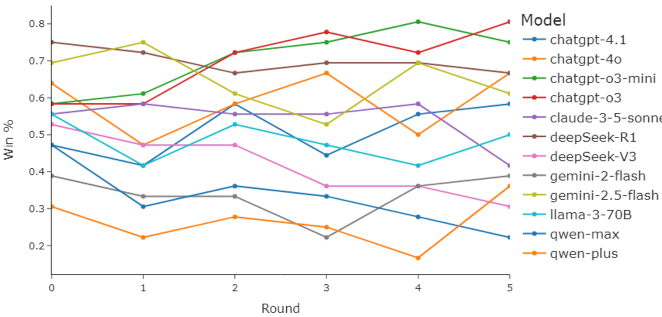


Figure 3: Win-rate trajectories across five rounds.

DeepSeek-R1 achieves the highest init-win (75.0%) but declines over time, suggesting rigid strategy design. In contrast, ChatGPT-o3 and o3-mini start with lower win rates (58.3%) yet steadily improve, indicating flexible planning. Models like Gemini-2.5-Flash and Claude-3.5-Sonnet perform well initially but regress, likely due to static heuristics. Qwen models show little progress, pointing to weak feedback integration. Across families, only ChatGPT models consistently improve, reflecting stronger adaptation. These patterns show that robust planning requires not just strong openings, but the ability to refine strategies under pressure—a key dimension captured by process-level metrics like the improvement slope.

4.3 Resource-constrained decision making

Figure 4 reports the Over-Budget Ratio (OBR), which quantifies the proportion of turns in which a model exceeds the environment's resource constraints. While most models stay within budget in over 80% of turns, the variation across models is notable. ChatGPT-o3 and ChatGPT-o3-mini maintain perfect budget adherence, never exceeding the allowed limits. In contrast, Qwen-Plus surpasses its budget in approximately half of its turns, and Qwen-Max records similarly high overuse (OBRs of 0.50 and 0.45, respectively). This pattern is strongly aligned with performance: the o3 series models not only exhibit the lowest OBRs but also achieve the highest win rates (74.7% and 74.2%), whereas the Qwen models, with the highest OBRs, perform worst in terms of win rate (30.5% and 25.6%).

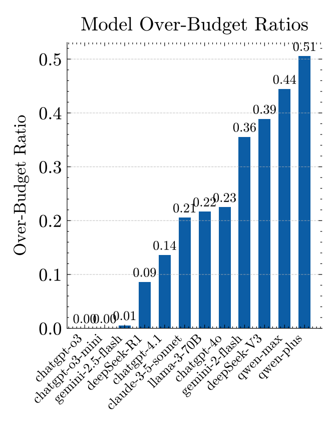
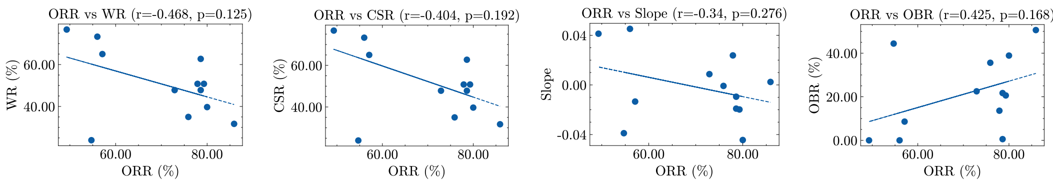


Figure 4: Over-Budget Ratio for Each Model

We further find a strong negative correlation between OBR and win rate (Pearson r = –0.95, p < 0.001), indicating that effective resource management is closely tied to model success. High OBRs are often associated with reactive, post-hoc revisions—corrections made after poor initial decisions—which typically fail to compensate for early mistakes. Conversely, models with low OBRs demonstrate more disciplined planning and efficient execution, avoiding costly errors in the first place. These results position OBR as a meaningful process-level indicator that goes beyond outcome accuracy, revealing how well models translate abstract constraints into concrete and consistent decision-making. Strong performers not only remain within budget but also allocate their resources strategically, contributing to higher correction success and overall coherence in behavior.

4.4 Does revising more really help?



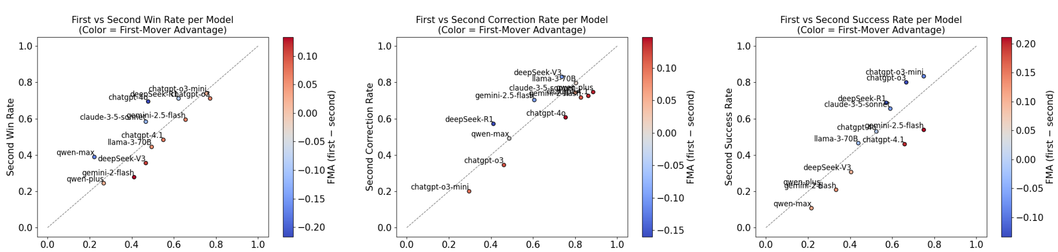
(a) Correlation between ORR and WR (b) Correlation between ORR and CSR (c) Correlation between ORR and Slope (d) Correlation between ORR and OBR

Figure 5: Correlation analysis between over-correction risk rate (ORR) and four main metrics across models

We quantify a model's tendency to revise reactively using the **over-correction risk rate (ORR)**—the probability that a model submits a new strategy immediately after receiving explicit negative feedback. **Figure5** presents the correlation between ORR and four process-level outcomes. We observe a moderate negative relationship between ORR and final win rate ($r = -0.47$, $p = 0.13$), suggesting that models which revise more frequently tend to achieve lower overall success. Similarly, ORR correlates negatively with improvement slope ($r = -0.34$, $p = 0.28$), indicating that frequent edits do not accelerate strategic refinement. In terms of budget use, models with higher ORR values are more likely to exceed resource constraints (OBR; $r = +0.43$, $p = 0.17$), and also show lower correction success rates ($r = -0.34$, $p = 0.28$), implying that high-frequency revision may undermine the quality of attempted corrections.

Although none of these effects reach conventional thresholds for statistical significance due to the limited sample size ($n = 12$), the consistency in directional trends is notable. Across all four measures, models with high over-correction risk tend to perform worse: they are less efficient, less successful overall, and less disciplined in their resource usage. In contrast, top-performing models such as CHATGPT-O3-MINI pair a low ORR with high correction success and zero budget violations. These results highlight a key insight: **precision in revision—not frequency—is the hallmark of effective strategy adjustment**.

4.5 Role symmetry and first-move bias



(a) Win rate: first-move vs. second-move (b) Over-correction risk rate: first-move vs. second-move (c) Correction success rate: first-move vs. second-move

Figure 6: First-mover advantage (FMA) across three behavioral dimensions.

We use First-Mover Advantage (FMA) to examine how model performance differs when initiating an action versus responding to a prior move. We analyze this effect across three dimensions: win rate, over-correction risk rate, and correction success rate. As shown in Figure 6a, most models exhibit relatively minor differences in win rate between first- and second-mover roles, with FMA values generally within five percentage points. This suggests limited systematic advantage based on turn order for overall success. However, several models deviate from this trend. Gemini-2-Flash (FMA = +13.2%) performs substantially better when acting first, while ChatGPT-4o (FMA = –21.7%) and Qwen-Max (FMA = –16.7%) exhibit the opposite pattern, achieving higher win rates when playing second. These results suggest that certain models are more sensitive to the structural asymmetries introduced by move order.

Stronger patterns emerge when examining correction behavior. In Figure 6b and Figure 6c, we observe that most models show a clear preference for initiating rather than responding. For example, ChatGPT-4o and ChatGPT-4.1 demonstrate significantly higher over-correction risk rates when acting first (FMA = +14.8% and +13.8%, respectively). Similarly, first-mover performance gains are evident in correction success rates for Gemini-2.5-Flash (+21.2%), Gemini-2-Flash (+12.5%), and ChatGPT-4.1 (+19.9%). These findings underscore the importance of accounting for role asymmetry in evaluation setups. Our dual-first configuration helps mitigate first-mover bias, offering a more balanced and interpretable view of model behavior under asymmetric game dynamics.

4.6 Holistic comparison via radar chart

To synthesize model performance across reasoning dimensions, we constructed a radar chart visualizing five normalized metrics: win rate (WR), correction success rate (CSR), improvement slope, 1 – over-correction risk rate (ORR), and 1 – over-budget rate (OBR). All metrics were scaled to a common range, with inversions applied where necessary so that higher values consistently indicate better performance. This unified view enables a comparative assessment of both outcome and process quality across models.

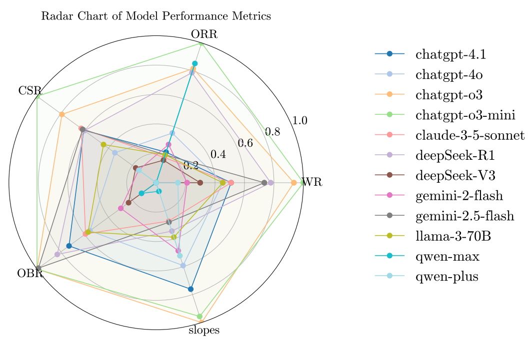


Figure 7: Model performance metrics

ChatGPT-o3 and o3-mini form the largest radar areas, reflecting strong, consistent performance across all dimensions. They pair high win rates with effective corrections, stable improvement, and disciplined resource use, indicating well-integrated reasoning. In contrast, models like Qwen-Plus and Qwen-Max show sharp imbalances, marked by frequent but ineffective revisions and frequent budget violations. Gemini models perform moderately in CSR and planning but are similarly constrained by high correction risk or poor budget control. These patterns highlight that top performance requires balance across planning, revision, and constraint adherence—not just isolated strength. Larger radar areas correspond to more robust reasoning pipelines, reinforcing that process quality is essential to understanding model competence.

4.7 Model-Specific Strengths and Underlying Mechanisms

Our evaluation shows that different LLMs exhibit distinct process-level strengths, often reflecting differences in architecture and alignment. Models from the ChatGPT family—especially o3 and o3-mini—achieve consistently high win rates while maintaining disciplined correction behavior and strict budget adherence. This pattern suggests a stable internal revision mechanism, likely shaped by reinforcement learning with human feedback (RLHF) and conservative fine-tuning objectives that prioritize reliability over exploration. In contrast, models such as Qwen-Plus and DeepSeek-V3 revise frequently but achieve low correction success and often exceed resource limits. These behaviors point to reactive decision-making and overly eager feedback incorporation, which can destabilize planning over time. We refer to this pattern as "over-correction," where excessive responsiveness leads to fragmented strategies and reduced overall performance.

Other models, including Claude-3.5-Sonnet and Gemini-2.5-Flash, show more balanced profiles across metrics. While they do not dominate any single dimension, they perform moderately well in planning, correction, and resource management. This may reflect broader instruction-tuning or multitask training that encourages general adaptability without specializing in any one skill. Taken together, these differences underscore that LLM capabilities are shaped by design trade-offs: between caution and flexibility, local reactivity and global coherence. Our process-level metrics—particularly ORR and improvement slope—help reveal these trade-offs, offering a more nuanced view of model behavior than outcome-based evaluations alone. They also provide actionable insights into how alignment strategies and decoding preferences influence long-horizon reasoning, suggesting concrete directions for model development and benchmarking.

5 Discussion

Process-rather-than-outcome evaluation. AdvGameBench purposefully shifts the evaluation lens from *what* an LLM answers to *how* it arrives there. By embedding models in three rule-bound strategy games, we can observe—and score—their behaviour along the three dimensions defined in Method section (see Method).: *planning* (initial strategy quality and improvement slope), *revision* (correction rate and success), and *resource-constrained decision making* (budget adherence).

Empirical studies. Our study of twelve production-scale LLMs across **4 752** adversarial rounds yields three consistent findings:

1. **Integrated skill trumps single metrics.** Models that balance the three dimensions—notably ChatGPT–O3-Mini with a 74.7 % win rate, 78.6 % correction-success rate, and positive improvement slope of $+0.0413$—outperform models that excel in only one aspect.
2. **"Spray-and-pray" revision is counter-productive.** Qwen-Plus issues corrections in 81.6 % of error states yet wins only 25.6 % of games and overspends in nearly half the turns. Across all systems, correction frequency and efficacy are negatively correlated (Pearson $r = -0.51$, $p = 0.093$), indicating that *calibrated* self-editing matters more than sheer persistence.
3. **Budget fidelity is a leading indicator of success.** The two models that never violated resource limits (ChatGPT–O3 and ChatGPT–O3-Mini) also posted the highest win rates, whereas both Qwen variants combine the largest over-budget ratios with the poorest outcomes.

Hallucation. In our tower defense experiments, all defensive units were consistently referred to as *soldiers*. However, several models frequently generated the term *peashooter*, which was never introduced in the task instructions. A review of the interaction logs reveals that this phenomenon does not stem from a reasoning failure, but rather from prior associations learned during pretraining—specifically, the frequent co-occurrence of "tower defense" and the game *Plants vs. Zombies* in web-scale corpora. This leads models to default to familiar terminology, even when it conflicts with the defined rules of the environment. Such behavior undermines the validity of the benchmark, effectively turning the evaluation into a test of memorized correlations rather than genuine planning or constraint adaptation. To eliminate this form of memory bias, we redesigned the game environment to neutralize lexical cues and ensure that performance reflects models' ability to engage with novel rules and dynamic constraints, rather than recalling pretraining artifacts.

Limitations. AdvGameBench currently (i) covers three turn-based genres but no real-time or cooperative play, (ii) logs unit-level actions yet does not attribute win contributions to individual decisions, and (iii) relies on synthetic opponents, which—although diversified—cannot fully mirror human play styles. These choices were deliberate to keep the study controlled and reproducible, but they constrain external validity.

6 Conclusion

Static accuracy benchmarks have become an insufficient proxy for real-world robustness. Deployment-ready systems must also *plan soundly*, *revise judiciously*, and *respect resource constraints* to function effectively in practical environments. AdvGameBench meets this need by turning strategic gameplay into an open, extensible laboratory in which those process-level traits can be systematically quantified, monitored, and improved over time.

By exposing the entire decision trace—from initial plan through budgeted actions and self-corrections—the benchmark reveals failure modes that outcome-only tests often conceal. These fine-grained signals enable not only diagnostic analysis of model behavior but also principled design of training objectives that reward disciplined, context-sensitive reasoning under pressure. They also help evaluate whether models can maintain stability across repeated trials, even in adversarial or resource-limited conditions. AdvGameBench supports controlled ablations, adversarial setups, and resource perturbations, making it a flexible platform for probing model resilience.

Ultimately, we see AdvGameBench as one step toward a broader shift in LLM evaluation: away from asking only *"Did the model answer correctly?"* and toward asking *"How did the model reason, adapt, and stay within bounds while answering?"* Such process-aware scrutiny is essential for building language models that are not only accurate but also reliable, accountable, and aligned with real-world deployment demands.

References

[1] Akata, E., Schulz, L., Coda-Forno, J., Oh, S. J., Bethge, M., and Schulz, E. (2023). Playing repeated games with large language models. In *arXiv preprint arXiv:2305.16867v1*.

[2] Anil, R., Borgeaud, S., Alayrac, J.-B., Yu, J., Soricut, R., Schalkwyk, J., Dai, A. M., Hauth, A., Millican, K., Silver, D., Johnson, M., Antonoglou, I., Schrittwieser, J., Glaese, A., Chen, J., Pitler, E., Lillicrap, T., Lazaridou, A., Firat, O., Molloy, J., Isard, M., Barham, P. R., Hennigan, T., Lee, B., Viola, F., Reynolds, M., Xu, Y., Doherty, R., Collins, E., Meyer, C., Rutherford, E., Moreira, E., Ayoub, K., Goel, M., Krawczyk, J., Du, C., Chi, E., Cheng, H.-T., Ni, E., Shah, P., Kane, P., Chan, B., Faruqui, M., Severyn, A., Lin, H., Li, Y., Cheng, Y., Ittycheriah, A., Mahdieh, M., Chen, M., Sun, P., Tran, D., Bagri, S., Lakshminarayanan, B., Liu, J., Orban, A., Güra, F., Zhou, H., Song, X., Boffy, A., Ganapathy, H., Zheng, S., Choe, H., Weisz, Á., Zhu, T., Lu, Y., Gopal, S., Kahn, J., Kula, M., Pitman, J., Shah, R., Taropa, E., Al Merey, M., Baeuml, M., Chen, Z., El Shafey, L., Zhang, Y., Sercinoglu, O., Tucker, G., Piqueras, E., Krikun, M., Barr, I., Savinov, N., Danihelka, I., Roelofs, B., White, A., Andreassen, A., von Glehn, T., Yagati, L., Kazemi, M., Gonzalez, L., Khalman, M., Sygnowski, J., Frechette, A., Smith, C., Culp, L., Proleev, L., Luan, Y., Chen, X., et al. (2025). Gemini: A Family of Highly Capable Multimodal Models. In *arXiv preprint arXiv:2312.11805v5*. Version v1 submitted on 19 Dec 2023, v5 (this version) revised 9 May 2025.

[3] Anthropic (2024). The claude 3 model family: Opus, Sonnet, Haiku. Technical report, Anthropic. Model Card.

[4] Bai, J., Bai, S., Chu, Y., Cui, Z., Dang, K., Deng, X., Fan, Y., Ge, W., Han, Y., Huang, F., Hui, B., Ji, L., Li, M., Lin, J., Lin, R., Liu, D., Liu, G., Lu, C., Lu, K., Ma, J., Men, R., Ren, X., Ren, X., Tan, C., Tan, S., Tu, J., Wang, P., Wang, S., Wang, W., Wu, S., Xu, B., Xu, J., Yang, A., Yang, H., Yang, J., Yang, S., Yao, Y., Yu, B., Yuan, H., Yuan, Z., Zhang, J., Zhang, X., Zhang, Y., Zhang, Z., Zhou, C., Zhou, J., Zhou, X., and Zhu, T. (2023). QWEN TECHNICAL REPORT. In *arXiv preprint arXiv:2309.16609v1*.

[5] Chiang, W.-L., Zheng, L., Sheng, Y., Angelopoulos, A. N., Li, T., Li, D., Zhu, B., Zhang, H., Jordan, M. I., Gonzalez, J. E., and Stoica, I. (2024). Chatbot arena: An open platform for evaluating LLMs by human preference. In *arXiv preprint arXiv:2403.04132v1*.

[6] Cobbe, K., Kosaraju, V., Bavarian, M., Chen, M., Jun, H., Kaiser, L., Plappert, M., Tworek, J., Hilton, J., Nakano, R., Hesse, C., and Schulman, J. (2021). Training verifiers to solve math word problems. In *arXiv preprint arXiv:2110.14168v2*. Version v1 submitted on 27 Oct 2021, v2 (this version) revised 18 Nov 2021.

[7] Costarelli, A., Allen, M., Hauksson, R., Sodunke, G., Hariharan, S., Cheng, C., Li, W., Clymer, J., and Yadav, A. (2024). GAMEBENCH: Evaluating strategic reasoning abilities of llm agents. In *arXiv preprint arXiv:2406.06613v2*.

[8] DeepSeek-AI, Liu, A., Feng, B., Xue, B., Wang, B., Wu, B., and et al. (2024). DeepSeek-V3 Technical Report. In *arXiv preprint arXiv:2412.19437*.

[9] Duan, J., Zhang, R., Diffenderfer, J., Kailkhura, B., Sun, L., Stengel-Eskin, E., Bansal, M., Chen, T., and Xu, K. (2024). GTBENCH: Uncovering the strategic reasoning limitations of LLMs via game-theoretic evaluations. In *arXiv preprint arXiv:2402.12348v2*.

[10] Fan, C., Chen, J., Jin, Y., and He, H. (2023). Can large language models serve as rational players in game theory? a systematic analysis. In *arXiv preprint arXiv:2312.05488v2*.

[11] Grattafiori, A., Dubey, A., Jauhri, A., Pandey, A., Kadian, A., Al-Dahle, A., Letman, A., Mathur, A., Schelten, A., Vaughan, A., Yang, A., Fan, A., Goyal, A., Hartshorn, A., Yang, A., Mitra, A., Sravankumar, A., Korenev, A., Hinsvark, A., Rao, A., Zhang, A., Rodriguez, A., Gregerson, A., Spataru, A., Roziere, B., Biron, B., Tang, B., Chern, B., Caucheteux, C., Nayak, C., Bi, C., Marra, C., McConnell, C., Keller, C., Touret, C., Wu, C., Wong, C., Ferrer, C. C., Nikolaidis, C., Allonsius, D., Song, D., Pintz, D., Livshorn, D., Wyatt, D., Esiobu, D., Choudhary, D., Mahajan, D., Garcia-Olano, D., Perino, D., Hupkes, D., Lakomkin, E., AlBadawy, E., Lobanova, E., Dinan, E., Smith, E. M., Radenovic, F., Guzmán, F., Zhang, F., Synnaeve, G., Lee, G., Anderson, G. L.,

Thattai, G., Nail, G., Mialon, G., Pang, G., Cucurell, G., Nguyen, H., Korevaar, H., Xu, H., Touvron, H., Zarov, I., Ibarra, I. A., Kloumann, I., Misra, I., Evtimov, I., Zhang, J., Copet, J., Lee, J., Geffert, J., Vranes, J., Park, J., Mahadeokar, J., Shah, J., van der Linde, J., Billock, J., Hong, J., Lee, J., Fu, J., Chi, J., Huang, J., Liu, J., Wang, J., Yu, J., Bitton, J., Spisak, J., Park, J., Rocca, J., Johnstun, J., Saxe, J., Jia, J., et al. (2024). The Llama 3 Herd of Models. In *arXiv preprint arXiv:2407.21783v3*. Version v1 submitted on 31 Jul 2024, v3 (this version) revised 23 Nov 2024.

[12] Gupta, A. (2023). Are chatgpt and gpt-4 good poker players? — a pre-flop analysis. In *arXiv preprint arXiv:2308.12466v2*.

[13] Hu, L., Li, Q., Xie, A., Jiang, N., Stoica, I., Jin, H., and Zhang, H. (2025). GAMEARENA: Evaluating LLM reasoning through live computer games. In *arXiv preprint arXiv:2412.06394v5*.

[14] Huang, J. and Chang, K. C.-C. (2023). Towards reasoning in large language models: A survey. In *arXiv preprint arXiv:2212.10403v2*.

[15] Light, J. and et al. (2023). Avalonbench: Evaluating llms playing the game of avalon. *arXiv preprint arXiv:2310.05036*.

[16] Liu, Y., Li, Z., Liu, P., Xie, Y., Wu, B., Zhang, Y., Wang, S., Yu, Y., Zhao, J., Lu, Z., Gao, Y., Qiao, Y., Fan, W., Ye, Y., Liang, S., and Zhao, Y. (2023). AgentBench: Evaluating LLMs as agents. In *arXiv preprint arXiv:2308.03688*.

[17] Lorè, N. and Heydari, B. (2023). Strategic behavior of large language models: Game structure vs. contextual framing. In *arXiv preprint arXiv:2309.05898v1*.

[18] Minderer, M., Djolonga, J., Romijnders, R., Hubis, F., Zhai, X., Houlsby, N., Tran, D., and Lucic, M. (2021). Revisiting the calibration of modern neural networks. In *arXiv preprint arXiv:2106.07998v2*. Appeared in: 35th Conference on Neural Information Processing Systems (NeurIPS 2021). Version v1 submitted on 15 Jun 2021, v2 (this version) revised 26 Oct 2021.

[19] Nananukul, N. and Wongkamjan, W. (2024). What if red can talk? dynamic dialogue generation using large language models. In *arXiv preprint arXiv:2407.20382v1*.

[20] OpenAI (2024). GPT-4 technical report. In *arXiv preprint arXiv:2303.08774v6*.

[21] Renze, M. and Guven, E. (2024). The effect of sampling temperature on problem solving in large language models. In *arXiv preprint arXiv:2402.05201v3*.

[22] Sudhakaran, S., González-Duque, M., Freiberger, M., Glanois, C., Najarro, E., and Risi, S. (2023). MarioGPT: Open-ended Text2Level generation through large language models. In *arXiv preprint arXiv:2302.05981v3*.

[23] Trivedi, H., Khot, T., Hartmann, M., Manku, R., Dong, V., Li, E., Gupta, S., Sabharwal, A., and Balasubramanian, N. (2024). AppWorld: A controllable world of apps and people for benchmarking interactive coding agents. In *arXiv preprint arXiv:2407.18901v1*.

[24] Tsai, C. F., Zhou, X., Liu, S. S., Li, J., Mei, H., and Yu, M. (2023). Can large language models play text games well? current state-of-the-art and open questions. In *arXiv preprint arXiv:2304.02868v1*.

[25] Wang, G., Xie, Y., Jiang, Y., Mandlekar, A., Xiao, C., Zhu, Y., Fan, L., and Anandkumar, A. (2023a). VOYAGER: An open-ended embodied agent with large language models. In *arXiv preprint arXiv:2305.16291v2*.

[26] Wang, S., Long, Z., Fan, Z., Wei, Z., and Huang, X. (2024). Benchmark self-evolving: A multi-agent framework for dynamic LLM evaluation. In *arXiv preprint arXiv:2402.11443v1*.

[27] Wang, Y., Yu, D., Dong, L., Bao, F. S., Wang, X., Wang, D., Yu, Z., Li, L., and Zhou, H. (2023b). MINT: Evaluating LLMs in multi-turn interaction with tools and language feedback. In *arXiv preprint arXiv:2310.06825*.

[28] Xu, Y., Wang, S., Li, P., and et al. (2023). Exploring large language models for communication games: An empirical study on werewolf. *arXiv preprint arXiv:2309.04658*.

[29] Yang, Z., Li, H., Chen, Y., Tian, W., Ren, Y., Su, H., Zhu, J., and Sun, L. (2023a). Large language models play StarCraft II: Benchmarks and a chain of summarization approach. In *arXiv preprint arXiv:2310.11432*.

[30] Yang, Z., Li, L., Wang, J., Lin, K., Azarnasab, E., Ahmed, F., Liu, Z., Liu, C., Zeng, M., and Wang, L. (2023b). MM-REACT: Prompting ChatGPT for multimodal reasoning and action. In *arXiv preprint arXiv:2303.11381v1*.

[31] Yim, Y., Chan, C., Shi, T., Deng, Z., Fan, W., Zheng, T., and Song, Y. (2024). Evaluating and enhancing LLMs agent based on theory of mind in Guandan: A multiplayer cooperative game under imperfect information. In *arXiv preprint arXiv:2408.02559v1*.

[32] Yu, X., Cheng, H., Liu, X., Roth, D., and Gao, J. (2024). ReEval: Automatic hallucination evaluation for retrieval-augmented large language models via transferable adversarial attacks. In *arXiv preprint arXiv:2310.10190v2*. Version v1 submitted on 19 Oct 2023, v2 (this version) revised 31 May 2024.

[33] Zhang, H., Da, J., Lee, D., Robinson, V., Wu, C., Song, W., Zhao, T., Raja, P., Zhuang, C., Slack, D., Lyu, Q., Hendryx, S., Kaplan, R., Lunati, M., and Yue, S. (2024a). A careful examination of large language model performance on grade school arithmetic. In *arXiv preprint arXiv:2405.00212v4*. Version v1 submitted on 1 May 2024, v4 (this version) revised 22 Nov 2024. Original v1 arXiv:2405.00212.

[34] Zhang, Y., Mao, S., Ge, T., Wang, X., de Wynter, A., Xia, Y., Wu, W., Song, T., Lan, M., and Wei, F. (2024b). LLM as a mastermind: A survey of strategic reasoning with large language models. In *arXiv preprint arXiv:2404.01230v1*.

A Appendix

A.1 Tower defense game

A.1.1 Game rules

1. Players can purchase characters and place them on the battlefield. The battlefield consists of 5 rows (corresponding to y-coordinates 0-4). The human side can place units in a designated area spanning 11 columns (corresponding to x-coordinates 0-10).
2. Demons spawn from the right side of the battlefield (x-coordinates 11) and move left. Human units are placed on the left side of the battlefield, remain stationary, and attack approaching enemies.
3. All units attack according to their attack interval, automatically attacking when their cooldown ends. Defending units fire bullets or activate skills to attack enemies. Invading units engage in melee attacks when they come into contact with defending units.
4. Each grid cell can only contain one human unit at a time. Placing a new unit in an occupied cell is not allowed.
5. When an attack hits, the target takes damage based on the attacker's power. If a unit's health drops to 0, it is eliminated and removed from the battlefield.
6. If all enemies are eliminated, the player wins. If any enemy successfully reaches the left side of the battlefield, the player loses.

A.1.2 Human units

Unit	Attributes
HandgunSoldier	Health: 3, Shooting interval: 1000ms, Cost: 100, Damage per shot: 1, No special abilities.
RifleSoldier	Health: 3, Shooting interval: 500ms, Cost: 200, Damage per shot: 1, No special abilities.
MachineGunSoldier	Health: 3, Shooting interval: 250ms, Cost: 400, Damage per shot: 1, No special abilities.

ShieldSoldier	Health: 15, Cost: 50, Only for defense, no attack capabilities.
EnhancedShieldSoldier	Health: 30, Cost: 100, Only for defense, no attack capabilities and Bouncing Demon cannot jump over.
FlamethrowerSoldier	Health: 2, Cost: 200, Shooting interval: 1000ms, Damage per shot: 1, Deals an additional 1 damage every 1000ms.
IceSoldier	Health: 2, Shooting interval: 1000ms, Cost: 200, Damage per shot: 1, Reduces enemy speed by half.
AntiAirSoldier	Health: 2, Shooting interval: 1000ms, Cost: 175, Damage per shot: 1, Can attack airborne units.
Bomb	Health: 50, Detonation time: 500ms, Cost: 200, Explosion range: 3×3, Damage per explosion: 30, Destroyed after detonation.
LinearExplosion	Health: 50, Detonation time: 500ms, Cost: 200, Explosion range: the entire row, Damage per explosion: 30, Destroyed after detonation.
MagneticSoldier	Health: 2, Shooting interval: 2000ms, Cost: 100, Damage per shot: 0, Releases a magnetic pulse that disables the defensive abilities of ShieldDemon and MachineDemon.
LightMage	Health: 2, Damage per shot: 0, Cost: 150, No attack capabilities, Changes the attributes of bullets in the same row, converting their damage type to light.
RocketLauncherSoldier	Health: 2, Shooting interval: 1000ms, Damage per shot: 2, Cost: 600, Launches rockets, dealing damage to enemies within one grid.

A.1.3 Demon units

Note: A speed of 2 requires 14 seconds to travel from spawn to the last human grid.

Unit	**Attributes**
NormalDemon	Health: 10, Speed: 2, Attack interval: 1000ms, Cost: 100, Damage per attack: 1, No special abilities.
GreatDemon	Health: 20, Speed: 2, Attack interval: 1000ms, Cost: 175, Damage per attack: 1, Higher health.
DemonKing	Health: 100, Speed: 2, Attack interval: 1000ms, Cost: 800, Damage per attack: 5.
SpeedyDemon	Health: 10, Speed: 4, Attack interval: 1000ms, Cost: 150, Damage per attack: 1, Moves faster.
ShieldDemon	Health: 10, Speed: 2, Attack interval: 1000ms, Cost: 175, Damage per attack: 1, Takes 70% less damage from normal attacks.
MachineDemon	Health: 20, Speed: 2 (increases to 3 when activated), Attack interval: 1000ms, Cost: 250, Damage per attack: 3, Reduced damage due to mechanical body.
BouncingDemon	Health: 10, Speed: 2, Attack interval: 1000ms, Cost: 150, Damage per attack: 1, Can jump over certain units except for the EnhancedShieldSoldier.
ShieldBreakerDemon	Health: 10, Speed: 2, Attack interval: 1000ms, Cost: 150, Damage per attack: 1 (×5 against shield units).
FireDemon	Health: 10, Speed: 2, Attack interval: 1000ms, Cost: 150, Damage per attack: 1, Immune to fire damage.
FrostDemon	Health: 10, Speed: 2, Attack interval: 1000ms, Cost: 150, Damage per attack: 1, Immune to ice damage and unaffected by slow effects.

FlyingDemon	Health: 10, Speed: 2, Attack interval: 1000ms, Cost: 200, Damage per attack: 1, Only affected by anti-air attacks and can pass through human units directly.
ShadowDemon	Health: 10, Speed: 2, Attack interval: 1000ms, Cost: 300, Damage per attack: 1, Can cast dark magic, making same-row allies immune to non-light damage.
SummoningDemon	Health: 10, Speed: 1, Attack interval: 1000ms, Cost: 300, Damage per attack: 1, Summons a Normal Demon to the left grid every 5000ms.

A.2 Battle card game

A.2.1 Game rules

1. At the start of the game, players can purchase all desired characters at once, up to a maximum of 7 characters. Gold characters cost three times as much as bronze characters, but their stats (attack, health, numerical skill effects, etc.) are twice as high. Non-numerical skills are not affected by this multiplier.
2. Initiative Determination: The side with more characters attacks first. If both sides have the same number of characters, the invader attacks first.
3. Elemental Advantage: Certain elements have an advantage over others, granting a bonus in combat (Fire > Nature, Nature > Water, Water > Earth, Earth > Fire).
4. Battle Process: Both sides will attack based on their respective target_priority (target priority). However, if there are Taunt minions on the opponent's side, attackers must prioritize attacking them. The attack order follows a left-to-right sequence. The first minion in the invaders or defenders list (as defined in the JSON file) will attack first, depending on which side has the initiative. After that, the first minion from the opposing side attacks. Then, the second minion from the attacking side follows, then the second minion from the opposing side, and so on in an alternating pattern. If a minion's health reaches zero, it is eliminated. The battle continues with both sides attacking in turns until one side is completely wiped out, resulting in victory for the other side.
5. If all characters on one side are eliminated, the other side wins.
6. If both sides are eliminated simultaneously in the same attack resolution, the Invader wins.

A.2.2 Invader units

Unit	**Attributes**
FireLizard	Attack: 2, Health: 2, Cost: 1, Ability: Deals 2 damage to the enemy that killed it upon death.
WaterElemental	Attack: 2, Health: 2, Cost: 1, Ability: Gains +1 Attack when attacking.
PoisonFrog	Attack: 1, Health: 1, Cost: 2, Ability: Instantly destroys any minion it damages.
MoltenHound	Attack: 3, Health: 1, Cost: 2, Ability: Deals 1 damage to all enemies upon death.
BattleFrenzy	Attack: 7, Health: 4, Cost: 2, Ability: Each attack reduces its Attack by 4.
BanditLeader	Attack: 8, Health: 3, Cost: 3, Ability: Any excess damage from an attack carries over to the next target.
LavaGolem	Attack: 1, Health: 8, Cost: 3, Ability: Forces enemies to attack this minion first, Burns the attacker for 3 damage per turn when hit.
TideGuardian	Attack: 4, Health: 2, Cost: 3, Ability: Absorbs the first source of damage taken (divine shield), Attacks twice each turn.
TideLord	Attack: 4, Health: 9, Cost: 5, Ability: Doubles its Attack when taking damage.

Phoenix	Attack: 5, Health: 5, Cost: 5, Ability: Deals damage equal to its Attack to the target and its adjacent enemies, Revives with full Health after being defeated once per game.
ShadowOverlord	Attack: 4, Health: 4, Cost: 5, Ability: Summons a Slow Skeleton (3/1) upon death.

A.2.3 Defender units

Unit	Attributes
Sapling	Attack: 2, Health: 2, Cost: 1, Ability: Gains +1 Health when attacking.
RockBeetle	Attack: 1, Health: 5, Cost: 1, Ability: Forces enemies to attack this minion before others.
ForestSeer	Attack: 2, Health: 2, Cost: 2, Ability: At the start of the game, grants +1 Attack and +2 Health to all Nature Allies.
StoneWarrior	Attack: 2, Health: 5, Cost: 2, Ability: Forces enemies to attack this minion before others. Summons a RockBeetle upon death.
EliteSoldier	Attack: 1, Health: 1, Cost: 2, Ability: At the start of the game, grants Divine Shield to adjacent minions and +1 Attack.
Paladin	Attack: 3, Health: 6, Cost: 3, Ability: Has Divine Shield; gains +2 Attack whenever a friendly minion loses its Divine Shield.
BlackRock	Attack: 5, Health: 1, Cost: 3, Ability: At the start of the game, gains +3 Health for each friendly minion.
VineProtector	Attack: 5, Health: 4, Cost: 3, Ability: Upon death, restores 2 Health to all friendly minions.
King	Attack: 3, Health: 10, Cost: 5, Ability: Summons a 2/2 Soldier with Divine Shield whenever it attacks (if there is an open space).
MountainGiant	Attack: 4, Health: 9, Cost: 5, Ability: Forces enemies to attack this minion first, Reduces the attack of the attacker by 2 when hit.
AncientTreant	Attack: 4, Health: 4, Cost: 5, Ability: At the start of the game, grants +3 Attack and +3 Health to all allied minions.

A.3 Turn-based attribute game

A.3.1 Game rules

1. This game is a turn-based character battle game divided into two factions: Invader and Defender. Each faction consists of three characters. The Invader faction includes Fire, Water, and Dark elements, while the Defender faction includes Wood, Earth, and Light elements. Characters appear and act in the order they are listed in the data.
2. Combat proceeds in rounds. In each round, the three Invader characters act first in order, followed by the three Defender characters. The sequence then repeats in the next round.
3. Each character has three skills that are used in a preset, looping sequence. On each turn, a character uses the next skill in their list and continues cycling through them in order.
4. The game features an elemental effectiveness system: Fire beats Wood, Wood beats Earth, Earth beats Water, and Water beats Fire (1.2$\times$ damage when effective, 0.8$\times$ when resisted). Light and Dark counter each other with 1.5$\times$ damage. All other combinations deal the standard 1.0$\times$ damage.
5. If all characters on one side are eliminated, the other side wins.

A.3.2 Invader skills

Skill Name	Description
Fire Skills	
flame_splash	Deals 12 damage and applies Burning for 2 rounds (1 layer, 5 damage per round). Cost: 1
residual_warmth	Increases the damage of the next fire-based skill by 30% for 1 round. Cost: 1
burst_flame_bomb	Deals 25 base damage, plus 3 additional damage for each Burning layer on the target. Cost: 2
flame_whirlwind	Applies 4 layers of Burning to the target, lasting 2 rounds. Each layer deals 5 damage per round. Cost: 2
magma_eruption	Deals 40 base damage, plus 5 extra damage per Burning layer. Removes all Burning after the attack. Cost: 3
hell_curtain	Deals 35 damage and grants a shield that reflects 30 melee damage, lasting 2 rounds (1 layer). Cost: 3
Water Skills	
stream_pierce	Deals 10 damage and grants 1 permanent layer of Tidal Surge. Cost: 1
water_barrier	Grants a 5-point shield for 3 rounds and increases Tidal Surge by 1 layer. Cost: 1
whirlpool_strangle	Deals 20 base damage, plus 4 additional damage per Tidal Surge layer. Cost: 2
ice_branded	Deals 15 damage and causes the target to take 50% more damage next turn (1 round). Cost: 2
tsunami_ending	Deals 30 base damage, plus 5 additional damage per Tidal Surge layer. Removes all Tidal Surge after the attack. Cost: 3
abyss_resonance	Deals 3 damage per Tidal Surge layer and grants a shield worth 6 per layer, lasting 3 rounds. Cost: 3
Dark Skills	
shadow_claw	Deals 14 damage and heals the user for 30% of the damage dealt (rounded down). Cost: 1
fear_whisper	Reduces the target's damage taken by 10% for 3 rounds (1 layer). Cost: 1
soul_siphon	Deals 25 damage. If the target's HP is below 50%, deals an extra 15 damage. Cost: 2
night_ambush	Deals 20 damage and causes the target to take 20% more damage next turn (1 round). Cost: 2
final_announcment	Deals 45 base damage, plus 5 extra damage for every 10% HP the target has lost. Cost: 3
void_assimilation	Sacrifices 20% of current HP to deal penetration damage equal to twice the HP sacrificed. Cost: 3

A.3.3 Defender skills

Skill Name	Description
Wood Skills	
bud_healing	Grants Bud Healing for 3 rounds, restoring 6 HP per round. Cost: 1
parasitic_seed	Applies Parasitic Seed for 3 rounds, immediately deals 10 damage. The target takes 5 counter damage each time they attack. Cost: 1
life_totem	Restores 25 HP and grants Life Totem for 3 rounds, increasing healing received by 10%. Cost: 2
natural_purification	Removes negative statuses from the user and deals 30 damage to the target. Cost: 2
forest_reincarnation	Restores 60 HP. If it exceeds max HP, the excess is converted into a shield (50% of excess HP) for 3 rounds. Also deals 20 damage to an enemy. Cost: 3

poison_vine	Applies Poison Vine for 3 rounds, dealing 25 damage per round. Cost: 3
Earth Skills	
rock_armor	Grants 12 shield for 3 rounds and reflects 5 melee damage while the shield is active. Cost: 1
earth_shock	Deals 20 damage. Cost: 1
granite_barrier	Grants Granite Barrier for 3 rounds, decreasing damage by 40%. Cost: 2
quicksand_trap	Applies Quicksand Trap for 3 rounds. The target's next 3 damage are delayed by 20% and each trigger deals 10 damage. Cost: 2
earth_pulse	Grants shield based on HP lost (8 shield per 10% HP lost), lasting permanently. Cost: 3
core_rebound	Deals 80% of stored damage to the target. Clears stored damage after use. Cost: 3
Light Skills	
holy_glimmer	Removes a negative status (if any) and restores 8 HP to the user. Also deals 8 light damage to an enemy. Cost: 1
faith_emblem	Grants Faith Emblem for 1 round. The next damage taken is reduced by 20% and converted into healing. When triggered, deals 10 damage to the attacker. Cost: 1
divine_link	Grants Divine Link for 1 round. The next damage taken is reflected back to the attacker. Cost: 2
luminous_dispel	Removes one buff from the target (if any) and applies a debuff for 2 rounds that reduces their attack by 15%. Cost: 2
angelic_sanctuary	Grants Angelic Sanctuary for 3 rounds, reducing all incoming damage by 30 points. Cost: 3
divine_sword	Deals 20 damage and grants a buff that increases the next skill's damage by 20. Cost: 3

B Additional evaluation metrics

This section details supplementary metrics used to provide a more granular understanding of LLM behavior in strategic game environments, complementing the core metrics presented in Section 3.4.

B.1 Rule violation Rate (RVR)

This metric measures how often a model's initial strategy proposal fails to adhere to the game's explicit rules, particularly budget constraints. Let M_i denote model i. Let T_i be the total number of initial strategy proposals made by model M_i across all games and rounds where it provides an initial strategy. For each initial strategy proposal $S_{i,t}^{(0)}$ (where t indexes these proposals, $t \in \{1, \dots, T_i\}$), let $V(S_{i,t}^{(0)})$ be an indicator function, such that $V(S_{i,t}^{(0)}) = 1$ if the strategy $S_{i,t}^{(0)}$ violates any game rule (including budget constraints), and $V(S_{i,t}^{(0)}) = 0$ otherwise. The Rule Violation Rate for model M_i is:

$$\text{RVR}_i = \frac{\sum_{t=1}^{T_i} V(S_{i,t}^{(0)})}{T_i} \tag{1}$$

A lower RVR indicates better adherence to explicit constraints during initial planning phases.

B.2 Constructive Rate (CnstrR)

This metric assesses whether a correction attempt, following negative feedback, leads to an objectively improved game state, even if it doesn't immediately result in a win or full rule compliance. Let $E_{i,g,k}$ be the event that negative feedback is received for strategy $S_{i,g,k}$ (model i, game instance g, k-th strategy in that game instance). Let $A_{i,g,k+1}$ be the event that model M_i proposes a new strategy $S_{i,g,k+1}$ in response. Let $\Phi(S)$ be a game-specific state evaluation function where higher values indicate a more advantageous position for the model (e.g., based on remaining unit health/cost difference, reduced enemy threat, or other heuristic measures of game state quality). A correction

$S_{i,g,k+1}$ is considered constructive if $\Phi(S_{i,g,k+1}) > \Phi(S_{i,g,k})$. The Constructive Rate for model M_i is:

$$\text{CnstrR}_i = \frac{\sum_{g=1}^{G_i} \sum_{k=0}^{K_{i,g}-1} \mathbb{I}\left(E_{i,g,k} \wedge A_{i,g,k+1} \wedge (\Phi(S_{i,g,k+1}) > \Phi(S_{i,g,k}))\right)}{\sum_{g=1}^{G_i} \sum_{k=0}^{K_{i,g}-1} \mathbb{I}\left(E_{i,g,k} \wedge A_{i,g,k+1}\right) + \varepsilon} \tag{2}$$

where G_i is the total number of game instances involving model M_i where corrections are possible, $K_{i,g}$ is the number of strategies proposed by model M_i in game instance g, $\mathbb{I}(\cdot)$ is the indicator function, and ε is a small constant to prevent division by zero. This captures the tendency for revisions to make incremental, positive progress. Calculating $\Phi(S)$ requires domain-specific heuristics tailored to each game environment.

B.3 Multi-aspect Strategic Similarity Ratio ($\mathcal{S}_{\text{MASR}}$)

This metric assesses the similarity between a corrected strategy $S^{(k+1)}$ and the preceding strategy $S^{(k)}$ by considering multiple facets: structural, semantic, and functional similarity. For a given model M_i, let $S_{i,g,k}$ be the k-th strategy in game instance g. Let $\text{Sim}_{\text{struct}}(S', S)$, $\text{Sim}_{\text{sem}}(S', S)$, and $\text{Sim}_{\text{func}}(S', S)$ be normalized similarity scores in $[0, 1]$ for these aspects:

- **Structural Similarity ($\text{Sim}_{\text{struct}}$)**: Measures overlap in concrete elements (e.g., unit types, positions, configurations). This can be quantified using metrics like the Jaccard index on sets of chosen units/actions, or a normalized graph edit distance if strategies are represented as graphs $G(S)$. For instance, $\text{Sim}_{\text{struct}}(S', S) = 1 - \frac{\text{GED}(G(S'), G(S))}{\text{max\_GED}}$, where GED is graph edit distance.
- **Semantic Similarity (Sim_{sem})**: Measures similarity in the underlying strategic intent or concept, often derived from embeddings of textual descriptions or structured representations of the strategy. If $\mathbf{e}(S)$ is an embedding vector for strategy S, then $\text{Sim}_{\text{sem}}(S', S) = \max(0, \text{cosine\_similarity}(\mathbf{e}(S'), \mathbf{e}(S)))$.
- **Functional Similarity (Sim_{func})**: Measures overlap in the intended strategic functions or roles fulfilled by the strategy components (e.g., defensive formations, offensive pushes, resource gathering focus). If $\mathcal{F}(S)$ is the set of strategic functions embodied by strategy S, then $\text{Sim}_{\text{func}}(S', S) = \frac{|\mathcal{F}(S') \cap \mathcal{F}(S)|}{|\mathcal{F}(S') \cup \mathcal{F}(S)| + \epsilon'}$, where ϵ' prevents division by zero for empty sets.

The multi-aspect similarity ratio for a specific correction from $S_{i,g,k}$ to $S_{i,g,k+1}$ is a weighted combination:

$$\mathcal{S}_{\text{MASR}}(S_{i,g,k+1}, S_{i,g,k}) = \sum_{j=1}^{N_{\text{aspects}}} \theta_j \cdot \text{Sim}_{\text{aspect}_j}(S_{i,g,k+1}, S_{i,g,k}) \tag{3}$$

where N_{aspects} is the number of similarity aspects (e.g., 3 for structural, semantic, functional), and θ_j are weights such that $\sum_{j=1}^{N_{\text{aspects}}} \theta_j = 1$ and $\theta_j \geq 0$. The average $\bar{\mathcal{S}}_{\text{MASR}}(i)$ for model M_i is calculated over all valid correction steps:

$$\bar{\mathcal{S}}_{\text{MASR}}(i) = \frac{\sum_{g=1}^{G_i} \sum_{k=0}^{K_{i,g}-1} \mathbb{I}(E_{i,g,k} \wedge A_{i,g,k+1}) \cdot \mathcal{S}_{\text{MASR}}(S_{i,g,k+1}, S_{i,g,k})}{\sum_{g=1}^{G_i} \sum_{k=0}^{K_{i,g}-1} \mathbb{I}(E_{i,g,k} \wedge A_{i,g,k+1}) + \varepsilon} \tag{4}$$

This metric quantifies the degree of strategic preservation or alteration during revisions. A high $\bar{\mathcal{S}}_{\text{MASR}}(i)$ indicates a tendency towards conservative revision, while a low value suggests more aggressive or radical strategy changes.

B.4 First-Mover Advantage (FMA)

First-Mover Advantage (FMA) quantifies the performance difference for a model when it acts first (initiates the interaction or round) versus when it acts second (responds to the opponent's initial move). This can be calculated for various performance metrics X, such as Win Rate (WR), Over-correction Risk Rate (ORR), or Correction Success Rate (CSR). Let M_i be the model under evaluation. Let $\mathcal{G}_{i,\text{first}}$ be the set of game instances where model M_i moved first, and $\mathcal{G}_{i,\text{second}}$ be the set of game instances where model M_i moved second. Let $N_{i,\text{first}}(X) = |\mathcal{G}_{i,\text{first}}|$ and $N_{i,\text{second}}(X) = |\mathcal{G}_{i,\text{second}}|$ be the respective counts of such game instances for which metric X is applicable. Let $X_{i,m}$ be the value of metric X observed for model M_i in a specific game instance m. The average performance for model M_i on metric X when moving first is:

$$\bar{X}_{i,\text{first}} = \frac{1}{N_{i,\text{first}}(X) + \varepsilon} \sum_{m \in \mathcal{G}_{i,\text{first}}} X_{i,m} \tag{5}$$

Similarly, the average performance for model M_i on metric X when moving second is:

$$\bar{X}_{i,\text{second}} = \frac{1}{N_{i,\text{second}}(X) + \varepsilon} \sum_{m \in \mathcal{G}_{i,\text{second}}} X_{i,m} \tag{6}$$

where ε is a small positive constant to prevent division by zero if a model never plays in one of the roles or if the metric is not applicable in those instances. The First-Mover Advantage for metric X and model M_i is then defined as the difference:

$$\text{FMA}_X(i) = \bar{X}_{i,\text{first}} - \bar{X}_{i,\text{second}} \tag{7}$$

A positive $\text{FMA}_X(i)$ indicates that model M_i performs better on metric X when it has the first-move advantage. Conversely, a negative value suggests better performance when moving second. The magnitude of $\text{FMA}_X(i)$ indicates the strength of this turn-order bias.

NeurIPS Paper Checklist

1. **Claims**

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

   Answer: [Yes]

   Justification: Yes. The main claims are backed by theoretical evidence in section 3 Method and the experimental evidence in section 4 Results.

   Guidelines:

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

2. **Limitations**

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

   Answer: [Yes]

   Justification: In Section 5 paragraph Limitation, we cover three potential limitations.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

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

   Answer: [NA]

   Justification: The paper does not include theoretical results.

   Guidelines:

   - The answer NA means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.

- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

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

   Answer: [Yes]

   Justification: The benchmark is fully open source and can be easily reproduced through our codes. The link is provided in supplementary material.

   Guidelines:

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

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

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

   Answer: [Yes]

   Justification: The benchmark is fully open source and can be easily reproduced through our codes. The link is provided in supplementary material.

   Guidelines:

   - The answer NA means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.

- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

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

   Answer: [Yes]

   Justification: We provide details in Appendices and in the GitHub repository with code.

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

7. **Experiment statistical significance**

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

   Answer: [NA]

   Justification: The paper does not include error bars.

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

8. **Experiments compute resources**

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

   Answer: [Yes]

   Justification: In view 4, we provide all the LLMs tested.

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

9. **Code of ethics**

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

Answer: [Yes]

Justification: The research is conducted with the NeurIPS Code of Ethics.

Guidelines:

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

10. **Broader impacts**

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

Answer: [NA]

Justification: The paper has no negative societal impacts.

Guidelines:

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

11. **Safeguards**

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

Answer: [NA]

Justification: The paper poses no such risks.

Guidelines:

- The answer NA means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

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

Answer: [Yes]

Justification: In section References.

Guidelines:

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

13. **New assets**

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

    Answer: [Yes]

    Justification: See Appendix.

    Guidelines:

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

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

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

    Answer: [NA]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

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

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

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

    Answer: [NA]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

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

16. **Declaration of LLM usage**

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

Answer: [Yes]

Justification: Yes. The benchmark basically aims to trace LLM reasoning processes. We evaluate the output of LLMs as part of research process.

Guidelines:

- The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy (`https://neurips.cc/Conferences/2025/LLM`) for what should or should not be described.