REPTHINK: SELF-EVOLVING REPRESENTATIONS FOR LLM MINIMAX REASONING IN ADVERSARIAL GAMES

Anonymous authors
Paper under double-blind review

ABSTRACT

LLMs possess extensive knowledge of games, yet often struggle to understand concrete states and reason about adversarial consequences. We address two coupled failures: failing to identify decision-relevant relations in game states and failing to compare actions by their worst-case consequences. A correct executable simulator alone is insufficient, because accurately simulated states may still be difficult for the LLM player to interpret for decision-making. We introduce REPTHINK, which self-evolves a representation skill that transforms game states into relational evidence and explains its meaning and applicability. The LLM player reuses this skill across current and simulated states within a MAX–MIN procedure, alternating between the two players' objectives and comparing the worst examined consequences. After games, an evaluator agent diagnoses decision traces, revises the skill, and validates revisions by replaying the same positions. Across five board games, continued evolution improves performance under fixed-depth minimax reasoning. Against Stockfish limited to Elo 1320, RepThink with depth-two reasoning raises the Chess win rate of DeepSeek-V4-Flash from 12% with Base ReAct to 42%. Our theory distinguishes evidence-interpretation errors from MAX–MIN reasoning errors and derives directional bounds on their effects on decision quality.

1 INTRODUCTION

Recent agent harnesses such as Codex and Claude Code use ReAct-style reasoning–tool–feedback loops to refine decisions (Bolin, 2026; Anthropic, 2026; Yao et al., 2023). In coding and theorem proving, failed attempts can be revised using execution or verifier feedback (Ma et al., 2026). Irreversible adversarial environments such as board games instead require anticipating consequences: a committed move changes the state, and an opponent can exploit a mistake before it can be corrected. Strong general-purpose reasoning does not automatically translate into reliable play: ChessArena evaluates 13 LLMs across over 800 games and reports that none beats Maia-1100, with some even losing to random play (Liu et al., 2026a). One response is policy training: supervised fine-tuning and reinforcement learning improve chess performance (Dionisopoulos et al., 2026), and MARSHAL extends self-play training across competitive and cooperative games (Yuan et al., 2026). These approaches broaden learned capabilities but leave open how a frozen, general-purpose LLM can improve its decisions across games.

Like a human player mentally considering both sides' moves, an LLM agent can use MAX–MIN lookahead (Shoham & Leyton-Brown, 2008) as an internal ReAct-style loop of imagined actions, feedback, and revision. Game of Thoughts shows that simulated self-play can improve strategic decisions in resource-allocation and auction settings (Kempinski et al., 2025). This motivates examining how an LLM can use simulated feedback in sequential games, where each candidate action produces a new state whose decision-relevant relations must be understood. Code World Models for General Game Playing translates rules and trajectories into executable Python environment models, then delegates decisions to MCTS and other planners (Lehrach et al., 2026). Yet accurate simulation alone does not make planning effective: finite search budgets may leave decisive continuations unexplored, and search quality remains sensitive to rollout policies and value estimates. This motivates

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

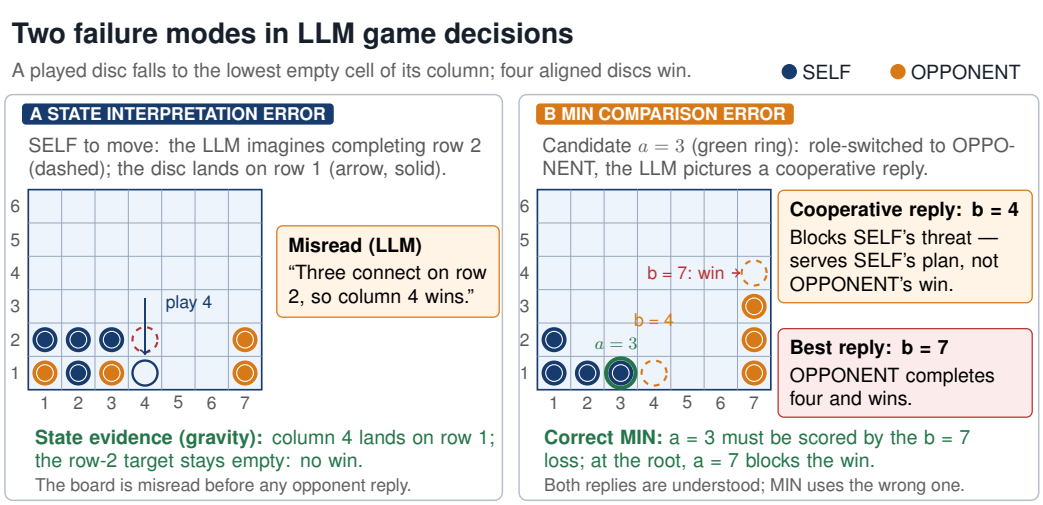


Figure 1: **Two decision failures in Connect-Four.** A played disc falls to the lowest empty cell of its column; four aligned discs win. Rows count from the bottom; a, b label actions, not players. **Left (state-interpretation error):** seeing three blue discs connected on row 2, the LLM reads column 4 as a winning move, imagining the disc completing row 2 (dashed). Gravity instead lands the disc at the column's lowest empty cell, row 1 (arrow tip, solid), so the imagined completion stays empty—the board's relations are misread. **Right (MIN-comparison error):** the green ring marks the candidate action $a = 3$ under evaluation. The LLM pictures the reply $b = 4$ that cooperates with its own plan by blocking SELF's threat, instead of OPPONENT's best reply $b = 7$, which completes four orange discs and wins immediately. Scoring a by the cooperative reply substitutes a favorable reply for the worst case, violating MIN.

using the LLM's game knowledge directly—interpreting state relations, selecting relevant continuations, and comparing their consequences; the challenge is to make executable representations usable by the LLM's own adversarial reasoning.

Our argument follows a dependency chain. Adversarial lookahead must account for an opponent pursuing its own objective, yet ReAct-style deliberation can produce cooperative continuations: the agent selects opponent replies that cooperate with its own plan rather than challenge it. We therefore use MAX–MIN role-play to alternate player perspectives and compare actions by their worst examined consequences. Such reasoning, in turn, depends on understanding both current and simulated states: a correctly simulated board is of little use if the LLM does not recognize the decisive relations within it or misinterprets their consequences. We therefore evolve the state evidence available to the LLM—how relations are exposed and explained—using feedback from its decisions, so that representation evolution supports the fixed adversarial reasoning procedure. Figure 1 illustrates both failures in the game Connect-Four: a misread landing cell creates an imaginary win, while a cooperative opponent reply hides an immediate loss.

We introduce REPTHINK, combining *representation* and *thinking*: an evolving representation skill supports a fixed minimax procedure. Python scripts expose relations in the current and imagined game states; Markdown explains their meaning and use. The Game-Agent examines its own moves and the opponent's replies, then compares the worst examined consequences. A separate Evaluator Agent diagnoses game traces, revises the skill, and checks the revised Game-Agent on the same positions. Game rules, model weights, and the reasoning procedure remain fixed.

Our contributions are:

1. **Executable representations for the LLM as decision maker:** we study optimizing code-based state evidence for a frozen LLM's own adversarial reasoning, complementing code world models consumed by external planners. RepThink evolves this representation through trajectory diagnosis and replay within fixed minimax, and tests whether the resulting LLM policy can defeat MCTS opponents.

2. **Theoretical decision-error analysis:** we separate preserved information from LLM interpretation, bound the effects of omitted moves and incorrect MAX–MIN comparisons, and give sufficient conditions for a skill revision to correct a decision.
3. **Cross-game evaluation:** continued evolution improves mean score in nine of ten settings. MiniMax Chess rises from 27.0 to 89.7 ± 6.4 against MCTS-300; against Stockfish at Elo 1320, DeepSeek Chess win rate rises from 12% with Base ReAct to 42% with RepThink. Ablations test representation and lookahead separately.

2 Related Work

LLM reasoning and planning in games. Supervised learning across eight card games studies transfer across game tasks (Wang et al., 2025). ThoughtSculpt uses MCTS to revise candidate solutions in generation and puzzle tasks (Chi et al., 2025); Schultz et al. (2025) trains LLMs for external-search rollouts and values or internal search-tree generation. In repeated games, Reflective Hypothetical Mind maintains opponent-strategy hypotheses and adapts decisions to updated beliefs (Liu et al., 2026b). RepThink keeps model weights fixed and lets the Game-Agent select branches and compare represented consequences under MAX–MIN; it does not predict an opponent's actual strategy.

State evidence and its use in decisions. PokerSkill combines a deterministic context engine with an expert-designed poker skill library and action constraints, demonstrating the value of structured interfaces for untrained LLM decision makers (Li et al., 2026). RepThink evolves state evidence and its usage guidance from game feedback, reusing the skill across current and counterfactual states without code selecting or ranking actions. The observation–belief–action distinction (Sobotka et al., 2026) complements our focus on external evidence and its use in adversarial comparison.

Self-evolving agents and skills. Gao et al. (2026) organizes self-evolution around models, context, tools, and architecture. Many approaches share a feedback–revision–evaluation loop, making *what evolves* a useful basis for comparison. Reflexion stores verbal reflections on task feedback in episodic memory (Shinn et al., 2023). Trace2Skill uses success and error analysts to distill successful and failed trajectories into consolidated skills (Ni et al., 2026). GEPA optimizes prompts (Agrawal et al., 2026); Voyager accumulates executable action skills (Wang et al., 2024). AutoHarness synthesizes the code surrounding an agent's actions (Lou et al., 2026), while Hyperagents edits both the task agent and its self-improvement procedure (Zhang et al., 2026a). Observation and world-model interfaces offer another target: TACO and ACON optimize observation filtering and compression (Ren et al., 2026; Kang et al., 2026), WorldEvolver revises predictive knowledge (Zhang et al., 2026b), and WorldCoder and OPINE-World synthesize and repair executable environment dynamics (Tang et al., 2024; Courtis et al., 2026). RepThink instead evolves the representation of known game dynamics for a frozen LLM to use directly.

3 Preliminaries

Game and objective. Following the perfect-information extensive-form setting of Shoham & Leyton-Brown (2008), we consider a finite, deterministic, alternating, two-player zero-sum game. Its state space is $\mathcal{S}$, initial state s_0, and terminal-state set $\mathcal{Z}$. A state records the board, acting player, and all rule-relevant history. At a nonterminal state s, $\mathcal{A}(s)$ is the complete set of legal actions, $T(s, a)$ is the successor after action a, and $p(s) \in \{\mathrm{MAX}, \mathrm{MIN}\}$ identifies the acting player. Both players observe the full state and rules. MAX denotes the Game-Agent; MIN denotes its opponent. Terminal utility $u : \mathcal{Z} \to [-1, 1]$ is always measured for MAX: larger is better for the Game-Agent and worse for its opponent.

Minimax and action quality. The true continuation value $V^*(s)$ is the outcome value under optimal play from state s:

$$V^*(s) = \begin{cases} u(s), & s \in \mathcal{Z}, \\ \max_{a \in \mathcal{A}(s)} V^*(T(s, a)), & p(s) = \mathrm{MAX}, \\ \min_{a \in \mathcal{A}(s)} V^*(T(s, a)), & p(s) = \mathrm{MIN}. \end{cases} \tag{1}$$

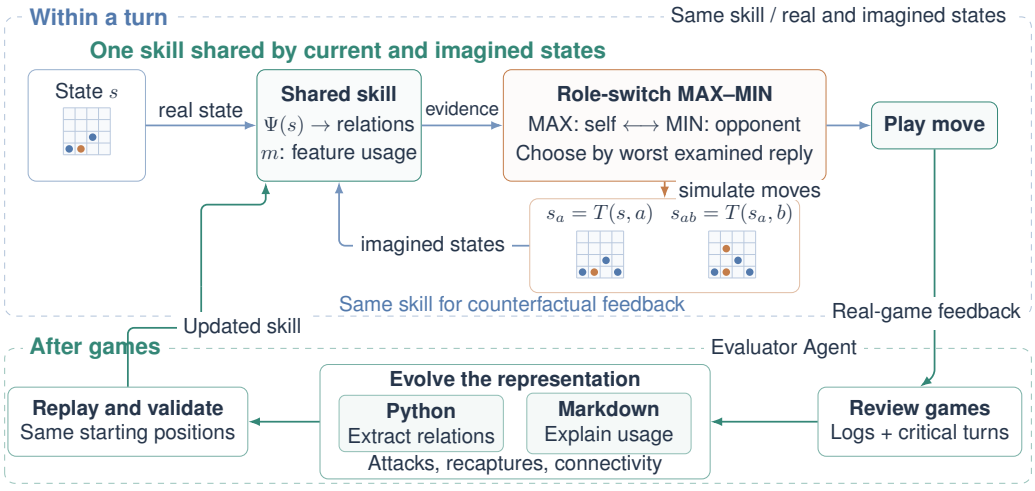


Figure 2: **Shared representation, two feedback timescales.** "Shared" means that within a turn, one skill $H = (\Psi, m)$ serves both the Game-Agent's reads of the current state and its reads of the imagined states after actions a, b, under both the MAX and MIN role perspectives. The LLM proposes branches, interprets relations, and compares outcomes under MAX–MIN before acting. After games, the Evaluator revises Python evidence and Markdown guidance and checks the changes by same-position replay. Model weights, rules, depth, and the reasoning procedure remain fixed. Boards are schematic.

At a Game-Agent decision state $\bar{s}$, write $\mathcal{A} = \mathcal{A}(\bar{s})$ and $Q(a) = V^*(T(\bar{s}, a))$. The regret of its chosen action $\hat{a}$ is

$$R = \max_{a \in \mathcal{A}} Q(a) - Q(\hat{a}). \tag{2}$$

These true values define decision quality for analysis; the agent is not given them.

Finite lookahead. Depth D counts legal actions from $\bar{s}$, not tool calls. At depth two, action a leads to $s_a = T(\bar{s}, a)$ and an opponent reply b leads to $s_{ab} = T(s_a, b)$. Here a, b are actions; MAX and MIN are players. The agent must judge nonterminal leaves because it does not solve the full game. Actual terminal states end a branch early. All legal moves are eligible for selection; receiving their complete list does not establish that every continuation has been evaluated.

4 Method

To improve decisions in adversarial perfect-information games, we propose RepThink, which uses counterfactual feedback before committing a move and real-game feedback to revise its representation skill afterward (Figure 2).

4.1 Shared Representation and Frozen Game-Agent

The evolving skill at round k is $H_k = (\Psi_k, m_k)$. Python Ψ_k maps a state to relational evidence, such as attacks, defenses, material changes, or connectivity; Markdown m_k explains the evidence's meaning, conditions, and use. The fixed engine supplies legal actions and exact transitions. The frozen Game-Agent, with model parameters θ, chooses which continuations to examine, interprets their evidence, and selects the committed move. The skill is *shared* in two senses: the same H_k represents the current state and every imagined successor, and the MAX and MIN role perspectives both interpret evidence from that same H_k; a skill version remains fixed throughout a game. Appendix D lists the interfaces, per-game relations, and editable components.

4.2 Fixed Minimax over Represented States

At each imagined node, the prompt asks the Game-Agent to consider the acting player's objective: prefer higher Game-Agent value at MAX nodes and lower Game-Agent value at MIN nodes. Values retain the same perspective when the acting player changes. Python executes requested legal lines and exposes state relations; it does not rank actions, assign overall utility, or call MCTS or Stockfish to construct evidence. Value judgment and comparison remain with the LLM.

At $D = 2$, the Game-Agent looks ahead for two plies: its own action a, followed by an opponent reply b. Unless the game has ended, the resulting leaf is a state in which the Game-Agent moves next; its next action is not expanded. The intended rule is to evaluate these leaves, score each considered action by its worst examined reply, and select the action with the best such score. A favorable reply cannot replace an examined losing reply, even if both leaves are understood correctly (Figure 1, right). At $D = 0$, the skill supplies current-state evidence without prescribed lookahead. At $D = 4$, the sequence is agent–opponent–agent–opponent, again ending with the agent to move unless the game ends earlier. Depth is a configuration parameter, not an evolution target. Appendix A.2 contrasts this LLM-controlled procedure with an explicit evaluator consumed by a search program.

Integrating representation and MAX–MIN thinking. The two components are fused at every lookahead node: the skill converts each state—current or imagined—into relational evidence, and the role-switched MAX–MIN comparison operates directly on that evidence. This internal loop reconstructs environment feedback before the irreversible move: the engine supplies exact transitions (the rule-determined part), while the MIN role internalizes the opponent half, answering each candidate with its worst examined reply rather than a cooperative one. The skill exposes threats and consequences but never ranks actions; evolution repairs exactly the coupling points where decisions fail—a missed reply, a misread relation, or a discarded worst case.

4.3 Environment-Representation Self-Evolution

Each evolution round begins with a batch of games played under the same fixed skill. An Evaluator Agent then reviews their outcomes and decision traces, identifies a failure mechanism, revises H, and replays the Game-Agent from the same saved positions. Python changes can expose missing relations; Markdown changes can clarify their interpretation and use under the fixed MAX–MIN rule. Replay checks whether the Game-Agent finds the relevant reply, understands its consequences, and uses it in the comparison. Earlier successful positions provide regression checks. The accepted skill is frozen for subsequent games; final evaluation games do not update it. Appendix E gives the complete procedure.

5 Theoretical Analysis

We analyze completed legal decisions for a fixed realization of the Game-Agent's randomness. Three questions guide the analysis: does the evidence preserve the relevant information, does the LLM interpret it correctly, and does its comparison follow MAX–MIN? Detailed proofs and finite-depth extensions are in Appendix A.

Information preservation and LLM interpretation. Let x range over a finite set of leaf-decision contexts, each recording the state and the Game-Agent's perspective, with true continuation value $U(x)$. Let $Z_H(x)$ contain all evidence available to the LLM, including the raw board, rules, history, and skill output. Write its scalar value judgment as $\widehat{U}(x) = f_\theta(Z_H(x))$, where f_θ denotes the frozen LLM's interpretation of that evidence. For each evidence value z, define $h_H(z)$ as the midpoint between the largest and smallest true values among contexts with $Z_H(x) = z$. This reference decoder is an analysis device, not a tool available to the agent. Define

$$\delta_H = \max_x |U(x) - h_H(Z_H(x))|, \qquad \kappa_{\theta,H} = \max_x |\widehat{U}(x) - h_H(Z_H(x))|. \tag{3}$$

Here δ_H measures information lost by the evidence, while $\kappa_{\theta,H}$ measures the LLM's deviation from the reference interpretation.

Proposition 5.1 (Information and interpretation error)**.** *The smallest uniform error attainable by any decoder of Z_H is δ_H. The Game-Agent's judgment error satisfies* $\max_x |\widehat{U}(x) - U(x)| \leq$

$\delta_H + \kappa_{\theta,H}$. *Adding information without removing existing distinctions cannot increase* δ_H*, but need not decrease* $\kappa_{\theta,H}$.

When the full state, rules, and relevant history determine U, the evidence is lossless and $\delta_H = 0$. Explicit attack and defense relations then add no information in this formal sense; they make existing information easier for the bounded LLM to use. RepThink therefore need not reduce δ_H: its benefit can come from better interpretation and branch discovery. Moreover, $\kappa_{\theta,H}$ includes both misunderstanding a state and misestimating its continuation value. An LLM can read a two-ply leaf correctly yet misjudge consequences beyond the horizon. Neither exact simulation nor lossless evidence guarantees a small judgment error.

From leaf judgments to the selected action. At the fixed root $\bar{s}$, define $\mathcal{B}(a) = \mathcal{A}(s_a)$ and $U(a,b) = V^*(s_{ab})$, so $Q(a) = \min_{b\in\mathcal{B}(a)} U(a,b)$. A terminal action has a dummy reply carrying its terminal utility. All legal moves belong to $\mathcal{A}$; let $C \subseteq \mathcal{A}$ be the nonempty set actually evaluated and compared, and $\mathcal{R}(a) \subseteq \mathcal{B}(a)$ the nonempty set of examined replies for each $a \in C$. If every legal move is evaluated, $C = \mathcal{A}$. Define

$$\alpha = \max_{a\in\mathcal{A}} Q(a) - \max_{a\in C} Q(a), \qquad \beta_a = \min_{b\in\mathcal{R}(a)} U(a,b) - Q(a). \tag{4}$$

The first term measures value lost by omitting good actions ($\alpha = 0$ when $C = \mathcal{A}$); the second measures optimism from omitting adverse replies.

The function $\widehat{U}(a,b)$ assigns the LLM's judgment to an examined leaf; it is not a value supplied by the representation code. Let $\hat{b}_a \in \mathcal{R}(a)$ be the reply the LLM uses to score action a, and write that score as $J(a) = \widehat{U}(a,\hat{b}_a)$. For final action $\hat{a} \in C$, define the nonnegative comparison errors

$$\eta_a = J(a) - \min_{b\in\mathcal{R}(a)} \widehat{U}(a,b), \qquad \eta_{\mathrm{MAX}} = \max_{a\in C} J(a) - J(\hat{a}). \tag{5}$$

Correct MIN makes $\eta_a = 0$; correct final MAX makes $\eta_{\mathrm{MAX}} = 0$. These errors distinguish misunderstanding a leaf from understanding it but using the wrong reply or selecting inconsistently. Scalar judgments must be specified independently of the selected action; prose-only traces do not identify their numerical errors.

Theorem 5.2 (Decision error of the Game-Agent)**.** *If each examined leaf satisfies* $|\widehat{U}(a,b) - U(a,b)| \le \epsilon$*, then*

$$R \le \alpha + \beta_{\hat{a}} + 2\epsilon + \eta_{\hat{a}} + \eta_{\mathrm{MAX}}. \tag{6}$$

By Proposition 5.1, one may take $\epsilon = \delta_H + \kappa_{\theta,H}$. The bound therefore separates omission, interpretation, and comparison failures. Figure 1, right, illustrates a positive MIN error even when both leaf outcomes are correctly understood. Exact MAX/MIN does not amplify a uniform leaf error merely by increasing depth; selective deeper reasoning can still omit replies or miscompare outcomes (Appendix A.5).

When a representation revision corrects a decision. Choose a target value τ. An action is "good enough" precisely when $Q(a) \ge \tau$; this is a condition on its true consequences, not on whether the LLM already knows it is good. After a revision, use C' and $\mathcal{R}'(a)$ for the evaluated actions and replies. Assume leaf error at most ϵ, MIN error at most $\bar{\eta}_{\min}$, and final MAX error at most $\bar{\eta}_{\max}$.

Theorem 5.3 (Decision correction by sufficient refutations)**.** *Suppose* C' *contains an action* g *with* $Q(g) \ge \tau$*. If every* $a \in C'$ *with* $Q(a) < \tau$ *has an examined reply* b_a *such that*

$$U(a,b_a) < \tau - 2\epsilon - \bar{\eta}_{\min} - \bar{\eta}_{\max}, \tag{7}$$

then the revised choice $\hat{a}'$ *satisfies* $Q(\hat{a}') \ge \tau$*. Taking* $\tau = Q(\hat{a}) + \gamma$ *for* $\gamma > 0$ *gives a regret reduction of at least* γ*, whenever these conditions hold.*

If $C' = \mathcal{A}$ and τ is attainable, the good-action condition is automatic. The other condition says that each inferior action has a revealed consequence worse than the target by more than the combined judgment and comparison errors. The LLM can then reject it without perfectly valuing every leaf. A revision must make the counterreply discoverable, its consequence understandable, and its loss effective in MIN comparison. The theorem does not guarantee that a code or guidance edit achieves these conditions; same-position replay tests the predicted behavior. This also explains why large-margin mistakes may be easier to exclude than to distinguish near-equal actions, once the relevant replies are found.

6 EXPERIMENTS

We test continued evolution at fixed depth, ablate representation and lookahead, and analyze Chess decision errors.

6.1 SELF-EVOLUTION ACROSS FIVE GAMES AND TWO MODEL–OPPONENT SETTINGS

Setup. We evaluate five games—Tic-Tac-Toe, Connect-Four, Generalized Tic-Tac-Toe, Hex-7, and Chess—with MiniMax-M2.5 (a general-purpose LLM, unrelated to the minimax procedure) against MCTS-300 and DeepSeek-V4-Flash against MCTS-1000 (Monte Carlo tree search with the indicated simulation budgets). MCTS anchors a comparison with the code-world-model planning route of Lehrach et al. (2026)—*can an LLM using executable representations compete with search that consumes code world models?*—rather than representing the strongest available planners. They use 300/1,000 simulations per move; Chess and Generalized-TTT use heuristic-guided rollouts, the other games uniform-random rollouts (Appendix D.2). This matches the research question rather than the exact implementation or compute budget of Lehrach et al. (2026).

GPT-5.4 is the Evaluator Agent for diagnosis, skill revision, and replay orchestration; MiniMax-M2.5 or DeepSeek-V4-Flash is the frozen Game-Agent. This follows GEPA's separation of reflection and task models, whose guidance recommends a capable reflection model while retaining the deployment model for task execution (Agrawal et al., 2026; GEPA Authors, 2026). Only the Game-Agent plays evaluation games, using the frozen skill. In every LLM condition, the Game-Agent moves first (White in Chess), including evolution games, ablations, and evaluation against Stockfish.

We first define the two compared checkpoints. One *evolution round* consists of five games played under a frozen skill plus one Evaluator diagnosis–revision–replay cycle (Section 4.3). *Round 1* denotes the checkpoint after the first such round, evaluated on 100 games; it is not an unevolved baseline. *Round 10* denotes the checkpoint after nine further rounds continuing from that same round-1 checkpoint (50 evolution games in total); per-run scores appear in Appendix F. Experiment 1 compares these two checkpoints under the same fixed depth-two minimax procedure: for each game–configuration setting, three independent continuations start from the same round-1 checkpoint, and each final artifact is evaluated on 100 games (300 evaluation games per setting). We report the round-1 reference score and the mean and sample standard deviation of the three round-10 scores. Score is $100(N_W + 0.5N_D)/(N_W + N_D + N_L)$, where N_W, N_D, N_L denote win, draw, and loss counts.

Baselines and evolution targets. Table 1 compares evolution targets. Base ReAct does not evolve. Reflection, our adaptation of Reflexion, stores verbal lessons from failed attempts in Markdown (Shinn et al., 2023); Trace2Skill distills successful and failed trajectories through success/error analysis and skill consolidation, here into Markdown (Ni et al., 2026). Our AutoHarness adaptation evolves next-action constraints (Lou et al., 2026): rejecting illegal moves, taking available one-move wins, and preventing avoidable opponent one-move wins—converting frequent LLM mistakes into executable safeguards. Reflection, Trace2Skill, and AutoHarness each undergo 50 evolution updates before evaluation. RepThink evolves state evidence and its usage documentation, leaving action selection to the LLM. Methods share models and opponents but differ in tools, editable artifacts, and update counts; comparing rounds 1 and 10 tests further evolution at fixed depth.

Results. Mean round-10 score exceeds round 1 in nine of ten settings; MiniMax Tic-Tac-Toe is the exception (47.5 to 47.0). Of 30 individual run scores, 28 exceed their setting's reference, one ties it, and one falls below. Equally weighting games, the mean gain is 26.4 points for MiniMax and 14.8 for DeepSeek. Round-10 means exceed every listed baseline in seven settings, tie the strongest baseline in DeepSeek Generalized-TTT, and fall below AutoHarness in MiniMax Tic-Tac-Toe and DeepSeek Connect-Four. MiniMax Chess has the largest gain, from 27.0 to 89.7 ± 6.4. Gains recur across settings but vary by game and evolution run; these are descriptive summaries, not independent-trial significance tests.

Reflection improves MiniMax Tic-Tac-Toe, Hex-7, and DeepSeek Chess but not Connect-Four, Generalized-TTT, or MiniMax Chess; Trace2Skill is similarly uneven, scoring 2.0 (MiniMax) and 5.0 (DeepSeek) in Connect-Four and 0.0 in MiniMax Chess. Textual experience alone thus provides

Table 1: Experiment 1: scores on a 0–100 scale (win $= 1$, draw $= 0.5$). Both RepThink checkpoints use $D = 2$. Round 10 reports mean $\pm$ sample SD across three independent continuations from the same round-1 checkpoint; individual scores appear in Appendix F.

Game	Base ReAct	Reflection	Trace2Skill	AutoHarness	RepThink round 1	RepThink round 10
MiniMax-M2.5 vs. MCTS-300						
Tic-Tac-Toe	20.5	40.0	47.5	50.5	47.5	47.0 ± 1.7
Connect-Four	0.0	0.0	2.0	20.0	6.0	23.8 ± 6.8
Hex-7	36.0	45.0	66.0	61.0	53.0	75.0 ± 5.6
Generalized-TTT	1.5	1.0	12.0	31.0	16.0	46.0 ± 3.0
Chess	2.0	0.0	0.0	9.0	27.0	89.7 ± 6.4
DeepSeek-V4-Flash vs. MCTS-1000						
Tic-Tac-Toe	47.5	47.0	49.5	49.0	46.0	49.8 ± 0.3
Connect-Four	0.0	0.0	5.0	13.0	7.0	12.8 ± 5.3
Hex-7	46.0	57.0	51.0	66.0	39.0	70.3 ± 5.0
Generalized-TTT	44.0	48.0	50.0	47.5	43.0	50.0 ± 3.8
Chess	72.0	86.0	89.0	81.0	72.0	98.0 ± 2.0

Table 2: MiniMax-M2.5 ablation against MCTS-300. Entries are scores (win $= 1$, draw $= 0.5$) on a 0–100 scale over 100 games per condition. Representation-only ($D = 0$) uses the evolved skill without prescribed lookahead. The depth-two row reports the ablated artifact's score and therefore differs from the three-run means in Table 1.

Method	Tic-Tac-Toe	Connect-Four	Generalized-TTT	Hex-7	Chess
Base ReAct	20.5	0.0	1.5	36.0	2.0
Language-only minimax ($D = 2$)	27.5	0.0	5.0	34.0	4.0
Representation only ($D = 0$)	49.5	14.5	41.0	83.0	67.0
RepThink $D = 1$	49.0	21.0	39.5	82.0	88.0
RepThink $D = 2$	48.0	30.5	49.0	80.0	97.0
RepThink $D = 3$	46.5	21.0	40.0	89.0	92.0
RepThink $D = 4$	49.0	26.0	29.5	96.0	86.0

uneven gains; RepThink instead computes relations for each encountered and imagined state rather than relying on a document to encode every state-dependent contingency.

Tic-Tac-Toe and our 5×5, four-in-a-row Generalized-TTT are perfect-play draws (Uiterwijk, 2019); their round-10 scores (47.0/49.8 and 46.0/50.0) approach the draw-equivalent 50, and score alone does not identify the draw rate. Of the other games, RepThink scores above 50 in Hex-7 and Chess under both configurations but below 50 in Connect-Four; its largest advantage is in Chess (89.7 and 98.0), showing that executable evidence lets the LLM outperform the tested MCTS budgets and heuristics.

6.2 Ablating Environment Representation and Lookahead

With MiniMax-M2.5 against MCTS-300, we hold each game's round-10 skill artifact fixed across ablations. Removing the skill leaves language-only minimax at $D = 2$; removing prescribed lookahead leaves representation-only ($D = 0$, including Markdown evidence-use guidance), which still permits spontaneous lookahead. Depths one through four vary the imagined horizon.

Language-only minimax improves Tic-Tac-Toe (20.5 to 27.5) but changes the larger games little: Connect-Four stays at zero, Hex-7 declines, and Generalized-TTT and Chess gain only 3.5 and 2.0 points. Representation-only substantially improves all five games. Adding depth-two lookahead raises Connect-Four from 14.5 to 30.5, Generalized-TTT from 41.0 to 49.0, and Chess from 67.0 to 97.0, but depth is not uniformly beneficial: Hex-7 falls from 83.0 ($D = 0$) to 80.0 ($D = 2$) before reaching 96.0 ($D = 4$), and Chess peaks at $D = 2$.

Deeper reasoning introduces more opportunities for branch omission and incorrect MAX/MIN comparison, and more evidence and intermediate conclusions to integrate: during aggregation the LLM may omit adverse evidence, mix branches, or revise earlier judgments without justification. Appendix A.5 accounts for omission and comparison errors across levels and allows aggregation to depart from the selected child's value; exact MAX/MIN alone does not amplify a fixed uniform leaf error, so the nonmonotonic scores are consistent with these failures but do not identify which increased.

6.3 Chess against Stockfish 1320

To test beyond MCTS, we evaluate DeepSeek-V4-Flash against Stockfish limited to Elo 1320 (Table 3), alongside MCTS-1000 and MCTS-2000, using 100 games per condition. An unrestricted Stockfish at depth 14 audits move quality via average centipawn loss (ACPL) and error rates (cp denotes centipawns); it supplies no evidence to the acting agent.

Table 3: Chess against Stockfish at Elo 1320. All conditions use 100 games, LLM conditions use DeepSeek-V4-Flash, and all agents move first (White); rates are percentages and audits use depth 14. Full metrics and protocol: Appendix B.

Metric	Base ReAct	RepThink $D = 2$	MCTS-1000	MCTS-2000
Win rate	12%	42%	2%	15%
ACPL (uncapped)	129.0	83.6	901.3	586.6
ACPL (cap 1000)	76.1	68.9	169.8	127.9
Severe error ≥ 500 cp	1.9%	1.2%	9.8%	6.6%
Gift rate	4.1%	0.8%	4.6%	3.2%

RepThink reduces several measures of costly tactical error. The gift rate—moves after which the opponent actually captures a piece worth at least three units at a net material loss—falls from 4.1% to 0.8% (a relative reduction of about 80%), and single-move losses of at least 500 cp fall from 1.9% to 1.2%. The mean excess loss beyond 1,000 cp, $\mathbb{E}[(\mathrm{CPL} - 1{,}000)_+]$, falls from 52.9 to 14.7 cp, accounting for about 84.1% of the uncapped ACPL reduction (129.0 to 83.6); it counts only the portion above 1,000 cp. Errors of at least 100 cp remain frequent (21.9% versus 20.8%), and capped ACPL improves from 76.1 to 68.9. The higher win rate accompanies fewer severe errors, consistent with improved avoidance of costly tactical mistakes, though these aggregate diagnostics do not establish how much of the win-rate gain each change causes.

MCTS-1000 wins 2 of 100 games and MCTS-2000 wins 15, with substantially higher move losses and more frequent severe errors than the LLM conditions; increasing the budget helps, but substantial errors remain. This characterizes the tested budgets and rollout policies, not an advantage over search generally.

7 Conclusion

We propose RepThink, which evolves executable state evidence and its usage guidance for a frozen LLM performing fixed-depth adversarial reasoning, reconstructing environment feedback before the irreversible move: exact transitions from the engine and worst-case opponent replies from MIN role-play, read through the same representation. Our analysis separates information preservation, LLM interpretation, omitted alternatives, and incorrect MAX–MIN execution, and gives conditions under which revised evidence corrects a decision. Across five games, continued evolution improves mean score in nine of ten settings, and ablations show that executable evidence and lookahead contribute differently across games. Against Stockfish at Elo 1320, DeepSeek's win rate rises from 12% to 42% with fewer severe errors. These findings motivate evaluating representations by how reliably an agent uses them before committing.

Beyond board games. The framework could also support decisions in negotiation, attack–defense, and competitive scheduling, where executable representations expose constraints and consequences of proposed actions while an LLM compares responses before committing; subsequent feedback would guide revisions. These applications remain untested, and hidden information and general-sum incentives require appropriate objectives.

AI USE STATEMENT

Generative AI tools assisted with conceptual framing, experimental design, implementation, trace analysis, interpretation of results, mathematical exposition, literature review, and manuscript and figure preparation. Separately, the experimental method uses LLMs as the Game-Agent and Evaluator Agent, as described in Section 4.1: the Evaluator Agent autonomously diagnoses trajectories, revises Python state evidence and its Markdown documentation, and validates revisions. The reported evaluation uses artifacts frozen after this LLM self-evolution phase. The authors are responsible for verifying the final claims, code, references, and reported results.

REFERENCES

Lakshya A. Agrawal, Shangyin Tan, Dilara Soylu, Noah Ziems, Rishi Khare, Krista Opsahl-Ong, Arnav Singhvi, Herumb Shandilya, Michael J. Ryan, Meng Jiang, Christopher Potts, Koushik Sen, Alexandros G. Dimakis, Ion Stoica, Dan Klein, Matei Zaharia, and Omar Khattab. GEPA: Reflective prompt evolution can outperform reinforcement learning. In *International Conference on Learning Representations*, 2026. URL https://arxiv.org/abs/2507.19457.

L. Victor Allis, Maarten van der Meulen, and H. Jaap van den Herik. Proof-number search. *Artificial Intelligence*, 66(1):91–124, 1994. doi: 10.1016/0004-3702(94)90004-3.

Anthropic. How the agent loop works. Claude Code Documentation, 2026. URL https://code.claude.com/docs/en/agent-sdk/agent-loop. Accessed September 16, 2026.

Michael Bolin. Unrolling the Codex agent loop. OpenAI Engineering, 2026. URL https://openai.com/index/unrolling-the-codex-agent-loop/.

Yizhou Chi, Kevin Yang, and Dan Klein. ThoughtSculpt: Reasoning with intermediate revision and search. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pp. 7700–7726, 2025. doi: 10.18653/v1/2025.findings-naacl.428. URL https://aclanthology.org/2025.findings-naacl.428/.

David Courtis, Wenhao Li, and Scott Sanner. OPINE-World: Programmatic world modeling with ontology-error-prioritized interactive exploration for ARC-AGI-3. *arXiv preprint arXiv:2607.01531*, 2026. URL https://arxiv.org/abs/2607.01531.

Lucas Dionisopoulos, Nicklas Majamaki, and Prithviraj Ammanabrolu. How reasoning evolves from post-training data: An empirical study using chess. In *International Conference on Machine Learning*, 2026. URL https://arxiv.org/abs/2604.05134.

Huan-ang Gao, Jiayi Geng, Wenyue Hua, Mengkang Hu, Xinzhe Juan, Hongzhang Liu, Shilong Liu, Jiahao Qiu, Xuan Qi, Yiran Wu, Hongru Wang, Han Xiao, Yuhang Zhou, Shaokun Zhang, Jiayi Zhang, Jinyu Xiang, Yixiong Fang, Qiwen Zhao, Dongrui Liu, Qihan Ren, Cheng Qian, Zhenhailong Wang, Minda Hu, Huazheng Wang, Qingyun Wu, Heng Ji, and Mengdi Wang. A survey of self-evolving agents: What, when, how, and where to evolve on the path to artificial super intelligence. *Transactions on Machine Learning Research*, 2026. URL https://openreview.net/forum?id=CTr3bovS5F.

GEPA Authors. GEPA documentation: Models and performance. Official documentation, 2026. URL https://gepa-ai.github.io/gepa/guides/faq/. Accessed September 17, 2026.

Christopher Grimm, André Barreto, Satinder Singh, and David Silver. The value equivalence principle for model-based reinforcement learning. In *Advances in Neural Information Processing Systems*, 2020. URL https://arxiv.org/abs/2011.03506.

Minki Kang, Wei-Ning Chen, Dongge Han, Huseyin A. Inan, Lukas Wutschitz, Yanzhi Chen, Robert Sim, and Saravan Rajmohan. ACON: Optimizing context compression for long-horizon LLM agents. In *International Conference on Machine Learning*, 2026. URL https://arxiv.org/abs/2510.00615.

Benjamin Kempinski, Ian Gemp, Kate Larson, Marc Lanctot, Yoram Bachrach, and Tal Kachman. Game of thoughts: Iterative reasoning in game-theoretic domains with large language models. In *Proceedings of the 24th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pp. 1088–1097, 2025.

Christian Kroer and Tuomas Sandholm. A unified framework for extensive-form game abstraction with bounds. In *Advances in Neural Information Processing Systems*, volume 31, 2018. URL https://proceedings.neurips.cc/paper/2018/hash/aa942ab2bfa6ebda4840e7360ce6e7ef-Abstract.html.

Wolfgang Lehrach, Daniel Hennes, Miguel Lazaro-Gredilla, Xinghua Lou, Carter Wendelken, Zun Li, Antoine Dedieu, Marc Lanctot, Atil Iscen, John Schultz, Marcus Chiam, Ian Gemp, Piotr Zielinski, Satinder Singh, and Kevin Murphy. Code world models for general game playing. In *International Conference on Learning Representations*, volume 2026, pp. 133870–133927, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/d8a12fde9e72444e1b356e8c37e53753-Paper-Conference.pdf.

Boning Li, Baoxiang Wang, and Longbo Huang. PokerSkill: LLMs can play expert-level poker without training or solvers. *arXiv preprint arXiv:2605.30094*, 2026.

Jincheng Liu, Sijun He, Jingjing Wu, Xiangsen Wang, Yang Chen, Zhaoqi Kuang, Siqi Bao, and Yuan Yao. ChessArena: A chess testbed for evaluating strategic reasoning capabilities of large language models. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 7901–7954, 2026a. doi: 10.18653/v1/2026.acl-long.360.

Naming Liu, Youzhi Zhang, and Ying Wen. Solving repeated games with large language model. In *Proceedings of the 25th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, 2026b.

Xinghua Lou, Miguel Lázaro-Gredilla, Antoine Dedieu, Carter Wendelken, Wolfgang Lehrach, and Kevin P. Murphy. AutoHarness: Improving LLM agents by automatically synthesizing a code harness. *arXiv preprint arXiv:2603.03329*, 2026. URL https://arxiv.org/abs/2603.03329.

David Ma, Kaijing Ma, Shawn Guo, Yunfeng Shi, Enduo Zhao, Jiajun Shi, Zhaoxiang Zhang, Gavin Cheung, Jiaheng Liu, and Zili Wang. OProver: A unified framework for agentic formal theorem proving. *arXiv preprint arXiv:2605.17283*, 2026.

Jingwei Ni, Yihao Liu, Xinpeng Liu, Yutao Sun, Mengyu Zhou, Pengyu Cheng, Dexin Wang, Erchao Zhao, Xiaoxi Jiang, and Guanjun Jiang. Trace2Skill: Distill trajectory-local lessons into transferable agent skills. arXiv preprint arXiv:2603.25158, 2026. URL https://arxiv.org/abs/2603.25158.

Jincheng Ren, Siwei Wu, Yizhi Li, Kang Zhu, Shu Xu, Boyu Feng, Ruibin Yuan, Wei Zhang, Riza Batista-Navarro, Jian Yang, and Chenghua Lin. A self-evolving framework for efficient terminal agents via observational context compression. *arXiv preprint arXiv:2604.19572*, 2026.

John Schultz, Jakub Adamek, Matej Jusup, Marc Lanctot, Michael Kaisers, Sarah Perrin, Daniel Hennes, Jeremy Shar, Cannada A. Lewis, Anian Ruoss, Tom Zahavy, Petar Veličković, Laurel Prince, Satinder Singh, Eric Malmi, and Nenad Tomasev. Mastering board games by external and internal planning with language models. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 53581–53644, 2025. URL https://proceedings.mlr.press/v267/schultz25a.html.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. In *Advances in Neural Information Processing Systems*, 2023.

Yoav Shoham and Kevin Leyton-Brown. *Multiagent Systems: Algorithmic, Game-Theoretic, and Logical Foundations*. Cambridge University Press, 2008. doi: 10.1017/CBO9780511811654. URL https://www.cambridge.org/core/books/multiagent-systems/B11B69E0CB9032D6EC0A254F59922360.

Jan Sobotka, Mustafa O. Karabag, and Ufuk Topcu. Why do LLMs struggle in strategic play? broken links between observations, beliefs, and actions. *arXiv preprint arXiv:2605.00226*, 2026.

Hao Tang, Darren Key, and Kevin Ellis. WorldCoder, a model-based LLM agent: Building world models by writing code and interacting with the environment. In *Advances in Neural Information Processing Systems*, volume 37, pp. 70148–70212, 2024. doi: 10.52202/079017-2243. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/820c61a0cd419163ccbd2c33b268816e-Paper-Conference.pdf.

Jos W. H. M. Uiterwijk. Solving strong and weak 4-in-a-row. In *IEEE Conference on Games*, 2019. URL https://ieee-cog.org/2019/papers/paper_115.pdf.

Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An open-ended embodied agent with large language models. *Transactions on Machine Learning Research*, 2024.

Wei Wang, Fuqing Bie, Junzhe Chen, Dan Zhang, Shiyu Huang, Evgeny Kharlamov, and Jie Tang. Can large language models master complex card games? In *Advances in Neural Information Processing Systems*, volume 38, pp. 24786–24821, 2025. doi: 10.52202/085713-0833. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/2341b14df6b4f684a30eb4e99807bea6-Paper-Conference.pdf.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. ReAct: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations*, 2023.

Huining Yuan, Zelai Xu, Zheyue Tan, Xiangmin Yi, Mo Guang, Kaiwen Long, Haojia Hui, Boxun Li, Xinlei Chen, Bo Zhao, Xiao-Ping Zhang, Chao Yu, and Yu Wang. MARSHAL: Incentivizing multi-agent reasoning via self-play with strategic LLMs. In *International Conference on Learning Representations*, 2026. URL https://arxiv.org/abs/2510.15414.

Jenny Zhang, Bingchen Zhao, Wannan Yang, Jakob Foerster, Jeff Clune, Minqi Jiang, Sam Devlin, and Tatiana Shavrina. Hyperagents. *arXiv preprint arXiv:2603.19461*, 2026a. URL https://arxiv.org/abs/2603.19461.

Xuan Zhang, Wenxuan Zhang, See-Kiong Ng, and Yang Deng. Self-evolving world models for LLM agent planning. *arXiv preprint arXiv:2606.30639*, 2026b. URL https://arxiv.org/abs/2606.30639. Accepted to Findings of EMNLP 2026.

APPENDIX

The appendices provide proofs and theoretical extensions, experimental diagnostics, mechanism examples, and implementation details. The guide below gives the starting page of each section.

	Section	Page
A	Theoretical Analysis and Proofs	13
B	Stockfish Diagnostics	20
C	Chess Mechanism Examples at Depth Two	22
D	Implementation Details	24
E	Reusable Self-Evolution Skill Prompt	25
F	Independent Evolution Scores	28

A THEORETICAL ANALYSIS AND PROOFS

This appendix proves the results stated in Section 5 and develops their extensions: arbitrary finite-depth error propagation, whole-game guarantees, and probabilistic refutation coverage. It also details the distinction between an explicit evaluator and an LLM that interprets state evidence.

A.1 STATE REPRESENTATIONS FOR THE FROZEN GAME-AGENT

The analysis concerns the Game-Agent's use of state evidence under fixed model weights and a fixed reasoning procedure. We consider the finite-horizon, deterministic, perfect-information, two-player zero-sum games defined in Section 3. States include the acting player and any history needed by the rules. Utilities share the Game-Agent's scale. The executor supplies exact legal transitions; $H = (\Psi, m)$ renders their states and guides the frozen LLM's interpretation. The interface need not define a closed abstract transition system: the executor computes $T(s, a)$ before Ψ represents it. This separates transition fidelity from the Game-Agent's ability to discover branches, interpret consequences, and compare them.

Condition on one realization of the LLM's randomness. The analysis assumes nonempty examined action sets and leaf judgments specified independently of the final action; it does not assert that every experimental trace records numerical judgments. The analysis concerns completed legal decisions; terminal branches use their terminal utilities. Its residuals characterize behavior rather than assuming that a minimax instruction guarantees correct execution. When traces contain no common-scale numerical judgments, the associated errors can be identified qualitatively but not assigned numerical values from prose alone.

Value equivalence formalizes models through their downstream value updates (Grimm et al., 2020); extensive-form abstraction has established connections between abstraction errors and strategy quality (Kroer & Sandholm, 2018). Here the transition model is fixed, and the analysis concerns how a bounded LLM consumes an evolving representation. Factual completeness alone does not specify the examined subtree or the interpretation of its leaves.

A.2 FROM EXPLICIT EVALUATORS TO AN LLM EVIDENCE-TO-DECISION MAP

Let T be an exact transition executor and $\phi(s)$ a state feature map. With evaluator parameters ω, an explicit-evaluator planning architecture returns an action through the policy π_{search}:

$$v_\omega(s) = f_\omega(\phi(s)), \qquad \pi_{\text{search}}(s) = \mathcal{P}_D[T, v_\omega](s). \tag{8}$$

Here f_ω may be a hand-designed heuristic or a learned nonlinear value function; it need not be a weighted sum. Rollout-based search obtains an alternative value estimate from simulated returns. $\mathcal{P}_D$ implements search and backup with depth or budget limits; finite-budget search need not find the best move. When MAX–MIN backup is implemented exactly, it cannot select a favorable reply at a MIN node contrary to its own child scores. Sampling, pruning, horizon, and value-approximation errors can nevertheless remain. This statement concerns the implemented operator, not a claim that UCT is exhaustive minimax.

RepThink instead uses a frozen, general-purpose model as the evidence-to-decision map:

$$\pi_{\theta,H}(s) = \mathcal{C}_{\theta,D}[T, H](s), \qquad H = (\Psi, m). \tag{9}$$

The operator $\mathcal{C}$ describes the Game-Agent's tool-interaction and decision process: it requests counterfactual states, discovers branches, interprets their relations using m, and chooses an action. Its weights θ, role-switching and MAX–MIN rules, and configured depth are fixed; H evolves, including its evidence-use guidance. For analysis, we describe a decision using examined sets $C, \mathcal{R}$, leaf judgments $\widehat{U}$, selected replies, and the root action; these judgments need not be recorded as numerical outputs. Thus, branch discovery, interpretation, and comparison all depend on the representation consumed, rather than the representation feeding only a scalar evaluator. The scalar judgments used in the bounds are independently defined analytic readouts, not scores computed by the representation code or invented after observing a choice.

The same frozen LLM can relate described consequences to preferences across games without a separately programmed feature-to-score rule for each game. This is not a universality theorem or an assumption that the conversion succeeds. Nor is value assessment removed: it is performed by the LLM, where semantic misunderstanding can occur. Fixed prompts also do not execute the MAX–MIN operator on the LLM's behalf. An LLM plugged into a programmatic scalar evaluator belongs to the first architecture; the distinction is the location of decision control, not the use of neural networks.

Table 4: Two architectures for converting state features into decisions.

Dimension	Explicit-evaluator search	RepThink
Use of features	Input to a value estimator or rollout policy	Evidence for branch discovery, consequence interpretation, and comparison
Feature-to-value map	Specified heuristic or learned evaluator	Frozen LLM's contextual reasoning
Representation code	Coupled to a computational evaluator or return estimator	Exposes relations without assigning utility weights or choosing actions
Backup correctness	Exact MAX–MIN implementation follows supplied scores	Prompted MIN/MAX can disagree with the model's own scores

A.3 Four levels of correctness

Simulation correctness requires legal transitions, acting-player tracking, and terminal results to agree with the game. **Representation correctness** requires faithful relations and preservation of distinctions relevant to the decision. **Interpretation and value-judgment correctness** requires the LLM to interpret the presented relations and assess their consequences, rather than merely repeat them. Branch discovery separately determines which alternatives are examined. **Adversarial-comparison correctness** requires those interpretations to affect the decision through MIN at opponent nodes and MAX at the agent's nodes. These are distinct requirements, not independent random events. Even correct relations may be presented in a form that makes the Game-Agent's interpretation or branch discovery unreliable.

To separate information preservation from use, fix a finite collection $\mathcal{X}$ of leaf-decision contexts x with reference continuation values $U(x)$. A context includes the root perspective and relevant history. Let $Z_H(x)$ contain *all* evidence available to the Game-Agent at that judgment, including any raw board, rule text, Markdown, and prior observations. For fixed θ, D and a fixed realization of its randomness, write its judgment as $\widehat{U}(x) = f_\theta(Z_H(x))$. For each nonempty information class $\mathcal{X}_z = \{x : Z_H(x) = z\}$, define

$$u_z^- = \min_{x \in \mathcal{X}_z} U(x), \qquad u_z^+ = \max_{x \in \mathcal{X}_z} U(x), \qquad h_H(z) = \tfrac{1}{2}(u_z^- + u_z^+).$$

The reference decoder h_H is an analysis device, not an implemented oracle. Define

$$\delta_H = \tfrac{1}{2} \max_z (u_z^+ - u_z^-), \qquad \kappa_{\theta,H} = \max_x |f_\theta(Z_H(x)) - h_H(Z_H(x))|. \tag{10}$$

Proposition 5.1 in the main text gives the information/interpretation bound and the refinement claim.

Proof of Proposition 5.1. For any $h(z)$, the triangle inequality gives $u_z^+ - u_z^- \leq |u_z^+ - h(z)| + |h(z) - u_z^-|$, so at least one error is half the span. The midpoint attains this bound in each class. Adding and subtracting $h_H(Z_H(x))$ proves the Game-Agent bound. Refinement partitions each old class into subsets, whose spans cannot increase. The fixed function f_θ may respond differently to the new inputs, so this argument places no constraint on its deviation from the midpoint decoder. □

If the complete board and history are already retained, Z_H can be lossless and $\delta_H = 0$. Adding explicit relations then need not add information at all: its purpose is to improve accessibility to the bounded Game-Agent, potentially changing $\kappa_{\theta,H}$ and branch-discovery behavior. A lower information-loss bound is therefore not sufficient evidence that an evolved interface is better. Conversely, if identical evidence hides states whose correct action values are $(1, 0)$ and $(0, 1)$, no deterministic evidence-only policy can be correct in both. A randomized policy choosing the first action with probability p incurs regrets $1 - p$ and p, so worst-case expected regret is at least $1/2$. No evidence-only decision rule can reliably recover a distinction absent from all of its inputs.

There is a second failure even when $\delta_H = 0$: a Game-Agent can reverse the true order of leaves, increasing $\kappa_{\theta,H}$. There is a third even when both errors vanish: it can back up MAX at an opponent's node. The counterexample in Appendix A.4 isolates this last channel. A hand-coded evaluator can also misvalue a state, but an exactly implemented backup does not have this additional prompt-execution failure.

Simulation error is a separate condition. The main analysis fixes an exact executor. For an approximate simulator, suppose its tree can be paired with the true tree with identical legal actions, node roles, and terminal structure. Let its reference leaf values $\widetilde{U}$ satisfy $|\widetilde{U} - U| \leq \epsilon_T$. Applying the proposition relative to $\widetilde{U}$ and using the triangle inequality lets us take $\epsilon = \epsilon_T + \delta_H + \kappa_{\theta,H}$. Without such structural alignment, omitted legal moves or incorrect termination cannot in general be represented by a small additive leaf error; rule correctness must be checked separately.

The four levels in the directional regret bound. Under the preceding conditions, Theorem 5.2 yields

$$R \leq \alpha + \beta_{\hat{a}} + 2(\epsilon_T + \delta_H + \kappa_{\theta,H}) + \eta_{\hat{a}} + \eta_{\mathrm{MAX}}. \tag{11}$$

Here $\epsilon_T = 0$ for the exact executor. Omission losses α, β depend jointly on evidence presentation and Game-Agent discovery; δ_H is information loss, $\kappa_{\theta,H}$ is interpretation and continuation-value judgment error, and η measures role/comparison execution. If the reference leaves are truncated-horizon estimates rather than true continuation values, the additional $2h_D$ term in Appendix A.5 is required. This decomposition separates correctness obligations without claiming that aggregate outcomes identify their individual sizes.

A.4 Adversarial roles and contingent strategies

At depth two, a threshold guarantee requires $\exists a \in \mathcal{A} \forall b \in \mathcal{B}(a) : U(a, b) \geq \tau$. Finding one favorable reply only establishes $\exists a \in \mathcal{A} \exists b \in \mathcal{B}(a)$. At greater depths, later MAX actions can depend on the opponent's earlier choices. Thus, a principal variation is a possible trajectory, whereas a strategy specifies responses to alternative opponent actions.

Proposition A.1 (Threshold strategy structure)**.** *For a finite game tree with specified true leaf values, $V(v) \geq \tau$ if and only if there exists a supporting subtree that retains at least one child at MAX nodes, all children at MIN nodes, and only leaves of value at least τ. A refuting subtree for $V(v) < \tau$ instead retains all children at MAX nodes and at least one at MIN nodes, with all retained leaves below τ.*

Proof. The statement holds at leaves. A maximum exceeds or equals τ exactly when at least one child does; a minimum does exactly when every child does. For strict inequality in the opposite direction, the existential and universal conditions reverse. Induction on tree height proves both statements. □

This is the classical AND/OR structure underlying game-tree proofs (Allis et al., 1994). It explains the logical task of role switching; RepThink does not implement proof-number search. In a selec-

tive search, unexamined alternatives contribute to omission error rather than satisfying a universal condition by default.

Even complete, correct evidence cannot compensate for an incorrect role operator. On a $[0, 1]$ utility scale, for $0 < \zeta < 1$, let the two root actions have reply values $(1 - \zeta, 1 - \zeta)$ and $(1, 0)$. Correct MAX–MIN chooses the first. Replacing MIN by MAX chooses the second and loses $1 - \zeta$ in guaranteed value, arbitrarily close to the full unit range. All leaves are known exactly. The failure is in comparison, not state reconstruction.

A.5 Directional propagation at arbitrary finite depth

Let $V(v)$ be the minimax value of a full finite tree with specified leaf values w. At each internal node v, the Game-Agent examines nonempty $\mathcal{E}(v)$, chooses $\hat{c}_v \in \mathcal{E}(v)$, and backs up $\widehat{V}(v) = \widehat{V}(\hat{c}_v)$. Define comparison residuals

$$\eta_v = \begin{cases} \max_{c\in\mathcal{E}(v)} \widehat{V}(c) - \widehat{V}(\hat{c}_v), & v \text{ MAX}, \\ \widehat{V}(\hat{c}_v) - \min_{c\in\mathcal{E}(v)} \widehat{V}(c), & v \text{ MIN}, \end{cases}$$

and true omission losses

$$\alpha_v = V(v) - \max_{c\in\mathcal{E}(v)} V(c) \quad (v \text{ MAX}), \qquad \beta_v = \min_{c\in\mathcal{E}(v)} V(c) - V(v) \quad (v \text{ MIN}).$$

All are nonnegative. At a leaf l, assume $-\epsilon_l^- \le \widehat{V}(l) - w(l) \le \epsilon_l^+$. Define $E^-(l) = \epsilon_l^-$, $E^+(l) = \epsilon_l^+$, and recursively

	$E^-(v)$	$E^+(v)$
MAX	$\alpha_v + \eta_v + \max_{c\in\mathcal{E}(v)} E^-(c)$	$\max_{c\in\mathcal{E}(v)} E^+(c)$
MIN	$\max_{c\in\mathcal{E}(v)} E^-(c)$	$\beta_v + \eta_v + \max_{c\in\mathcal{E}(v)} E^+(c)$.

(12)

Theorem A.2 (Directional propagation and root choice)**.** *Every represented node satisfies $-E^-(v) \le \widehat{V}(v) - V(v) \le E^+(v)$. At a MAX root r, let $a_C \in \arg\max_{a\in\mathcal{E}(r)} V(a)$ and let $\hat{a}$ be the chosen action. Then*

$$V(r) - V(\hat{a}) \le \alpha_r + \eta_r + E^-(a_C) + E^+(\hat{a}) \le \alpha_r + \eta_r + \max_{a\in\mathcal{E}(r)} E^-(a) + \max_{a\in\mathcal{E}(r)} E^+(a). \quad (13)$$

Proof. The leaf bound holds by assumption. At MAX, any selected child's true value is at most $V(v)$, hence $\widehat{V}(v) \le V(v) + \max_{\mathcal{E}} E^+$. Choose $c^* \in \arg\max_{\mathcal{E}} V$. The comparison residual gives

$$\widehat{V}(v) \ge \widehat{V}(c^*) - \eta_v \ge V(v) - \alpha_v - E^-(c^*) - \eta_v.$$

At MIN, any selected child's true value is at least $V(v)$, hence $\widehat{V}(v) \ge V(v) - \max_{\mathcal{E}} E^-$. Choosing $c^* \in \arg\min_{\mathcal{E}} V$ gives

$$\widehat{V}(v) \le \widehat{V}(c^*) + \eta_v \le V(v) + \beta_v + E^+(c^*) + \eta_v.$$

These inequalities establish the recursion by induction. At the root,

$$\begin{aligned} V(a_C) - V(\hat{a}) &\le V(a_C) - \widehat{V}(a_C) + \eta_r + \widehat{V}(\hat{a}) - V(\hat{a}) \\ &\le E^-(a_C) + \eta_r + E^+(\hat{a}). \end{aligned}$$

Adding $V(r) - V(a_C) = \alpha_r$ proves the root bound. □

MAX omissions and comparison failures enter the lower-error radius; their MIN counterparts enter the upper radius. The two radii can be maximized on different paths.

Evidence aggregation error. An LLM may select a reply yet change its assessment when summarizing the evidence, for example by omitting a loss or importing compensation from another branch. This differs from selecting the wrong reply under unchanged judgments. To model this additional error, replace the exact readout above by

$$\widehat{V}(v) = \widehat{V}(\hat{c}_v) + z_v, \qquad -\xi_v^- \le z_v \le \xi_v^+, \quad \xi_v^-, \xi_v^+ \ge 0. \quad (14)$$

The comparison residual η_v still measures selection among the child judgments; z_v measures the subsequent change in the selected judgment. These are analytic quantities, not errors that can be assigned numerical values from prose alone.

Define aggregation-aware radii $\widetilde{E}^{\pm}(l) = \epsilon_l^{\pm}$ at leaves. At internal nodes, the modified recursion is

$$\widetilde{E}^{-}(v) = \max_{c \in \mathcal{E}(v)} \widetilde{E}^{-}(c) + \xi_v^{-} + \begin{cases} \alpha_v + \eta_v, & v \text{ MAX}, \\ 0, & v \text{ MIN}, \end{cases} \tag{15}$$

$$\widetilde{E}^{+}(v) = \max_{c \in \mathcal{E}(v)} \widetilde{E}^{+}(c) + \xi_v^{+} + \begin{cases} 0, & v \text{ MAX}, \\ \beta_v + \eta_v, & v \text{ MIN}. \end{cases} \tag{16}$$

The same induction gives $-\widetilde{E}^{-}(v) \le \widehat{V}(v) - V(v) \le \widetilde{E}^{+}(v)$: apply the original child-selection inequalities, then add the lower and upper bounds on z_v. The root-choice bound becomes

$$V(r) - V(\hat{a}) \le \alpha_r + \eta_r + \widetilde{E}^{-}(a_C) + \widetilde{E}^{+}(\hat{a}). \tag{17}$$

Here the final MAX residual is computed from the action judgments actually used for selection. Distorting an action's summary before that comparison contributes to its aggregation error; failing to select the best of the resulting judgments contributes to η_r. A change only to the root's reported value after the action is chosen does not affect that action and adds no separate term to its regret bound.

Thus, aggregation errors at multiple levels can accumulate along reasoning paths, even when each local MIN/MAX selection is correct relative to its inputs. More depth creates additional possible aggregation stages, but does not imply that their errors must increase; the depth ablation does not separately measure z_v or η_v.

Depth-two specialization.

Proof of Theorem 5.2. Under the exact-readout assumption of the main theorem ($z_v = 0$), a MIN child a has $E^{-}(a) = \epsilon$ and $E^{+}(a) = \epsilon + \beta_a + \eta_a$. Substituting into Eq. equation 13 gives the stated inequality, with $\alpha_r = \alpha$ and $\eta_r = \eta_{\text{MAX}}$. □

Depth and horizon error. If all branches are present, comparisons are exact, and $z_v = 0$, MAX and MIN are non-expansive in the sup norm: uniformly bounded leaf error ϵ produces value error at most ϵ at every depth. Depth alone does not multiply this error. For a truncated tree whose reference leaf values w_D differ from true game continuation values V^* by at most h_D, the same induction bounds every root action's reference-value error by h_D. Adding and subtracting the two root action values shows that Eq. equation 13, or its aggregation extension in Eq. equation 17, bounds true action regret with an additional $2h_D$. Neither h_D nor the Game-Agent's omission, comparison, or aggregation errors must decrease with D.

A.6 From local errors to a whole-game guarantee

Let V^* be the true minimax continuation value of a finite game with terminal utility in $[-1, 1]$. At a MAX action define $r(s, a) = V^*(s) - V^*(T(s, a)) \ge 0$; at a MIN action define $g(s, b) = V^*(T(s, b)) - V^*(s) \ge 0$. The latter measures the opportunity returned by an opponent who does not choose its strongest continuation.

Proposition A.3 (Security gap from local continuation losses)**.** *For a MAX policy π and any MIN policy σ, every complete trajectory $s_0, \ldots, s_n$, ending at terminal state s_n, satisfies*

$$u(s_n) = V^*(s_0) - \sum_{\text{MAX}} r(s_t, a_t) + \sum_{\text{MIN}} g(s_t, b_t).$$

Consequently,

$$V^*(s_0) - \inf_{\sigma} \mathbb{E}_{\pi,\sigma} u(s_n) \le \sup_{\sigma} \mathbb{E}_{\pi,\sigma} \sum_{\text{MAX}} r(s_t, a_t). \tag{18}$$

Proof. Each MAX transition changes V^* by $-r$ and each MIN transition by $+g$. Summing these differences telescopes to $V^*(s_n) - V^*(s_0)$; terminal $V^* = u$. Taking expectations, dropping the nonnegative opponent term, and taking the supremum over σ proves the bound. □

A bound on local regret valid at all reachable MAX states can therefore be inserted into Eq. equation 18. This is a single policy's security guarantee. Evaluation against a specified MCTS or Stockfish opponent measures realized outcomes, not the infimum over all opponents; opponent mistakes

explain why local move diagnostics complement W/D/L. Centipawn losses are diagnostic reference scores, not exact V^* differences, and are not substituted into this identity.

A.7 Refutation coverage and decision-correcting revision

Fix the same true objective Q before and after a skill revision. The revised Game-Agent uses candidates C' and examined replies $\mathcal{R}'(a)$ with $|\widehat{U}' - U| \le \epsilon$. Its selected MIN readout is $J'(a) = \widehat{U}'(a, \hat{b}_a)$, with MIN residual at most $\bar{\eta}_{\min}$ and final MAX residual at most $\bar{\eta}_{\max}$.

Theorem 5.3 in the main text states the sufficient coverage condition.

Proof of Theorem 5.3. Every reply to g has true utility at least τ, so $J'(g) \ge \tau - \epsilon$. The examined witness for any inferior a gives

$$J'(a) \le \widehat{U}'(a, b_a) + \bar{\eta}_{\min} \le U(a, b_a) + \epsilon + \bar{\eta}_{\min} < \tau - \epsilon - \bar{\eta}_{\max} \le J'(g) - \bar{\eta}_{\max}.$$

Selecting such an a violates the MAX residual bound. The regret-reduction statement follows by subtracting the two true action values. □

The result requires a sufficiently damaging reply, not necessarily the unique strongest reply. It also requires covering all inferior compared candidates: removing one error can otherwise expose another. These are sufficient conditions on true utilities, not runtime certificates. The Evaluator Agent proposes revisions and uses replay to test their behavioral effect; frozen-artifact games assess transfer to other states.

Proposition A.4 (Excluding a previous action)**.** *If $a_b, g \in C'$ and $J'(g) - J'(a_b) > \bar{\eta}_{\max}$, then a choice satisfying the MAX residual bound cannot equal a_b.*

Proof. Choosing a_b would give a MAX residual at least $J'(g) - J'(a_b) > \bar{\eta}_{\max}$. □

Exclusion does not by itself guarantee improvement: a third action can be worse. Theorem 5.3 supplies the additional value and coverage conditions. A representation revision can change discovery, interpretation, and observed comparison errors by changing the evidence consumed under the same fixed prompt. This behavioral effect does not imply that the candidate-selection or MAX–MIN instructions are revised; Python and Markdown are not assigned to disjoint error terms.

A.8 Discoverability and the probability of costly errors

At a fixed depth-two decision, condition on a nonempty candidate set C. Let $q^* = \max_{\mathcal{A}} Q$, $\alpha = q^* - \max_C Q$, and choose $x > \alpha + 2\epsilon + \bar{\eta}_{\min} + \bar{\eta}_{\max}$. Define the costly candidates and sufficient refutations as

$$\mathcal{B}_x = \{a \in C : Q(a) < q^* - x\}, \qquad F(a) = \{b \in \mathcal{B}(a) : U(a, b) < q^* - \alpha - 2\epsilon - \bar{\eta}_{\min} - \bar{\eta}_{\max}\}.$$

Every $a \in \mathcal{B}_x$ has a nonempty $F(a)$. Suppose a receives n_a reply proposals. Conditional on every prior history in which no proposal for a has hit $F(a)$, proposal j has hit probability at least $p_{a,j}$. This allows adaptive, dependent proposals; the bound is conditional, not an independence assumption. Let all examined leaf errors meet ϵ except on an event of probability at most δ_E, and the MIN/MAX residual bounds hold except on an event of probability at most δ_R.

Proposition A.5 (Costly-error probability)**.** *Under these conditions,*

$$\Pr(R > x) \le \min\left\{1,\ \delta_E + \delta_R + \sum_{a \in \mathcal{B}_x} \prod_{j=1}^{n_a} (1 - p_{a,j})\right\}.$$

If $p_{a,j} \ge p_a$, the product is at most $\exp(-n_a p_a)$.

Proof. Choose $g \in \arg\max_C Q$. On the leaf-accuracy event, $J(g) \ge q^* - \alpha - \epsilon$. If a costly candidate has an examined $b \in F(a)$, then on the comparison event $J(a) < q^* - \alpha - \epsilon - \bar{\eta}_{\max} \le J(g) - \bar{\eta}_{\max}$. Such an action cannot be selected. Thus, a costly choice requires a leaf/comparison failure or an unrefuted costly candidate. Iterating the conditional failure probabilities bounds each candidate's miss probability by the product. A union bound proves the result; $1 - p \le e^{-p}$ gives the last statement. □

For a fixed LLM and reasoning procedure, the representation can change these discovery and interpretation probabilities. Extra replies help only when they supply effective coverage. This gives repair targets without assuming that every revision improves them. The experiments measure outcomes and trace mechanisms, rather than estimating the conditional probabilities in this proposition. A lower upper bound alone does not establish a lower realized risk.

A.9 Relative distortion as a complementary view

For any fixed finite action values Q, candidates C, and independently defined scores J, set $d = J - Q$, $\alpha = \max_{\mathcal{A}} Q - \max_C Q$, and $\eta_{\mathrm{MAX}} = \max_C J - J(\hat{a})$. Then

$$R \le \alpha + \big(\max_C d - \min_C d\big) + \eta_{\mathrm{MAX}}.$$

Indeed, for $a_C \in \arg\max_C Q$, adding and subtracting the two scores yields $R = \alpha + J(a_C) - J(\hat{a}) + d(\hat{a}) - d(a_C)$; the claimed inequality follows. Common additive score shifts cancel, which is why value-relevant distinctions matter more than exact numerical agreement.

Even accurate additional evidence need not improve a selective decision. For reply values $(0.9, 0.5)$ and $(0.8, 0)$, examining the first reply of each action chooses the better first action. Adding only its second reply changes the comparison to $(0.5, 0.8)$ and chooses the worse second action. No fact became less accurate; coverage changed unevenly.

A.10 Actionable repair predictions and existing evidence

A margin criterion for interventions. In Theorem 5.3, define the refutation margin

$$M = \tau - \max_{a \in C' : Q(a) < \tau} U(a, b_a), \qquad E = 2\epsilon + \bar{\eta}_{\min} + \bar{\eta}_{\max}.$$

When inferior candidates exist, a good candidate is retained, and their witnesses are examined, $M > E$ guarantees the threshold-correct choice. For fixed objective, candidates, and witnesses, reducing interpretation error by d reduces the required margin by $2d$; reducing either comparison residual by d reduces it by d. Increasing witness severity increases M. Thus the analysis prescribes different repairs for an absent refutation, a misunderstood refutation, and an ignored refutation. These are conditional improvement guarantees, not an assertion that every code or Markdown edit realizes them.

Sufficient refutation rather than complete evaluation. The code need not assign weights to every feature or reconstruct all leaf utilities. It must expose evidence that the Game-Agent can convert into sufficiently reliable comparisons. The threshold theorem requires witnesses against inferior candidates and one retained good candidate, rather than enumeration of every reply or a unique strongest reply. This is a property of selective adversarial decision-making, also exploited by classical pruning. RepThink's architectural distinction is that a general-purpose LLM performs the relational-to-value conversion and the comparisons, introducing interpretation and role-execution failure channels. The theorem does not give an LLM a search advantage for free.

Coverage rather than raw branch count. Proposition A.5 gives the miss term $\prod_j (1 - p_{a,j})$. For a constant conditional lower bound $p_a > 0$, $n_a \ge \log(1/q)/p_a$ is sufficient for a miss bound at most $q \in (0, 1)$ via the exponential bound. Increasing reply proposals is therefore useful only with effective refutation coverage; duplicates or irrelevant mechanisms need not raise p_a. Improving how a relation is exposed can improve discovery at the same proposal count. None of these conditional probabilities is inferred numerically from aggregate wins.

The evidence tests the dependency chain rather than identifying separate causal coefficients. In MiniMax Chess, language-only minimax scores 4.0, representation-only 67.0, and their depth-two combination 97.0. This supports usable state evidence as a prerequisite for the tested Game-Agent's effective lookahead; it does not imply that every LLM requires the same interface. The separate round-1/round-10 comparison tests continued adaptation at fixed depth. Wider search and deeper traces alone do not establish either refutation coverage or correct adversarial backup.

The Chess cases drawn from real LLM gameplay and decision traces distinguish reply omission (Appendix C.1), leaf interpretation (Appendix C.2), and incorrect MIN comparison (Appendix C.3).

Table 5: Conditional theoretical predictions, empirical results, and illustrative mechanisms. The Chess examples explain failure channels at $D = 2$; they do not measure intervention effects.

Failure channel	Predicted intervention	Result or illustration
Representation use: $\kappa_{\theta,H}, \alpha, \beta$	Evolve relation extraction and usage guidance at fixed D; replay tests whether the same Game-Agent uses the evidence better	Mean round-10 scores exceed round 1 in nine settings; no new rules or model weights are needed (Table 1)
Missing refutation: β	Expose reply mechanisms for the candidate state; test discovery of a damaging witness rather than count alone	Appendix C.1: an omitted $\ldots Q \times h2+$ leaves check and queen protection unexamined
Seen but misunderstood: ϵ	Clarify ownership, state changes, and net consequences to reduce error below the refutation margin	Appendix C.2: the knight's apparent fork is absent after the examined pawn capture
Incorrect MIN/MAX: η	Check which examined reply determines each action's score, on a common player scale	Appendix C.3: using the higher of two examined leaf judgments violates MIN
Depth and severe errors	Additional depth helps only if effective coverage, horizon, and interpretation improve; large-margin refutations may exclude catastrophic actions	Nonmonotonic depth ablation; Stockfish severe-error rate falls from 1.9% to 1.2% while ordinary errors do not improve uniformly (Tables 2, 3)

They provide qualitative mechanism analyses without estimating failure frequencies or revision effects. The score and move-quality experiments measure behavior; the theoretical error radii require reference utilities and independently defined judgments, which natural-language traces do not always provide.

Why tail errors can improve more than ordinary moves. One sufficiently adverse reply can exclude an inferior action with a large refutation margin. Distinguishing near-equal actions requires smaller interpretation and comparison errors. This provides a mechanism consistent with the observed reduction in costly mistakes, not a claim that all catastrophic errors are easy to expose.

For the same move set and weights, nonnegative centipawn losses obey $\overline{L} - \overline{\min(L, 1000)} = \overline{(L - 1000)_+}$. The reported rounded means give $129.0 - 83.6 = (76.1 - 68.9) + (52.9 - 14.7) = 7.2 + 38.2$ and $136.3 - 83.6 = (73.3 - 68.9) + (63.0 - 14.7) = 4.4 + 48.3$. The excess-tail shares of the uncapped ACPL reductions are therefore approximately 84.1% and 91.7%. The capped component still contains the first 1,000 cp of each severe error; it is not an isolated average of ordinary mistakes.

B STOCKFISH DIAGNOSTICS

The following diagnostics complement game outcomes by measuring move quality and material losses. We first report the aggregate results, then define their denominators.

Table 6: Chess move diagnostics against Stockfish at Elo 1320. All conditions use 100 games; LLM conditions use the DeepSeek-V4-Flash API. Representation only ($D = 0$) uses the evolved skill without prescribed lookahead, as in Table 2. All rates are percentages.

Metric	Base ReAct	Representation only ($D = 0$)	RepThink $D = 2$	MCTS-1000	MCTS-2000
Win rate	12%	17%	42%	2.0%	15.0%
ACPL (uncapped)	129.0	136.3	83.6	901.3	586.6
ACPL (cap 1000)	76.1	73.3	68.9	169.8	127.9
Error ≥ 100 cp	21.9%	20.1%	20.8%	38.8%	31.1%
Blunder ≥ 300 cp	6.4%	6.3%	5.0%	13.5%	9.6%
Severe error ≥ 500 cp	1.9%	2.4%	1.2%	9.8%	6.6%
Gift rate	4.1%	5.5%	0.8%	4.6%	3.2%

MCTS batches. Each MCTS condition plays 100 games against the same Stockfish limit, with MCTS moving first (White), as in the LLM conditions; these games are separate from the DeepSeek protocol, and no older pilot games are pooled into these results. MCTS-1000 wins 2 games and MCTS-2000 wins 15. Exact-best agreement with the first move in the engine's principal variation is 18.7% for MCTS-1000 and 21.3% for MCTS-2000.

Increasing the search budget reduces capped ACPL from 169.8 to 127.9 and the rate of losses of at least 100 cp from 38.8% to 31.1%, and raises the win rate from 2.0% to 15.0%. MCTS therefore remains fallible at these budgets but is not uniformly unable to win. Mate scores are mapped to $\pm 10{,}000$ cp; transitions into detected forced mate strongly inflate uncapped ACPL.

The move audit uses Stockfish at depth 14. ACPL and error rates include only *moves without a pre-move mate score*: Stockfish returns no mate score for the position before the evaluated action. A pre-move mate score for either side excludes the move. This filter does not depend on the number of legal actions, and no detected mate at depth 14 does not establish that forced mate is mathematically absent. A move that first enters a detected forced-mate position after the action remains included, with mate scores mapped to $\pm 10{,}000$ cp.

For included moves, centipawn loss is $\max(0, v_{\text{best}} - v_{\text{played}})$, with both evaluations taken from the mover's perspective. Uncapped ACPL averages these losses; capped ACPL first clips each loss to 1,000 cp. Error rates at 100, 300, and 500 cp divide the corresponding error counts by the number of included moves. Exact-best agreement counts equality with the first move in the engine's principal variation and is not a measure of all near-equivalent moves.

A gift requires a move without a pre-move mate score, at least 150 cp loss, an actual capture by the opponent on the next move of a piece worth at least three material units, and a net material loss of at least two units after the evaluated player's following move. Unlike ACPL and error rates, gift rate divides these events by *all moves of the evaluated player* in the corresponding group, including those excluded by the pre-move mate-score filter. A gift need not follow a capture by the LLM and does not diagnose the agent's motivation. A non-losing-to-losing event has best evaluation at least zero and played evaluation at most -150 cp; its denominator is moves without a pre-move mate score in the corresponding group.

Additional state-conditioned diagnostics are reproduced in Table 7.

Table 7: State-conditioned diagnostics for DeepSeek-V4-Flash against Stockfish at Elo 1320. Ahead/equal/behind are based on the pre-move reference evaluation, with cutoffs at approximately ± 200 cp. ACPL is capped at 1,000 cp.

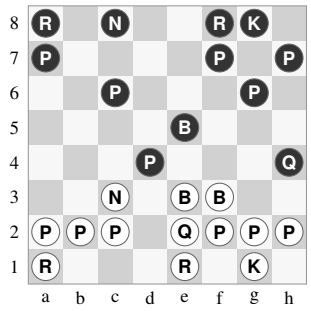
Metric	Base ReAct	Representation only ($D = 0$)	RepThink $D = 2$
Ahead: ACPL	119.3	88.4	93.2
Ahead: gift rate	7.1%	4.1%	0.5%
Ahead: non-losing to losing	6.1%	2.9%	2.7%
Equal: ACPL	67.3	64.0	57.2
Behind: ACPL	76.8	81.1	62.3

C CHESS MECHANISM EXAMPLES AT DEPTH TWO

The following examples are drawn from real LLM gameplay and decision traces and presented as qualitative mechanism analyses under the depth-two procedure in Section 4.2. White is the Game-Agent and Black the opponent. Each board shows the position before White's action a; a displayed line ends after Black's reply b. The presentations summarize the relevant decisions and state relations rather than reproduce complete traces. Alternative lines support the analysis, and mathematical judgments formalize the failure mechanisms rather than report measured model scores. These cases do not estimate failure frequencies or the effect of a skill revision.

Neither side has castling rights or an en-passant capture in these positions. White and dark circles denote the two sides; K/Q/R/B/N/P denote king, queen, rook, bishop, knight, and pawn. In the move sequences, $\times$ denotes a capture and $+$ check. Material accounting uses pawn/knight/bishop/rook/queen values of 1/3/3/5/9. The representation exposes relations at the displayed states; the LLM selects branches, interprets their consequences, and applies MAX–MIN.

C.1 MISSING AN ADVERSE REPLY



White to move, move 17. The bishop on f3 can capture the pawn on c6 and attack Black's rook on a8. But Black's queen on h4 and bishop on e5 threaten White's king.

Tempting a: $B \times c6$.
Overlooked b: $\ldots Q \times h2+$.
Defensive alternative: $g3$, attacking the queen and blocking the bishop's diagonal to h2.

Failure mechanism. The case is summarized by the LLM considering $a = B \times c6$ with $b = \ldots d \times c3$, while overlooking the checking reply below. The bishop's attack on the a8-rook can draw attention away from Black's direct king threat. A current-board mention of that threat does not establish that its consequences were examined after the candidate action. The omitted depth-two branch is

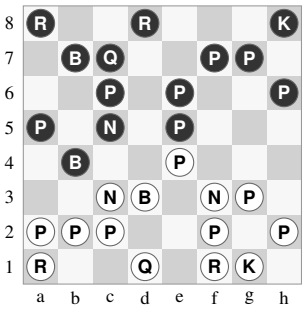
$$\underbrace{17.B \times c6}_{a} \quad \underbrace{Q \times h2+}_{b}.$$

At this leaf, White is in check and Black's bishop on e5 defends the queen on h2. These are relations in the two-ply state; no longer continuation is needed to expose them.

Evidence within the fixed horizon. For comparison, the line $17.g3\ Q \times h2+$ also ends in check, but White's pawn on g3 blocks the bishop's e5–h2 diagonal. Exposing check, protection, and blocking relations makes the two leaves distinguishable. The LLM must discover the relevant reply and use these distinctions when comparing candidates. Neither line alone certifies a forced outcome or the globally best root action.

Connection to theory. This illustrates the reply-omission channel β_a: the minimum over examined replies can exceed the minimum over all legal replies. Its magnitude requires continuation values; displaying the omitted check identifies a coverage issue without numerically estimating β_a.

C.2 Misinterpreting the State after a Reply



White to move, move 20. White's knight on c3 can jump to d5, attacking Black's queen on c7 and bishop on b4. Black's c6-pawn can capture that knight.

Tempting a: $Nd5$.
Examined b: $\ldots c \times d5$.
Leaf fact: the knight is gone, so its apparent fork no longer exists.

Failure mechanism. The move $Nd5$ appears attractive because the knight attacks Black's queen on c7 and bishop on b4. The failure is to examine the pawn's reply yet continue to credit that fork when judging the resulting leaf:

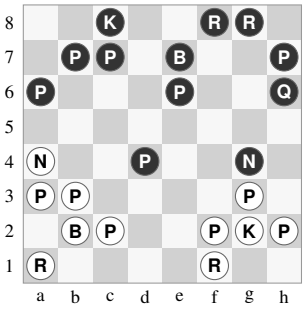
$$\underbrace{20.Nd5}_{a} \quad \underbrace{c \times d5}_{b}.$$

After these two plies, White's knight has been captured, Black's pawn occupies d5, and Black's queen and bishop remain on c7 and b4. White has lost three material units and Black has lost none along this line. The apparent fork in the intermediate a-state is absent from the (a, b) leaf.

Evidence within the fixed horizon. The representation can attach the captured-piece identity, surviving pieces, and current attacks to this leaf. Markdown can clarify that an intermediate threat must still exist after the reply to count as current evidence. Possible compensation beyond $D = 2$ remains unresolved. The three-unit material change is a state fact, not an overall utility assigned by the code.

Connection to theory. This illustrates the interpretation channel contributing to ϵ: the relevant reply is present, but the judgment uses a relation that no longer holds. Finding more replies alone does not correct that misreading. The LLM must interpret the represented leaf before comparing it with other candidates.

C.3 Using a Favorable Reply in Place of MIN



White to move, move 23. After White plays $h3$, Black can capture either f2 with the g4-knight or h3 with the h6-queen. The two replies lead to different leaf relations.

Candidate a: $h3$.
Reply b_1: $\ldots N \times f2$.
Reply b_2: $\ldots Q \times h3+$.

Two examined leaves. Both legal lines stop at depth two:

$$23.h3\ N \times f2, \qquad 23.h3\ Q \times h3+.$$

In the first leaf, the knight on f2 is attacked by White's f1-rook and defended by Black's f8-rook. In the second, the queen on h3 gives check and is attacked by White's king on g2. These attack and defense relations are visible in the leaf states without simulating an additional exchange.

Failure mechanism. To formalize the comparison failure between these replies, let the LLM's leaf judgments on White's scale satisfy $\widehat{U}(a, b_1) < \widehat{U}(a, b_2)$. Correct MIN must use the first judgment when scoring a. Using the more favorable second judgment instead creates the residual

$$\eta_a = \widehat{U}(a, b_2) - \widehat{U}(a, b_1) > 0.$$

The ordering is an illustrative assumption about the Game-Agent's judgments, not a claim that these features determine the true continuation values. Even if both judgments were accurate, substituting the higher one would still violate MIN.

Connection to theory. This isolates the comparison channel η_a from whether a reply was examined or its value was correct. Adding facts cannot ensure that the LLM uses the least favorable examined reply. Together, the three examples distinguish discovering a relevant branch, interpreting its leaf, and applying the prescribed operator, all within the same $D = 2$ interface.

D Implementation Details

This appendix separates the state evidence exposed to the LLM from the fixed opponent implementation.

Table 8: State relations exposed by the representation skill.

Game	State relations (examples)
Tic-Tac-Toe	Winning completion cells; intersecting threats; forks
Connect-Four	Playable winning cells; support dependencies; opponent threats; double threats
Generalized-TTT $5 \times 5, k = 4$	Open lines; immediate threats; multi-line forks
Hex-7	Connected components; bridges; path cost; cut locations
Chess	Material; attacks and defenses; hanging pieces; forks; king-related threats

D.1 Game interfaces and evolution boundaries

The game engine executes legal state transitions; the representation skill exposes decision-relevant relations in the resulting states. All games share `get_valid_move`, which returns the complete legal-action list, and `play_move`, which commits a selected action. As in Section 4.1, Python constructs this evidence and Markdown explains its use.

Table 9: Fixed configuration and evolving representation skill.

Fixed configuration	**Evolving skill** $H = (\Psi, m)$
Environment and agent Game rules, opponent, model weights, generic tool interfaces	**Python** Ψ State evidence and its presentation
Reasoning configuration Player objectives, role-switching and MAX–MIN rules, depth, budget	**Markdown** m Evidence meaning, applicability, and use in counterfactual comparison

D.2 MCTS opponents

Search and backed-up value. MCTS-300 and MCTS-1000 run 300 and 1,000 simulations per committed move, respectively. Tree selection uses upper confidence bounds applied to trees (UCT), with exploration coefficient 1.414; expansion samples an untried legal move uniformly. The final root action maximizes mean backed-up return, rather than visit count. Each rollout continues until termination or 500 additional plies. Returns are $+1$ for a win, -1 for a loss, and 0 for a draw or an unresolved cutoff, measured from the rollout-start player's perspective and sign-flipped between tree levels. The simple scores below guide rollout action selection; they are not substituted for terminal returns as leaf values. All opponent rules remain fixed during agent evolution.

Rollout policies. Tic-Tac-Toe, Connect-Four, and Hex sample uniformly from legal actions. Chess scores legal moves using captured material, promotion, checks, central placement, minor-piece development, post-move material balance, and a penalty for an attacked moved queen or rook with more attackers than defenders. Piece values are 100/320/330/500/900 for pawn/knight/bishop/rook/queen. Immediate checkmate receives score 100,000. With small integer score noise, the policy samples uniformly among the eight highest-scoring moves that lie within 80 points of the best score.

Generalized Tic-Tac-Toe takes an available immediate win; otherwise it scores moves using immediate-threat blocking, creation of winning continuations and forks, opponent winning continuations, central placement, and unblocked length-four line patterns. Blocking an existing immediate threat adds 500; own winning continuations add 160 each (count capped at two), with a further 260 for a fork. Opponent continuations subtract 220 each, with a further 320 penalty for two. After adding shape scores and small integer noise, the policy samples uniformly from the top eight moves within 35 points of the best. At the search root, this game additionally restricts candidates to immediate wins if present, otherwise immediate-threat blocks if present; with at most eight occupied cells, it filters moves allowing an opponent immediate win or fork whenever a surviving candidate exists. Other tree expansions use legal moves.

E Reusable Self-Evolution Skill Prompt

This portable prompt instructs an evaluator/coding agent to evolve a representation skill for a separate Game-Agent. It assumes no particular model provider, API, repository layout, or game-specific move format. The evaluator revises the skill; the Game-Agent uses the accepted skill to play. Model weights are not updated.

E.1 Experimental instantiation

Our experiments use five games per evolution round. After producing one round-1 checkpoint for each game–configuration setting, three independent continuations start from that shared checkpoint and evolve through round 10. Round 1 denotes the checkpoint after the first five evolution games, not the unevolved initialization. The round-1 checkpoint is evaluated over 100 games; each frozen round-10 artifact is evaluated separately over 100 games. Both checkpoints use $D = 2$. Evaluation games do not update the skill. The reusable instructions below take the round count, batch size, model, opponent, depth, and budget as configuration parameters.

E.2 Editable skill and fixed configuration

Act as the evaluator and coding agent. Improve one complete representation skill $H = (\Psi, m)$ through game feedback and same-position replay. Reuse the existing game executor, Game-Agent, tools, runner, and logs. Locate their actual interfaces before editing. Verify that the running Game-Agent receives the revised skill; a standalone function test does not establish deployment.

Python: construct state evidence. Map the full state to decision-relevant relations. Include the acting player, coordinate conventions, legal actions, terminal status, and history-dependent rights or resources where relevant. Apply requested hypothetical actions to a copy, validate legality, and stop at an actual terminal outcome. Keep the real state immutable and reuse the same representation for current and imagined states.

Expose concrete relations with witnesses, preconditions, and limits: for example, attacks and defenses, playable completion cells, support dependencies, connectivity, resource changes, or competing threats. Distinguish facts, conditional consequences, and heuristic features. A legal capture does not establish that the opponent will choose it, and an available recapture does not establish compensation against other replies.

Do not assign overall position values, rank actions by utility, choose or replace the agent's action, or call external search engines to construct evidence. Do not implement branch-search policies or recursive game solvers inside the representation. Execute the lines requested by the Game-Agent;

let it select branches and judge outcomes. Record rule or wiring corrections separately from representation improvements.

Markdown: explain meaning and evidence use. Python constructs state evidence; Markdown explains its meaning, applicability, and use in counterfactual comparison. Specify player perspectives, state references, conditions, and limitations, and improve how the Game-Agent uses this evidence to compare continuations. When evidence is already clear but comparison is wrong, revise usage guidance instead of adding unnecessary features.

Explain the conditions under which a consequence holds and distinguish conditional outcomes from guaranteed ones. Keep evidence consistent with the state, action sequence, and assumptions of the branch being compared. Interpret consequences with respect to the relevant participant's objective. Use compact examples without memorizing saved positions or prescribing their best actions. Remove contradictory or redundant instructions.

Fixed configuration and reasoning rule. Keep the Game-Agent model, game rules, opponent, tool access, depth, and budget fixed within old/new comparisons. The fixed reasoning prompt specifies perspective switching and MAX–MIN comparison. The evolving Markdown explains how to use the representation within that procedure; it must not change the objectives, role operators, or configured depth.

At each node, consider the acting player's objective while keeping outcome comparisons on the root player's scale. At depth two, compare $\max_{a \in C} \min_{b \in \mathcal{R}(a)} \widehat{U}(a, b)$ over the examined actions C and replies $\mathcal{R}(a)$, using the leaf judgments $\widehat{U}$ defined in Section 5. At greater depths, alternate MAX and MIN accordingly. A favorable opponent reply does not establish safety against adverse alternatives; one poor self-response does not establish that every response is poor. Selective comparisons are not exhaustive minimax certificates.

Depth counts actions from the current real root, not tool calls. Complete evaluated nonterminal lines to the configured D; actual terminal states may end earlier. Intermediate queries may expose states for choosing the next action. Do not hide deeper simulated continuations inside feature helpers or reset root depth by resetting a local board. At $D = 0$, provide current-state evidence without prescribed hypothetical lookahead. Audit tool calls and explicit continuations in available traces; do not claim to observe unrecorded internal reasoning. Record incomplete or depth-violating decisions separately.

For hidden-information games, expose only permitted observations and explicit beliefs. For stochastic games, distinguish chance from adversarial choice. General-sum or multiplayer settings require appropriate objectives and response rules established before evolution; do not silently impose two-player zero-sum MAX–MIN.

E.3 Evolution loop

Use five feedback games per batch by default, or the configured batch size. Batch size is separate from concurrency; respect repository resource limits. Follow the configured round count and budget. Maintain a resumable checkpoint with the accepted version, batch, replay results, and next action.

1. Play and preserve evidence. Freeze the accepted skill for the batch, save its hashes and configuration, and run the existing command. Preserve outcomes, move logs, model-visible tool responses, available reasoning traces, chosen actions, and reported costs. Do not edit artifacts consumed by ongoing games. Incomplete games can supply diagnostic decisions but not completed-game outcomes.

2. Select informative decisions. Review wins and losses. Locate missed opportunities, sudden deterioration, delayed consequences, successful defenses, and successful attacks. Trace losses back to an actionable error rather than repeatedly repairing terminally lost positions. Normally select three to five diagnostic positions, including a successful control when available. Positions may come from any game stage.

3. Diagnose behavior. Inspect the exact observations and tool outputs received, queried branches, recorded judgments, and final action. A passing mention does not prove a branch was checked. Do not invent reasoning absent from the trace. Distinguish:

- **State/interface error:** wrong turn, ambiguous coordinates, stale state, incorrect transition, or truncated output. Repair and verify delivery of the corrected evidence.
- **Missing alternative or refutation:** a relevant candidate or adverse reply was not considered. Expose the missing relation or explain how existing evidence reveals it, leaving branch selection to the Game-Agent.
- **Misinterpreted consequence:** a branch was examined but its losses, terminal outcome, or conditional threat was misunderstood. Clarify facts and their interpretation without prescribing an overall action value.
- **Incorrect comparison:** a cooperative opponent reply was selected, one poor self-response ruled out an entire candidate, or evidence was combined across incompatible branches. Repair ambiguous player, branch, or condition labels. If the evidence is already clear, improve Markdown guidance on how to use and compare it under the fixed MAX–MIN rule.
- **Scope/budget limitation:** decisive effects remain outside the horizon or repetitive calls exhaust the budget. Expose unresolved dependencies and reduce redundant output while retaining depth and budget.

One position may involve multiple mechanisms. For successful decisions, identify the evidence and usage that supported the action so revisions preserve those capabilities.

4. Make a targeted revision. State a testable hypothesis: "The agent missed or misused X because Y; changing the evidence or usage guidance in Z should correct this behavior." Edit Python for missing or misleading relations and Markdown for meaning, applicability, and evidence use. A revision need not change both files. Prefer small, interpretable changes to accumulating features and repeated warnings. Never encode replay identifiers or reference moves.

5. Replay the same decisions. Save each pre-decision state with all rule-relevant metadata. Test the old accepted skill and proposed skill through the actual Game-Agent interface with the same model, depth, budget, tools, and history construction. Let the agent choose its own queries and final action. Do not reveal the diagnosis, reference action, or recorded future continuation. Test targeted failures and successful controls.

Check the predicted behavioral change: did the agent identify a counteraction, interpret its consequences, keep branch evidence separate, and compare outcomes correctly? A changed action alone is insufficient; a sound alternative need not match a uniquely designated best move. Repeat ambiguous comparisons when budget permits and report all trials rather than selecting a favorable replay.

6. Accept, revise further, or revert. Provisionally accept a revision when evidence checks pass, target replays support the predicted correction, and controls show no material regression. Otherwise narrow the change or retain the previous skill. Freeze the accepted version for the next batch. New games test performance beyond replayed states and supply further feedback. One improved replay or one five-game score does not establish stable superiority. Continue for the configured rounds and budget, including rounds without an accepted revision.

E.4 Validation and reporting

Keep output compact and structurally complete. Identify the current or hypothetical state, applied line, acting player, terminal status, and relevant evidence. Retrieve detail on demand rather than forcing every game to emit an exhaustive inventory. Do not truncate structured output mid-record.

Separate two coverage records. The representation reports which relations or legal actions it enumerated and the scope of omitted detail. The harness records actually queried branches from successful tool calls and state identifiers. Neither record proves that the Game-Agent understood the evidence.

For each revision, record:

- **Configuration:** old/new artifact hashes, model, opponent, depth, tools, and budget.
- **Hypothesis and change:** target mechanism, Python changes, and Markdown changes.
- **Correctness:** agreement with the rule executor, valid state/branch references, output completeness, and depth compliance.
- **Behavior:** examined alternatives, evidence use, outcome interpretation, MIN/MAX comparison, and chosen actions in old/new replays.
- **Transfer and cost:** subsequent game counts and outcomes, controls, unresolved failures, and timing/token costs when available.

Separate replay acceptance from final evaluation. Freeze the final skill and evaluate on new games without further revision. Preserve individual run results and distinguish variation across evolved artifacts from game-level outcome uncertainty. On interruption, save the accepted version and checkpoint; do not report unfinished tests as completed results.

F Independent Evolution Scores

Scores use wins $= 1$, draws $= 0.5$, and losses $= 0$, expressed on a 0–100 scale. Each row lists three round-10 artifacts obtained by independent continuations from the same round-1 checkpoint. SD is the sample standard deviation ($n - 1$ denominator, $n = 3$); it describes between-run score variation, not a per-game standard error. Gains are differences from the shared round-1 checkpoint's reference score, not paired game-level gains; the three continuations do not measure variation in producing that initial checkpoint. Score aggregates alone do not recover W/D/L counts or per-game confidence intervals.

Game	Model	Round 1	Round-10 runs	Mean $\pm$ SD	Range
Tic-Tac-Toe	MiniMax	47.5	48, 48, 45	47.0 ± 1.7	45–48
Tic-Tac-Toe	DeepSeek	46.0	50, 49.5, 50	49.8 ± 0.3	49.5–50
Connect-Four	MiniMax	6.0	30.5, 17, 24	23.8 ± 6.8	17–30.5
Connect-Four	DeepSeek	7.0	17.5, 7, 14	12.8 ± 5.3	7–17.5
Hex-7	MiniMax	53.0	80, 76, 69	75.0 ± 5.6	69–80
Hex-7	DeepSeek	39.0	65, 75, 71	70.3 ± 5.0	65–75
Generalized-TTT	MiniMax	16.0	46, 49, 43	46.0 ± 3.0	43–49
Generalized-TTT	DeepSeek	43.0	53.5, 50.5, 46	50.0 ± 3.8	46–53.5
Chess	MiniMax	27.0	97, 85, 87	89.7 ± 6.4	85–97
Chess	DeepSeek	72.0	100, 96, 98	98.0 ± 2.0	96–100