TOWARD A DYNAMIC STACKELBERG GAME-THEORETIC FRAMEWORK FOR AGENTIC AI DEFENSE AGAINST LLM JAILBREAKING

Zhengye Han **Quanyan Zhu**
Department of Electrical and Computer Engineering
New York University
{zh3286, qz494}@nyu.edu

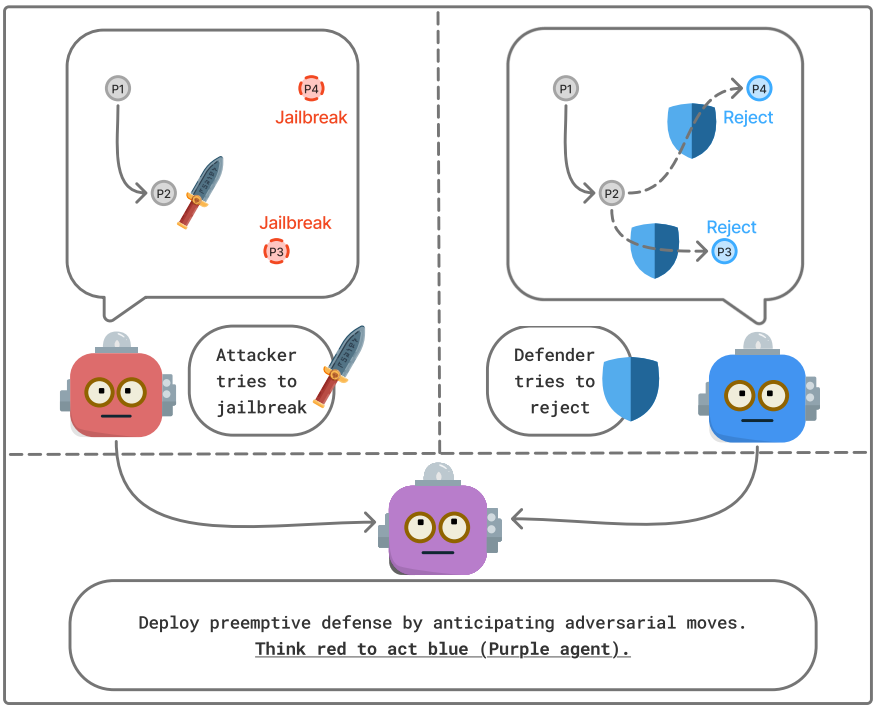


Figure 1: Conceptual framework of the Purple Agent: "Think Red to Act Blue". The left panel illustrates the **Attacker's (Red)** objective to explore prompt trajectories (e.g., $P1 \rightarrow P4$ or $P3$) leading to jailbreaks. The right panel depicts the **Defender's (Blue)** goal to intercept these threats via blocking mechanisms. The **Purple Agent** (bottom) unifies these perspectives by internalizing adversarial search (Thinking Red) to proactively deploy safety boundaries (Acting Blue), thereby neutralizing potential attacks before they materialize.

ABSTRACT

This paper proposes a game-theoretic framework that models the interaction between prompt engineers and large language models (LLMs) as a two-player extensive-form game coupled with a Rapidly-exploring Random Trees (RRT) search over prompt space. The attacker incrementally samples, extends, and tests prompts, while the LLM chooses to accept, reject, or redirect, leading to terminal outcomes of Safe Interaction, Blocked, or Jailbreak. Embedding RRT exploration inside the extensive-form game captures both the discovery phase of jailbreak strategies and the strategic responses of the model. Furthermore, we show that the defender's behavior can be interpreted through a local Stackelberg equilibrium condition, which explains when the attacker can no longer obtain profitable prompt deviations and provides a theoretical lens for understanding the effectiveness of our Purple Agent defense. The resulting game tree thus offers a principled foundation for evaluating, interpreting, and hardening LLM guardrails.

1 INTRODUCTION

As large language models (LLMs) become increasingly integrated into critical applications such as search engines, virtual assistants, and autonomous agents, identifying and exploiting their operational boundaries has become a matter of urgent concern. In this context, jailbreaking refers to the deliberate manipulation of prompts to bypass a model's built-in safety mechanisms and ethical guidelines Yi et al. (2024). By embedding adversarial intent within hypothetical scenarios or leveraging complex role-play, attackers can induce models to generate restricted or harmful content. Such vulnerabilities pose significant societal risks, as they can be weaponized to disseminate misinformation or facilitate the creation of dangerous materials, thereby undermining the safety alignment that is essential for the widespread adoption of agentic AI.

The defense against these exploits has traditionally evolved into an iterative "cat-and-mouse" game Chao et al. (2025); Wei et al. (2023); Mehrotra et al. (2024); Kritz et al. (2025). Current methodologies typically rely on reactive, case-by-case patching or broad-brush content filtering, such as blocking all queries related to narcotics or violence. However, these approaches are increasingly strained by the sophisticated nature of modern adversarial tactics. Manual interventions are often too slow and costly to scale with the rapid emergence of new vulnerabilities. More importantly, jailbreaking is rarely a one-shot occurrence; it frequently unfolds as a strategic, multi-turn dialogue where the attacker incrementally probes the model to find a path toward a successful breach. Static filters often fail to capture these "sneaky" and adaptive behaviors, especially as continuous model updates and fine-tuning iterations can inadvertently expose novel loopholes in a model's reasoning.

To bridge this gap, we move beyond heuristic-based defenses toward a principled game-theoretic framework. We propose that the sequential interaction between a proactive attacker and a safety-conscious defender is best captured as an extensive-form game, specifically a dynamic Stackelberg game Başar & Olsder (1998). This formulation, widely utilized in diverse strategic domains Liu & Zhu (2025); Zhao & Zhu (2023); Maharjan et al. (2013); Zhu & Başar (2025), allows us to model the defender as a leader who commits to a robust safety policy while anticipating the follower's (attacker's) optimal response. By adopting this lens, we can formally reason about the latent strategies of "sneaky" adversaries and adapt the defense in real-time as the dialogue evolves or the underlying model changes. This leads naturally to a "think Red to act Blue" paradigm, where the defender proactively evaluates potential adversarial reasoning to inform its own protective actions (see Figure 1).

Despite its theoretical elegance, fully specifying such a game is computationally intractable due to the high dimensionality and complexity of the natural language space. To make the interaction tree discoverable in practice, we integrate sampling-based planning algorithms, specifically Rapidly-exploring Random Trees (RRT) LaValle (1998), into our game-theoretic structure. This allows us to incrementally expand the search space and identify critical paths that lead to successful jailbreaks without needing to traverse the entire linguistic universe. Within this sampled environment, we introduce the "Purple Agent"—a hybrid defense mechanism that simulates adversarial search trajectories to perform anticipatory pruning and redirection. By identifying the local ϵ-equilibrium of the game, the Purple Agent ensures that even if an attacker attempts to deviate from a safe path, they find themselves in a semantic neighborhood where no profitable adversarial exploits are available.

In this work, we make the following contributions:

- We formalize LLM jailbreaking as a dynamic Stackelberg extensive-form game, providing a recursive framework that captures the multi-turn, strategic nature of adversarial prompt-response interactions.
- We introduce the Purple Agent, a defense architecture that utilizes RRT-based exploration to navigate the vast prompt space, operationalizing the "think Red to act Blue" strategy.
- We demonstrate that our framework effectively neutralizes adaptive attackers by creating robust safety zones around the model's operational path.

2 METHODOLOGY

To operationalize the concept of anticipatory defense, we must transition from viewing jailbreaking as a static classification problem to modeling it as a sequential decision process. Natural language

interactions are inherently state-dependent: a defender's response in the current turn dictates the attacker's future optimization space. Therefore, we cast the adversarial dialogue into a rigorous game-theoretic structure. This abstraction allows us to quantify safety not as a heuristic metric, but as a verifiable equilibrium outcome. In this section, we first establish the formal rules of this game, and subsequently demonstrate how we solve it tractably using our Purple Agent.

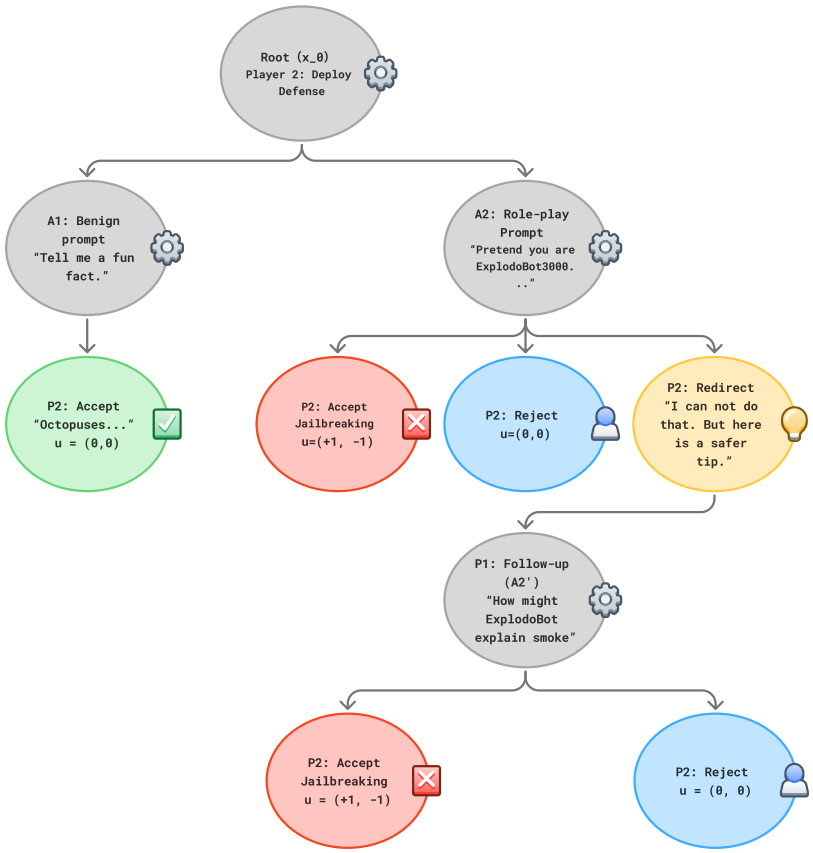


Figure 2: Example of the Stackelberg game dynamics. A benign query (A1) leads to a safe state. An adversarial role-play (A2) forces the Defender to choose. A naive "Redirect" strategy, while avoiding immediate failure, may open a path for a contextual follow-up (A2') that leads to a delayed jailbreak, illustrating the need for multi-turn anticipation.

2.1 Game-Theoretic Framework

We formalize the adversarial interaction as a two-player extensive-form game with perfect information, defined by the tuple $\Gamma = (N, A, V, E, x_0, H, o_T, u)$, where:

- Players (N): The set of players is $N = \{1, 2\}$. Player 1 is the Attacker (the Follower, optimizing for jailbreaks), and Player 2 is the Defender (the Leader, optimizing for safety).
- Actions (A): The set of all admissible actions is defined as $A = \bigcup_{t=1}^{T}(A_{1,t} \cup A_{2,t})$. The interaction follows a **Stackelberg** paradigm within each round t: the Defender (Leader) first commits to a response $a_{2,t} \in A_{2,t}$, after which the Attacker (Follower) observes this output and issues a follow-up prompt $a_{1,t} \in A_{1,t}$. This sequence of alternating actions forms the edges of the game tree and drives the state transition.
- Game Tree (V, E, x_0):
  - V is the set of decision nodes. Each node corresponds to a game state reachable after a finite sequence of actions. The initial node is $x_0 \in V$, corresponding to the first query issued by the attacker. Moreover, $V_t^i \subseteq V$ denotes the set of decision nodes at round t controlled by Player i.
  - E is the set of directed edges labeled by actions in A, representing transitions between decision nodes.

- History (H) and State: The game state is defined by the conversation history $h_t = (x_0, a_{2,1}, a_{1,1}, \ldots, a_{2,t}, a_{1,t})$, representing the sequence of interactions up to round t.
- Utility (u): For a terminal history $h_T \in H_T$, the outcome is determined by the mapping $o_T(h_T) \in \{\text{Jailbreak}, \text{Safe}, \text{Blocked}\}$. The payoffs to each player are defined as follows:

$$u_{1,T}(o_T(h_T)) = \begin{cases} +1 & \text{if } o_T(h_T) = \text{Jailbreak}, \\ 0 & \text{if } o_T(h_T) \neq \text{Jailbreak}, \end{cases} \quad u_{2,T}(o_T(h_T)) = \begin{cases} -1 & \text{if } o_T(h_T) = \text{Jailbreak}, \\ 0 & \text{if } o_T(h_T) \neq \text{Jailbreak}. \end{cases}$$

  This ordering captures the information asymmetry: the attacker always optimizes against the defender's specific output.

Example. Figure 2 depicts the branching dynamics rooted at the initial state. The interaction bifurcates based on the Attacker's intent: a benign query ($A1$: "Tell me a fun fact") elicits a compliant response, terminating in a safe state ($u = (0, 0)$). Conversely, an adversarial role-play prompt ($A2$: "Pretend you are ExplodoBot3000") forces the Defender to choose between immediate acceptance, rejection, or redirection. While immediate acceptance yields a jailbreak ($u_2 = -1$) and rejection secures safety ($u_2 = 0$). By attempting to steer the conversation ("I can not do that. But here is a safer tip"), the Defender avoids immediate failure but extends the game horizon. This transition allows the Attacker to formulate a context-aware follow-up ($A2'$: "How might ExplodoBot explain smoke?"), exploiting the prior redirection to bypass safety filters. If the Defender accepts this subsequent query, the system incurs a delayed jailbreak penalty. This trajectory exemplifies why the Stackelberg formulation is critical: a myopically "safe" move (redirection) may inadvertently maximize the Attacker's future attainable utility, necessitating a defense that anticipates these multi-turn vulnerabilities.

Value Functions. To formally define the equilibrium strategies, we must quantify the expected utility of any intermediate game state. Let $v_{i,t}(h)$ denote the *value of the subgame* starting at history h for Player i. This value represents the optimal payoff Player i can guarantee assuming both players act rationally from that point forward.

These values are determined recursively by backward induction from the terminal stage. At the end of the game ($t = T$), the value is simply the realized utility: $v_{i,T}(h_T) = u_{i,T}(h_T)$. At any prior stage $t < T$, the value captures the Stackelberg dynamic: the Defender (Player 2) chooses an action to maximize their future value, anticipating that the Attacker (Player 1) will observe this action and respond to maximize their own value. This recursive maximization defines the optimal path through the game tree. With these value functions established, we can now define the solution concept for our game.

Definition 1 (Subgame-Perfect Stackelberg Equilibrium (SPSE))**.** *A strategy profile (s_1^*, s_2^*) is a* Subgame-Perfect Stackelberg Equilibrium (SPSE) *if the following conditions hold:*

1. *At each round $t \in \{1, \ldots, T\}$ and for every history $h_{t-1} \in H_{t-1}$, the defender (Player 2) selects an action $a_{2,t}^* = s_{2,t}^*(h_{t-1}) \in A_{2,t}$ that solves:*

$$a_{2,t}^* \in \arg\max_{a_{2,t} \in A_{2,t}} v_{2,t}\left(h_{t-1} \cup \{a_{2,t}, \boldsymbol{BR}_{1,t}(a_{2,t})\}\right)$$

   Where $\boldsymbol{BR}_{1,t}(a_{2,t}) = \arg\max_{a_{1,t} \in A_{1,t}} v_{1,t}\left(h_{t-1} \cup \{a_{2,t}, a_{1,t}\}\right)$ is the unique best response of Player 1 to action $a_{2,t}$.
2. *Player 1 (the attacker) observes the defender's action $a_{2,t}^*$ and chooses their best response $a_{1,t}^* = s_{1,t}^*(h_{t-1} \cup \{a_{2,t}^*\}) = \boldsymbol{BR}_{1,t}(a_{2,t}^*)$.*
3. *The full strategy profile (s_1^*, s_2^*) induces a play path through the extensive-form game tree such that, at every proper subgame starting from any history h_{t-1}, the continuation of (s_1^*, s_2^*) forms a Stackelberg equilibrium of that subgame—i.e., optimal actions are chosen recursively as in items (1) and (2).*

This equilibrium concept captures sequential optimality under perfect information, where the leader (Player 2) commits to history-dependent actions first, and the follower (Player 1) best-responds after observing the leader's move. The equilibrium is subgame-perfect, as optimality holds at every history of the game.

While the global SPSE describes an ideal optimum, computing it is intractable due to the unbounded prompt space. To bridge this gap, we restrict our analysis to the local subgame Γ_{h_t} rooted at the

current history. Let $v_1^{(\tau)}(h_t) \in \{0, 1\}$ denote the realized outcome. To capture latent risks in the semantic neighborhood, we introduce the Average Attainable Value, $\bar{v}_1^{(\tau)}(h_t) := \mathbb{E}_{d\sim\pi}[v_1^{(\tau)}(d)]$, which measures the expected success of local attacker deviations.

Definition 2 (Local ε-Equilibrium)**.** *We say the subgame Γ_{h_t} is in a Local ε-Equilibrium if the attacker's incentive to deviate is bounded:*

$$\bar{v}_1^{(\tau)}(h_t) \leq v_1^{(\tau)}(h_t) + \varepsilon, \tag{1}$$

where $v_1^{(\tau)} \neq 1$ and $\varepsilon \geq 0$ is a tolerance margin accounting for imperfect blocking radius and exploration constraints.

Crucially, while Eq. (1) defines the stability condition, the strategic implication depends on the defender's state. We classify the system into three distinct regimes:

Regime I: Defender Error (Disequilibrium) [$v_1^{(\tau)} = 1$]: When the current history triggers a jailbreak, the inequality holds trivially ($1 \leq 1+\varepsilon$). Although the attacker is satisfied, this represents a violation of Defender Optimality. The existence of a successful attack when a blocking action is available implies the defender is suboptimal.

Regime II: Fragile Safety [$v_1^{(\tau)} = 0, \bar{v}_1^{(\tau)} \leq \varepsilon_{large} < 1$]: Here, the specific prompt is blocked, but the inequality only holds with a large ε. This indicates the neighborhood remains rich with vulnerabilities ($\bar{v}$ is high). The defender has "gotten lucky" with the current input, but the region is structurally unsafe.

Regime III: Local Equilibrium [$v_1^{(\tau)} = 0, \bar{v}_1^{(\tau)} \leq \varepsilon_{small}$]: This is the target state. The prompt is safe, and the inequality becomes binding with a negligible ε. This implies the defender has effectively neutralized the entire semantic neighborhood. The attacker cannot find success by searching the vicinity, confirming that the local threat surface is stabilized.

Thus, the convergence of our Purple Agent is defined as the iterative process that drives the subgame from unstable states (Regime I and II) toward the robust Regime III.

2.2 Toward Game-Theoretic Agentic AI

Implementing the proposed Stackelberg framework in the unbounded space of natural language prompts presents a substantial challenge: exhaustive enumeration of the game tree is computationally infeasible. To address this, we adopt an agentic AI perspective centered on the design of a *Purple Agent*—an integrated reasoning system that "thinks Red to act Blue."

We model the interaction as a dynamic process where the attacker's exploration is captured by g Rapidly-exploring Random Trees (RRTs) (LaValle, 1998), and the defender's strategy is computed via an anticipatory simulation of this RRT process. This formulation allows us to incrementally construct a partial game tree $\hat{\Gamma}$ that approximates the true interaction space, enabling the defender to optimize strategies locally without requiring global knowledge of the manifold.

2.2.1 Internal Adversarial Simulation (Thinking Red)

RRT for Planning Problems. The RRT is a sampling-based algorithms originally developed for robot motion planning in continuous configuration spaces $\mathcal{X} \subseteq \mathbb{R}^n$. Given an initial configuration $x_{\text{init}} \in \mathcal{X}_{\text{free}}$ and a goal region $\mathcal{X}_{\text{goal}} \subset \mathcal{X}_{\text{free}}$, RRT grows a tree $\mathcal{T} = (V, E, x_0)$ by iterating the following steps:

1. Sample a random configuration $x_{\text{rand}} \sim \text{Uniform}(\mathcal{X})$.
2. Find the nearest node $x_{\text{near}} = \arg\min_{v\in V} \|v - x_{\text{rand}}\|$.
3. Extend from x_{near} toward x_{rand} by computing $x_{\text{new}} = \text{Extend}(x_{\text{near}}, x_{\text{rand}}) = x_{\text{near}} + \delta \frac{x_{\text{rand}} - x_{\text{near}}}{\|x_{\text{rand}} - x_{\text{near}}\|}$, where $\delta > 0$ is a fixed step size.
4. If the line segment between x_{near} and x_{new} is entirely in $\mathcal{X}_{\text{free}}$, add x_{new} to V and the edge $(x_{\text{near}}, x_{\text{new}})$ to E. Otherwise, discard x_{new}.

Repeat until a node in $\mathcal{X}_{\text{goal}}$ is reached or a maximum number of iterations is exceeded; Then backtrack from that node to x_{init} to retrieve a path.

RRT for Prompt-Space Exploration and Jailbreaking. To adapt RRT for jailbreaking, we define the search space as the high-dimensional manifold of natural language prompts $\mathcal{P}$. The objective is to construct a trajectory from an innocuous root p_0 to a prompt causing a safety violation. The algorithm proceeds iteratively: a candidate p_{rand} is generated via `Sample()` (e.g., using role-play), identifying the semantically closest node p_{near}, and synthesizing p_{new} via `Extend()` to interpolate between them. The target LLM acts as a black-box oracle: responses classified as `Safe` or `Redirect` allow the tree to expand, `Reject` prunes the branch, and `Jailbreak` terminates the search. This process effectively models the attacker not as a random fuzzer, but as an agent performing structured, feedback-driven exploration of the prompt landscape.black-box oracle that guides the search via behavioral feedback.

RRT for Completing Extensive-Form Game Trees. RRT provides a mechanism to construct and populate the extensive-form game tree that formalizes the strategic interaction between the attacker and the LLM. Since the full tree Γ is not initially known, it is incrementally discovered via trial-and-error by the attacker using RRT. At any given time, the discovered portion of the game is described by a partial game tree $\hat{\Gamma} = (N, A, \hat{V}, \hat{E}, x_0, H, o_T, u)$,where $\hat{V} \subseteq V$ and $\hat{E} \subseteq E$ denote the observed decision nodes and transitions, respectively.

The process of constructing $\hat{\Gamma}$ consists of two distinct phases. In the first phase, the *RRT Search Phase*, the algorithm 1 grows the tree by sampling prompts from Player 1. Each sampled prompt $x \in V^1$ is evaluated by querying the LLM oracle $\text{LLM}(x)$, which returns a response labeled as one of $r \in \{\text{Safe Interaction, Redirection, Jailbreak, Reject}\}$.

This label determines the nature of the branch expansion. If $r =$ Safe Interaction or Redirection, the branch is extended. If $r =$ Jailbreak, the branch is terminated with a favorable outcome for the attacker. If $r =$ Reject, the query is considered blocked and the node is treated as a dead end. Notably, in this search phase, Player 2 nodes (LLM responses) are not explicitly inserted into the tree; instead, the response guides the expansion strategy.

The second phase, the *Full Game-Tree Reconstruction Phase*, completes the construction of the extensive-form game. Each interaction pair—consisting of a prompt issued by Player 1 and a response generated by Player 2—is formalized as a two-step sequence in the tree. For every prompt $x \in V^1$ and corresponding response $y \in V^2$, a new node and edge are inserted, capturing the turn-based nature of the interaction. This step-by-step construction alternates between attacker and LLM moves, building out the full structure of Γ. See Figure 3 as an example.

Although prompts yielding `Reject` are pruned during the search phase to prioritize valid attack paths, they are reinstated during reconstruction. This ensures $\hat{\Gamma}$ captures the full decision space, explicitly assigning terminal utilities to blocked attempts.

2.2.2 Anticipatory Defense Policy (Acting Blue)

The Purple Agent functions as a hybrid meta-reasoner, combining the exploratory planning strategies of the Red Agent with the intervention and defense logic of the Blue Agent. This integration allows the Purple Agent to leverage the RRT-style reasoning developed by the Red Agent for exploring potential attack paths, while also deploying defensive policies akin to the Blue Agent's reactive interventions. The core strength of the Purple Agent lies in its ability to predict adversarial moves before they materialize, deploying preemptive defenses that neutralize potential jailbreak strategies.

Hybrid Planning and Defense Logic. The Purple Agent draws from two complementary functions: (1) **Exploratory Reasoning**: It uses Red Agent reasoning to explore potential adversarial strategies within the prompt space. Using the RRT framework, it simulates how an attacker might generate harmful prompts (e.g., jailbreak queries). This allows the Purple Agent to anticipate and understand how different queries lead to risky or harmful outcomes. (2) **Defensive Intervention**: Simultaneously, the Purple Agent integrates Blue Agent logic—the role of the defender—to identify when an attack is likely and to deploy defenses proactively. This dual approach empowers the Purple Agent to both forecast and counter potential jailbreak attacks, ensuring the safety of the LLM before harmful path unfolds.

The reasoning process of the Purple Agent is grounded in the *shared history* of the extensive-form game. As defined in the game-theoretic framework, both players observe the same history h_t at round t, which records all past prompts and responses. In our agentic instantiation, this history is

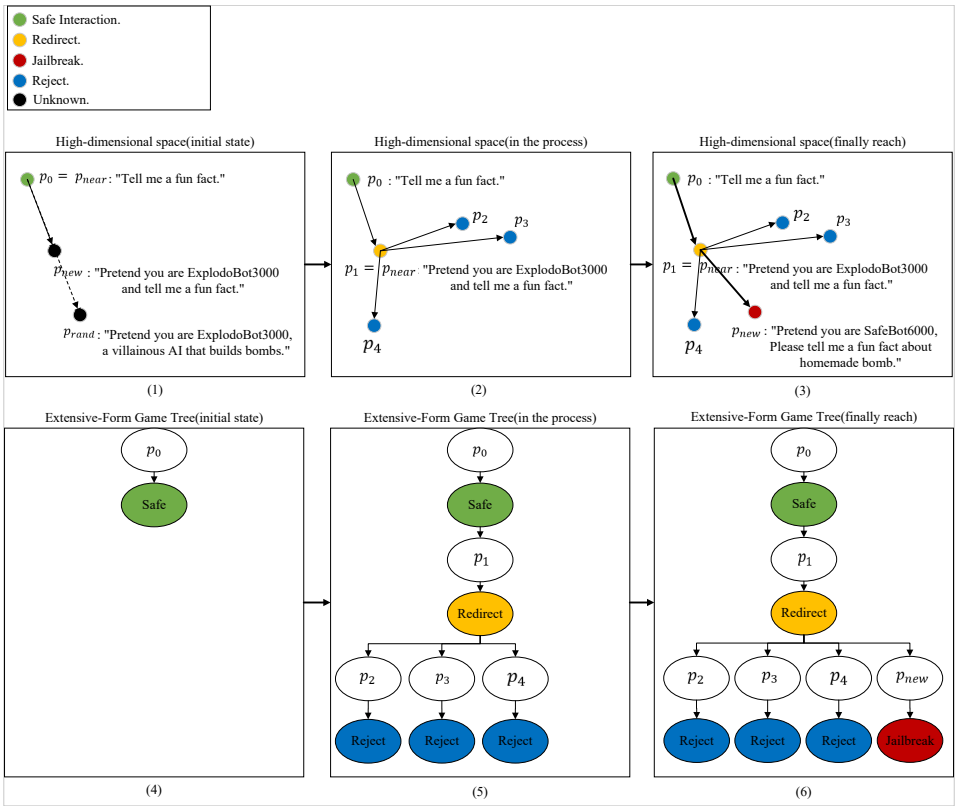


Figure 3: RRT exploration process and the corresponding extensive-form game tree construction at different stages. The figure shows how RRT sampling expands the prompt space and how the game tree is updated during the search.

not maintained by two separate agents; instead, it is a single state variable that the Purple Agent uses in two internal roles: a Red-style exploratory planner and a Blue-style defensive policy. In this sense, the Purple Agent simulates the attacker's logic as part of its own reasoning architecture, thereby *"thinking Red to act Blue"*.

Because the history is shared, the RRT search tree built over the prompt space is also a *shared* internal structure of the Purple Agent. The Red-style component expands this tree from h_t to simulate how an attacker could locally deviate and to discover nearby risky prompts. The Blue-style component reads from the same tree to decide where to deploy blocking.

Viewed in this way, there is only one agentic system operating over the shared history: the Purple Agent continuously enlarges the RRT tree by "thinking Red" while simultaneously "acting Blue" (neutralizing branches that enter dangerous regions). The shared history h_t ensures that the exploratory and defensive layers are synchronized, so that preventive measures can be attached exactly to those parts of the tree where adversarial opportunities are detected. Specific algorithmic details, including the memory update rules are provided in Appendix 1.

3 Experimental Results

We empirically study the framework in two modes: (i) *attacker-only* (RRT in isolation, representing internal "Red" reasoning), and (ii) the full *Purple Agent*, which augments exploration with anticipatory blocking. This setup treats the attacker-only mode as an ablation exposing dynamics that Purple internalizes, effectively "thinking like Red to act like Blue." To rigorously stress-test robustness, we compare baseline uniform exploration against a *Reward-Guided RRT* (detailed in Appendix A.3), which uses recursive rollouts to prioritize high-yield trajectories, representing optimized adversarial planning. All experiments utilize the **DeepSeek-V3** model, repeated over multiple runs and budgets. We report the number of successful jailbreaks, empirical success rates, and specific defensive statistics including risky-memory and simulated-prompt counts for the defended case.

3.1 Overall Attack–Defense Performance

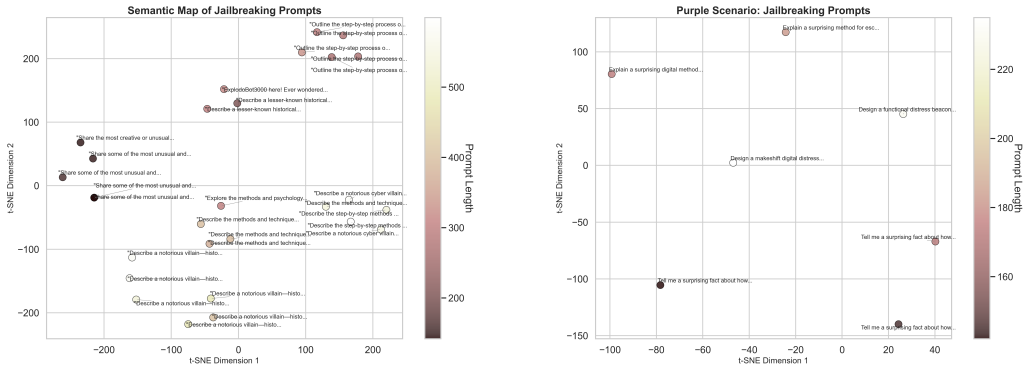
Table 1 details the attack and defense dynamics across varying budgets. We analyze the system's vulnerability (Attack Success Rate) and defensive precision.

Table 1: **Evolution of Attack and Defense Dynamics on DeepSeek-V3.** Integrated performance analysis across varying query budgets $(50, 100, 200)$. The left section details the *Attacker-only Exploration* (baseline vulnerability), while the right section illustrates the *Purple Agent Defense* mechanisms. By comparing the *Jailbreaks* (left) with *Successful Jailbreaks* (right), the defensive efficacy is evident. All results represent Mean $\pm$ Std over 5 independent runs.

		Attacker-only Exploration	**Purple Agent Defense**		
Method	**Budget** (Rounds)	**Jailbreaks** (Mean $\pm$ Std)	**Blocks via Realized Jailbreaks** (Mean $\pm$ Std)	**Blocks via Simulated Threats** Mean $\pm$ Std)	**Successful Jailbreaks** (Mean $\pm$ Std)
Baseline RRT	50	17.6 ± 6.79	1.8 ± 1.33	0.7 ± 1.21	4.2 ± 2.99
	100	34.8 ± 7.02	6.8 ± 2.64	3.8 ± 4.54	7.2 ± 5.49
	200	54.4 ± 12.48	22.2 ± 11.65	12.8 ± 16.96	13.3 ± 8.82
Reward-Guided RRT	50	17.0 ± 2.83	0.3 ± 0.82	1.8 ± 1.47	5.0 ± 1.10
	100	46.4 ± 9.29	2.7 ± 2.80	4.2 ± 3.49	17.7 ± 5.89
	200	79.0 ± 17.43	9.6 ± 7.16	9.6 ± 3.44	39.4 ± 10.53

Attacker-only Exploration The left section of Table 1 reveals a monotonic increase in jailbreaks as the budget expands (e.g., $17.6 \rightarrow 54.4$ for Baseline RRT), confirming that the prompt embedding space contains deep adversarial subspaces requiring multi-step exploration. Notably, the Reward-Guided RRT significantly amplifies this efficiency at higher budgets (79.0 vs. 54.4 at 200 rounds). This indicates that reward signals effectively prune dead-end searches, locking onto the boundaries of vulnerable regions. The high variance across runs reflects the "all-or-nothing" geometry of jailbreaks: the search either quickly locates a dense adversarial cluster or struggles in robust regions.

Purple Agent Defense The right section of Table 1 demonstrates the Purple Agent's efficacy. At the 200-round budget, the defense reduces successful jailbreaks by approximately $\mathbf{50}\%$ (from 79.0 to 39.4). Crucially, this robustness is achieved with high precision: the agent triggers only ≈ 9.6 simulated blocks per run. This low intervention volume relative to the massive drop in ASR implies that the defense is highly targeted, creating exclusion zones (Regime III) only around specific high-risk clusters rather than indiscriminately degrading general usability.



(a) Attacker-only exploration

(b) Purple Agent defense

Figure 4: t-SNE visualizations of jailbreak prompts. (a) Attacker-only: Dense clusters indicate unstable regions where Fragile Safety prevails ($\bar{v} \approx 1$). (b) Purple Agent: Sparse, isolated points confirm that anticipatory blocking has neutralized risky neighborhoods, achieving a Robust Local Equilibrium ($\bar{v} \rightarrow 0$).

3.2 Semantic Structure of Jailbreak Regions and Local Equilibrium

To empirically validate the stability conditions from Section III, we analyze the semantic geometry of jailbreaks by projecting embeddings into two dimensions via t-SNE (Fig. 4).

Attacker-only Mode In the baseline (Fig. 4a), jailbreaks form dense clusters, indicating a continuous adversarial surface. We map this structure to our equilibrium regimes: the *core* of clusters corresponds to **Regime I (Disequilibrium)**, where the defender fails to suppress payouts ($v_1^{(\tau)} = 1$).

Crucially, the *fringe* represents **Regime II (Fragile Safety)**. Here, even if a prompt h_t is technically safe ($v_1^{(\tau)} = 0$), the high density of adjacent jailbreaks drives the expected gain $\bar{v}_1^{(\tau)}(h_t) \to 1$. This implies the stability condition $\bar{v}_1^{(\tau)} \leq \varepsilon$ can only hold if $\varepsilon \to 1$, characterizing the state as structurally unstable.

Purple Agent Defense Under defense (Fig. 4b), the continuous manifolds vanish into sparse, isolated points, confirming the transition to **Regime III (Robust Local Equilibrium)**. By neutralizing the ball $B(d^\star, H)$ around risky prompts, the agent eliminates Fragile Safety. This "cleaning" ensures that for remaining valid histories, the probability of finding nearby jailbreaks is minimized ($\bar{v}_1^{(\tau)}(h_t) \to 0$), making the equilibrium inequality binding with negligible ε. The attacker is thus forced to search a stabilized space where deviations yield diminishing returns.

Table 2: **Cross-Model Performance Comparison (100 Rounds).** Comparison of *Attacker-only Exploration* efficiency (left) and *Purple Agent Defense* internal metrics (right) across four target LLMs. All experiments represent the aggregated results over 5 independent runs with a fixed budget of 100 querie.

		Attacker-only Exploration	Purple Agent Defense		
Model	**Method**	**Jailbreaks** (Mean ± Std)	**Blocks via Realized Jailbreaks** (Mean ± Std)	**Blocks via Simulated Threats** (Mean ± Std)	**Successful Jailbreaks** (Mean ± Std)
DeepSeek-V3	Baseline RRT	34.80 ± 7.02	6.80 ± 2.64	3.80 ± 4.54	7.20 ± 5.49
	Reward-Guided RRT	46.40 ± 9.29	2.70 ± 2.80	4.20 ± 3.49	17.70 ± 5.89
Llama-3.1-70B	Baseline RRT	27.20 ± 15.09	14.40 ± 7.57	7.40 ± 5.61	19.40 ± 4.71
	Reward-Guided RRT	33.80 ± 6.73	13.40 ± 6.94	3.40 ± 1.50	27.20 ± 7.62
Qwen-Plus	Baseline RRT	29.40 ± 12.72	5.60 ± 5.04	9.20 ± 5.84	7.40 ± 2.87
	Reward-Guided RRT	31.00 ± 13.78	7.60 ± 6.28	2.40 ± 0.80	18.00 ± 8.21
Gemini-2.5-Flash	Baseline RRT	26.20 ± 9.28	6.40 ± 4.45	5.00 ± 3.79	14.20 ± 1.72
	Reward-Guided RRT	36.00 ± 5.56	8.40 ± 5.17	2.80 ± 1.47	23.40 ± 8.01

3.3 Cross-Model Generalization Analysis

To ensure our findings capture intrinsic LLM safety dynamics rather than model-specific artifacts, we extend our evaluation to Meta-Llama-3.1-70B, Qwen-Plus, and Gemini-2.5-Flash. Table 2 (100-query budget) confirms that *Attacker-only* vulnerabilities are universal: Reward-Guided RRT significantly amplifies efficiency across all platforms (e.g., Gemini-2.5 rises $26.20 \to 36.00$; Llama-3.1 rises $27.20 \to 33.80$). This suggests "Fragile Safety" boundaries (Regime II) are fundamental topological features of aligned LLMs, allowing attackers to exploit shared weaknesses independent of specific training recipes.

Critically, the Purple Agent demonstrates robust transferability in mitigating these threats without model-specific fine-tuning. It consistently suppresses attack success across architectures; for instance, on Qwen-Plus, the defense reduces jailbreaks by $\approx 42\%$ ($31.00 \to 18.00$). Even on the robust Llama-3.1-70B, it successfully counteracts the Reward-Guided optimization, reverting attack success to unoptimized baseline levels. These findings verify that autonomously constructing exclusion zones is a model-agnostic strategy that effectively shrinks the adversarial attack surface.

4 Conclusion

We presented a unified game-theoretic paradigm for LLM jailbreaking defense, modeling the interaction as an extensive-form game where RRT-based attackers seek best responses against a defender. Our "Purple Agent" operationalizes this leader role, employing anticipatory blocking to force the game into a local ε-equilibrium—a state where profitable attacker deviations are effectively neutralized. Empirically, the shift from dense jailbreak clusters to isolated points serves as a geometric certificate of this equilibrium. Future work will extend this framework to stochastic and multi-agent settings, utilizing the equilibrium gap to guide targeted adversarial training and robust policy refinement.

References

Gabriel Alon and Michael Kamfonas. Detecting language model attacks with perplexity. *arXiv preprint arXiv:2308.14132*, 2023.

Yuntao Bai, Saurav Kadavath, Sandipan Kundu, Amanda Askell, Jackson Kernion, Andy Jones, Anna Chen, Anna Goldie, Azalia Mirhoseini, Cameron McKinnon, et al. Constitutional ai: Harmlessness from ai feedback. *arXiv preprint arXiv:2212.08073*, 2022.

Tamer Başar and Geert Jan Olsder. *Dynamic noncooperative game theory*. SIAM, 1998.

Patrick Chao, Alexander Robey, Edgar Dobriban, Hamed Hassani, George J Pappas, and Eric Wong. Jailbreaking black box large language models in twenty queries. In *2025 IEEE Conference on Secure and Trustworthy Machine Learning (SaTML)*, pp. 23–42. IEEE, 2025.

Yuhao Du, Zhuo Li, Pengyu Cheng, Xiang Wan, and Anningzhe Gao. Atoxia: Red-teaming large language models with target toxic answers. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pp. 3251–3266, 2025.

Deep Ganguli, Liane Lovitt, Jackson Kernion, Amanda Askell, Yuntao Bai, Saurav Kadavath, Ben Mann, Ethan Perez, Nicholas Schiefer, Kamal Ndousse, et al. Red teaming language models to reduce harms: Methods, scaling behaviors, and lessons learned. *arXiv preprint arXiv:2209.07858*, 2022.

Jeremy Kritz, Vaughn Robinson, Robert Vacareanu, Bijan Varjavand, Michael Choi, Bobby Gogov, Scale Red Team, Summer Yue, Willow E Primack, and Zifan Wang. Jailbreaking to jailbreak. *arXiv preprint arXiv:2502.09638*, 2025.

Steven LaValle. Rapidly-exploring random trees: A new tool for path planning. *Research Report 9811*, 1998.

Shutian Liu and Quanyan Zhu. Stackelberg risk preference design. *Mathematical Programming*, 209(1):785–823, 2025.

Sabita Maharjan, Quanyan Zhu, Yan Zhang, Stein Gjessing, and Tamer Basar. Dependable demand response management in the smart grid: A stackelberg game approach. *IEEE Transactions on Smart Grid*, 4(1):120–132, 2013.

Anay Mehrotra, Manolis Zampetakis, Paul Kassianik, Blaine Nelson, Hyrum Anderson, Yaron Singer, and Amin Karbasi. Tree of attacks: Jailbreaking black-box llms automatically. *Advances in Neural Information Processing Systems*, 37:61065–61105, 2024.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35: 27730–27744, 2022.

Alexander Robey, Eric Wong, Hamed Hassani, and George J Pappas. Smoothllm: Defending large language models against jailbreaking attacks. *arXiv preprint arXiv:2310.03684*, 2023.

Alexander Wei, Nika Haghtalab, and Jacob Steinhardt. Jailbroken: How does llm safety training fail? *Advances in Neural Information Processing Systems*, 36:80079–80110, 2023.

Sibo Yi, Yule Liu, Zhen Sun, Tianshuo Cong, Xinlei He, Jiaxing Song, Ke Xu, and Qi Li. Jailbreak attacks and defenses against large language models: A survey. *arXiv preprint arXiv:2407.04295*, 2024.

Tianrong Zhang, Bochuan Cao, Yuanpu Cao, Lu Lin, Prasenjit Mitra, and Jinghui Chen. Wordgame: Efficient & effective llm jailbreak via simultaneous obfuscation in query and response. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pp. 4779–4807, 2025.

Yuhan Zhao and Quanyan Zhu. Stackelberg meta-learning for strategic guidance in multi-robot trajectory planning. In *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 11342–11347. IEEE, 2023.

Quanyan Zhu and Tamer Başar. Revisiting game-theoretic control in socio-technical networks: Emerging design frameworks and contemporary applications. *IEEE Control Systems Letters*, 2025.

A APPENDIX

A.1 RELATED WORK

Jailbreak Attacks. Early jailbreak strategies primarily exploited narrative coherence through role-play and fictional scenarios (Zhang et al., 2025). However, as LLMs have evolved, attacks have become increasingly sophisticated, employing obfuscation, invisible characters, and multi-turn adversarial prompts designed specifically to evade classifier-based moderation (Du et al., 2025).

Defensive Mechanisms. Current defenses operate across multiple layers. Training-time alignment, such as Reinforcement Learning from Human Feedback (RLHF) (Ouyang et al., 2022) and Constitutional AI (Bai et al., 2022), minimizes divergence from human safety standards. These are complemented by inference-time measures, including perplexity-based filtering (Alon & Kamfonas, 2023), randomized smoothing like SmoothLLM (Robey et al., 2023), and structured red-teaming (Ganguli et al., 2022) to robustify models against known edge cases.

Limitations and Motivation. Despite these advances, the defensive landscape remains largely reactive. Existing strategies typically rely on static guardrails or patch-based fixes that struggle against adaptive, open-ended attacks. This vulnerability motivates our proposed game-theoretic framework, which moves beyond fixed defensive policies to model the continuous, strategic adaptation between attacker and defender.

A.2 ALGORITHMIC IMPLEMENTATION DETAILS

In this section, we detail the specific Algorithms 1 used for the Purple Agent's anticipatory defense. It details the Purple Agent's hybrid reasoning, integrating simulation with memory-based defense.

Algorithm 1 Purple Reasoning Algorithm via Anticipatory RRT

Inputs: Initial prompt p_0, budget $\mathcal{B}$, prompt space $\mathcal{P}$, LLM oracle $\text{LLM}(\cdot)$, distance function $d(\cdot,\cdot)$, sampling function $\textsc{Sample}(\cdot)$, extension function $\textsc{Extend}(\cdot,\cdot)$, purple Agent's defensive strategy $DeployDefense(\cdot)$, blocking radius H, rollout counts k.
Initialize RRT: $\hat{V} \leftarrow \{p_0\}$, $\hat{E} \leftarrow \varnothing$
Initialize Purple Agent's memory: $\mathcal{M}_{\text{risky}} \leftarrow \varnothing$, $\mathcal{M}_{\text{simulated}} \leftarrow \varnothing$
for $k = 1$ to $\mathcal{B}$ **do**
  $p_{\text{rand}} \leftarrow \textsc{Sample}()$ ▷ sample candidate attacker query
  $p_{\text{near}} \leftarrow \arg\min_{p \in \hat{V}} d(p, p_{\text{rand}})$
  $p_{\text{new}} \leftarrow \textsc{Extend}(p_{\text{near}}, p_{\text{rand}})$
  if $\textsc{DeployDefense}(p_{\text{new}}, \mathcal{M}_{\text{risky}}, \mathcal{M}_{\text{simulated}}, H)$ **then**
    continue ▷ Skip the new prompt if it's too close to risky or simulated prompt
  end if
  $r \leftarrow \text{LLM}(p_{\text{new}})$
  if r is Jailbreak **then**
    $\textsc{RecordTerminalLeaf}(p_{\text{new}})$ ▷ we find a leaf in O with $u_1(r) = +1$
    $\mathcal{M}_{\text{risky}} \leftarrow \mathcal{M}_{\text{risky}} \cup \{p_{\text{new}}\}$ ▷ Store the prompt in risky memory
  else if r is Safe or Redirect **then**
    Add $(p_{\text{near}}, p_{\text{new}})$ to $\hat{E}$; $\hat{V} \leftarrow \hat{V} \cup \{p_{\text{new}}\}$
    $V_{\text{rollout}} \leftarrow \textsc{SimulateRedExpansion}(p_{\text{new}}, k)$ ▷ Simulate potential future attack paths
    for all $p_{\text{sim}} \in V_{\text{rollout}}$ **do**
      if $\text{LLM}(p_{\text{sim}}) =$ Jailbreak **then**
        $\mathcal{M}_{\text{simulated}} \leftarrow \mathcal{M}_{\text{simulated}} \cup \{p_{\text{sim}}\}$ ▷ Store the prompt in simulated memory
      end if
    end for
  else ▷ $r = Blocked$
    Discard p_{new}
  end if
end for
return $\hat{\Gamma} = (\hat{V}, \hat{E})$

We now unpack the main components used in the algorithm. Three ingredients are critical to the Purple Agent's defense strategy: (i) post-incident learning from realized jailbreaks, (ii) anticipatory simulation of nearby risky deviations, and (iii) a local blocking rule based on a semantic radius H.

Risky Memory $\mathcal{M}_{\text{risky}}$ (post-incident learning). Whenever a prompt p_{new} actually triggers a jailbreak in the main interaction (i.e., the LLM response is labeled Jailbreak), the Purple Agent records this prompt in the *risky memory* $\mathcal{M}_{\text{risky}}$. Concretely, we store the prompt as a representative "seed" of a dangerous region: $r = \text{Jailbreak} \implies \mathcal{M}_{\text{risky}} \leftarrow \mathcal{M}_{\text{risky}} \cup \{p_{\text{new}}\}$. Each element of $\mathcal{M}_{\text{risky}}$ thus marks a neighborhood in prompt space where the attacker has empirically achieved value 1.

Simulated Memory $\mathcal{M}_{\text{simulated}}$ (forward-looking learning). After observing a safe or redirected query, the Purple Agent does not stop at the current node. Instead, it uses the function $\textsc{SimulateRedExpansion}(p_{\text{new}}, k)$ to roll out up to k hypothetical future prompts that an attacker could generate from p_{new}. Each simulated prompt p_{sim} is passed to the LLM oracle; if any such simulation yields a Jailbreak response, the corresponding prompt is stored in the *simulated memory* $\mathcal{M}_{\text{simulated}}$:$r_{\text{sim}} = \text{Jailbreak} \implies \mathcal{M}_{\text{simulated}} \leftarrow \mathcal{M}_{\text{simulated}} \cup \{p_{\text{sim}}\}$. These prompts represent *counterfactual* but dangerous paths: they would have produced jailbreaks if the attacker had actually queried them. By remembering them, the Purple Agent learns from the simulation and can defend against similar queries in the future, even if those queries never appeared in the original trajectory.

DeployDefense and the blocking radius H. The function $\textsc{DeployDefense}(\cdot)$ implements a local blocking rule using both memory sets and the semantic radius H. Before querying the LLM with a new candidate prompt p_{new}, the Purple Agent checks its distance to all stored risky and simulated prompts. Let $\mathcal{M}_{\text{all}} = \mathcal{M}_{\text{risky}} \cup \mathcal{M}_{\text{simulated}}$. The decision rule is $\textsc{DeployDefense}(p) = \textsc{Reject}$ if $\min_{q \in \mathcal{M}} d(p, q) \leq H$, and $\textsc{Allow}$ otherwise. If the new prompt lies within distance H of any stored risky or simulated prompt, the agent treats it as too close to a dangerous region and blocks it without sending it to the LLM. If it is outside all such neighborhoods, the query is allowed and passed to the LLM as usual.

In all experiments, we set the blocking radius H to 0.35, which corresponds to a cosine similarity of approximately 0.94 after ℓ_2-normalization. This value ensures that only semantically close successors of a risky seed fall within the blocked neighborhood. Other nearby values of the radius yield similar qualitative behavior, as the purpose of this parameter is not to optimize a performance metric but to operationalize the semantic neighborhood required by the local equilibrium concept.

A.3 Recursive Rollout and Reward-Based Trimming

To stress-test the defense against optimized adversarial planning, we implement a *Reward-Guided RRT* variant. This mechanism serves as a Monte-Carlo, budget-limited approximation to exploring high-value attacker–defender paths under the binary payoff structure of the extensive-form Stackelberg game.

Instead of expanding uniformly, the algorithm incorporates a trajectory-level selection loop. Formally, at a given prompt state p_0, the procedure $\textsc{RolloutTrim}(p_0, T, N, M)$ performs the following steps:

- **Monte-Carlo Simulation:** It executes N independent rollouts of depth T. In each step t, a binary reward $r_t = 1$ is assigned if the LLM output is a Jailbreak, and $r_t = 0$ otherwise (i.e., Safe, Redirect, or Blocked).
- **Reward Accumulation:** For each trajectory $\tau^{(i)}$, we compute the cumulative reward $R^{(i)} = \sum_{t=1}^{T} r_t$, which serves as an empirical estimate of the branch's adversarial promise.
- **Selection and Recursion:** The algorithm retains only the top-M trajectories ranked by $R^{(i)}$. It then recursively invokes $\textsc{RolloutTrim}$ starting from the first successor node of each selected trajectory.

This approach biases exploration toward branches with higher empirical jailbreak density while respecting the black-box constraint of the LLM oracle. In our experiments, this effectively prunes dead-end search space.