Divide-Fuse-Conquer: A Progressive Reinforcement Learning Strategy for Stabilizing LLMs across Diverse Games

Anonymous ACL submission

Abstract

Large language models (LLMs) are increasingly employed as language-based agents capable of reasoning, acting, and adapting through reinforcement learning (RL) with verifiable feedback. Such training often leads to sudden improvements in domains such as mathematics, coding, and vision. However, generalizing these agent behaviors across diverse interactive environments remains a major challenge. The heterogeneity of game rules, interaction modes, and environmental complexities frequently causes policies that perform well in one scenario to fail in others. Moreover, naively combining multiple environments during RL tends to introduce instability and hinder convergence. To address these issues, we propose Divide-Fuse-Conquer, a framework designed to enhance generalization in multi-task RL. The framework begins by heuristically grouping games based on characteristics such as rules and difficulty levels. Specialized models are then trained for each group to excel at games in the group, which we refer to as the divide step. This enables agents to develop expertise in specific subsets of environments. Next, we fuse model parameters across groups and continue training on multiple groups iteratively until all scenarios are conquered. Experiments on 18 TextArena games show that the Divide-Fuse-Conquer strategy substantially outperforms both rejection-sampling fine-tuning and naive multi-task RL in terms of training stability and overall performance. We hope this approach will inspire further research on developing more adaptive and general-purpose language agents through RL.

1 Introduction

In recent years, LLMs have evolved beyond static predictors into agentic systems capable of reasoning, acting, and interacting within complex environments. They demonstrate interpretable interaction patterns and growing proficiency in strategic decision-making across diverse domains, including mathematics, coding, and vision tasks (Shao et al., 2024; Xie et al., 2025; Guo et al., 2024; Huang et al., 2025). Building on these advances, games have emerged as a natural and rigorous testbed for developing and evaluating LLM-based agents. Unlike single-task settings such as math or coding, games involve diverse mechanics and shifting objectives, requiring agents to plan flexibly, adapt continuously, and generalize effectively across varied environments (Xu et al., 2021; Qi et al., 2024). Consequently, games provide a rich ground for studying the integration of RL with LLM-based agents, presenting both exciting opportunities and significant challenges (Sweetser, 2024; Singh et al., 2025; Alabi, 2025; Baek et al., 2025; Zhang et al., 2025). Reinforcement Learning with Verifiable Rewards (RLVR) (Yue et al., 2025; Guo et al., 2025; Wang et al., 2025) has recently shown notable success in improving LLM reasoning, especially in mathematics and programming tasks. However, when extending RLVR to a broader and more heterogeneous set of games, we observe severe issues such as training collapse and poor performance. These challenges arise primarily from the limited

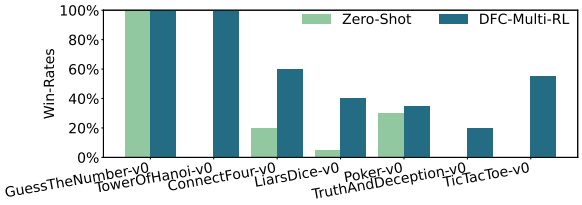

Figure 1: The win rates of Zero-Shot and DFC-Multi-RL methods. Zero-Shot refers to the base model without fine-tuning, while DFC-Multi-RL denotes the fine-tuned model using the Divide-Fuse-Conquer strategy. For single-player games such as GuessTheNumber-v0 and TowerOfHanoi-v0, the win rate represents the success of interacting with the environment. For all other two-player games, the win rate indicates the frequency with which the model outperforms Claude-3.5-Sonnet.

ability of RLVR to generalize reliably across diverse tasks and the lack of sufficient supervised guidance to stabilize learning.

To address these challenges, we propose a Divide-Fuse-Conquer (DFC-Multi-RL) strategy. In this framework, games are first heuristically grouped based on characteristics such as game rules and difficulty levels. This process is referred to as the divide step. During this stage, specialized models are trained to perform effectively within each group. Next, we initiate the fuse phase by merging model parameters from different groups into a new model, which is then further trained to conquer the combined set of games. We repeat this fuse and conquer process progressively until the model has been trained on all groups. The core of our approach lies in the conquer phase, where we focus on optimizing performance by refining the training process to effectively consolidate and expand the knowledge learned from different game groups. To achieve this, we introduce a suite of game-specific techniques aimed at enhancing the stability, efficiency, and performance of training among groups. For stability, we incorporate format reward shaping to ensure valid action for environments and apply half-negative sampling to prevent training collapse by reducing the influence of overwhelming negative gradients. To improve efficiency, we adopt mixed priority sampling, which assigns higher sampling probabilities to trajectories from games with low or intermediate win rates. This encourages learning in games where the model has not yet converged or remains uncertain. To enhance performance, we employ an ϵ-greedy disturbance (Cesa-Bianchi and Fischer, 1998) along with a randomized start to diversify the distribution of actions and initial states, encouraging broader exploration. Additionally, we introduce a step reward called a hasty-action penalty to discourage impulsive decisions and guide the model toward deeper, more strategic reasoning.

To evaluate our method in gaming scenarios, we conducted experiments across 18 games of varying types and difficulty levels within TextArena (Guertler et al., 2025), using the Qwen2.5-14B-Align model. In single-player games, DFC-Multi-RL rapidly converges to optimal performance. In two-player games against opponents such as Qwen2.5-14B-Align, Claude-3.5-Sonnet, and Llama-4-Maverick, our method consistently outperforms all baselines.

In summary, our contributions are as follows:

- We propose DFC-Multi-RL, a progressive training framework designed to enhance training stability in multi-task reinforcement learning. This framework addresses the challenge of training collapse caused by heterogeneous task distributions across games.
- We integrate a suite of techniques to improve the stability, efficiency, and performance of multi-task RL, including format reward shaping, half-negative sampling, ϵ-greedy disturbance, randomized start, mixed priority sampling, and the step reward shaping by hasty-action penalty. These techniques mitigate issues such as poor performance in games, slow convergence at the beginning, and shallow reasoning in multi-step scenarios.
- We conduct extensive experiments across 18 diverse games in TextArena and provide in-depth analysis of model behaviors during training. Our findings offer insights into LLM-based RL, guiding future efforts to scale it for more robust and generalizable diverse games.

2 Related works

Reinforcement learning for LLMs. Recent studies increasingly leverage reinforcement learning to improve LLMs' reasoning and alignment (Guo et al., 2025; Huang et al., 2025; Zhou et al., 2025). Rafailov et al. proposed Direct Preference Optimization (DPO), which bypasses reward models and sampling by analytically optimizing for human preferences. Amini et al. extended this with Offset-DPO, giving more weight to stronger preferences, improving alignment under limited data. Beyond alignment, RL has shown promise in domain-specific reasoning. For math, Shao et al. introduced DeepSeekMath and Group Relative Policy Optimization (GRPO), achieving strong MATH benchmark results. In the domain of code generation, Dutta et al. proposed a learning framework that leverages GPT-3.5 to generate preferences and trains smaller models through reinforcement learning from AI feedback, thereby improving code executability. In vision-language tasks, Zhai et al. developed Reasoning with Actions, enabling models to refine reasoning through environment interaction, achieving state-of-the-art performance. These advances highlight RL's growing role in enhancing LLMs across modalities and tasks.

Reinforcement learning in strategy-based games. The integration of reinforcement learning and large language models has seen significant ad-

Algorithm 1: Divide-Fuse-Conquer

Initialize the game set G that consists of multiple single-player and two-player games
Divide N games $[g_1, g_2, g_3, ..., g_N]$ into K groups $G = \{\mathcal{G}_1, \mathcal{G}_2, \ldots, \mathcal{G}_K\}$ according to game characteristics
Initialize the policy π
for $k = 1$ to K **do**
    Train on group $\mathcal{G}_k$ using the conquer recipe 2 with base policy π
    Obtain group-specific policy $\pi_{\mathcal{G}_k}$
end for
Set the best policy $\pi' \leftarrow \pi_{\mathcal{G}_1}$
for $k = 2$ to K **do**
    Fuse π' and $\pi_{\mathcal{G}_k}$ to obtain a fused policy: $\pi' \leftarrow \text{Fuse}(\pi', \pi_{\mathcal{G}_k})$
    Train on accumulated groups $\bigcup_{j=1}^{k} \mathcal{G}_j$ using the conquer recipe with policy π', obtaining the policy $\pi'_{\mathcal{G}_k}$
    Update best policy: $\pi' \leftarrow \pi'_{\mathcal{G}_k}$
end for
Return the best policy π'

vancements in gaming applications (Shaheen et al., 2025; Liu et al., 2021). Early work leveraged LLMs to design reward functions and game settings that support RL training. For example,Kwon et al. used LLMs to craft reward signals for Deal or No Deal, while Todd et al. applied them to generate Sokoban levels, improving agent robustness. Moving beyond auxiliary roles, LLMs now interact with RL agents in feedback loops. In Werewolf,Xu et al. had LLMs suggest strategies refined through RL trials. Likewise,Tanaka and Simo-Serra combined supervised fine-tuning and GRPO to iteratively improve agent-generated game descriptions. In multi-agent settings, where stability is difficult, Shi et al. introduced a sample-efficient, model-based method for robust learning under distributional shifts in Markov games.

3 Methods

3.1 Training Recipe

Initially, RLVR demonstrates strong performance in certain single-task settings. For instance, it rapidly masters TowerOfHanoi-v0 in TextArena, achieving perfect play after just a few updates—despite starting from a near-zero win rate. However, this success does not generalize to all tasks. Some environments remain entirely untrainable under standalone reinforcement learning, with performance stagnating at zero regardless of training duration. This reveals a key limitation—single-task training works for simple tasks but often lacks the learning signals needed for more complex ones. To address this, we extend training to multi-task settings. Encouragingly, RLVR retains its strengths on previously solvable tasks and begins to improve on those that were untrainable in isolation. For example, TowerOfHanoi-v0-medium, which showed no learning progress under single-task training, benefits immediately once the model has mastered the simpler TowerOfHanoi-v0, suggesting positive transfer between structurally related tasks. Still, this advantage diminishes when the task set becomes too diverse. As shown in Figure 2, introducing highly dissimilar games into the training mix leads to unstable learning dynamics and overall performance degradation. This indicates that both extremes—isolated single-task training and overly broad multi-task training—are suboptimal. These observations motivate a more balanced strategy that selects a focused set of related tasks to maintain training stability while supporting generalization. Therefore, we introduce the Divide-Fuse-Conquer strategy, which divides tasks and enables stable, scalable training across diverse games. Our method is detailed in Algorithm 1.

3.1.1 Divide: grouping games by scenario characteristics

The games in TextArena exhibit wide diversity in both rules and difficulties. Some games start from a fixed initial state with identical observable information and conditions for all players (e.g., ConnectFour-v0, TicTacToe-v0, TowerOfHanoi-v0), while others begin with varied configurations, such as private hands or asymmetric observations (e.g., Poker-v0, LiarsDice-v0). When starting from the same initial state, the model faces a limited observation and action space, which restricts exploration and increases the risk of getting trapped in local optima compared to using randomized starting states. For easier games, the model tends to reinforce high-reward behaviors, effectively learning within its comfort zone. Since optimizing these high-reward behaviors typically yields faster returns, training with easier games and harder games together often leads the model to prioritize the former, thereby slowing learning on the latter. To ensure a consistent learning pace within each group, we divide the games into four distinct categories based on the rules and difficulties of the game. Specifically, we first classify the games into two categories: those with random initial states and those with fixed initial states. Within each category, games are further divided based on whether the base model achieves a non-zero win rate—that is, whether it can win at all at the beginning. This

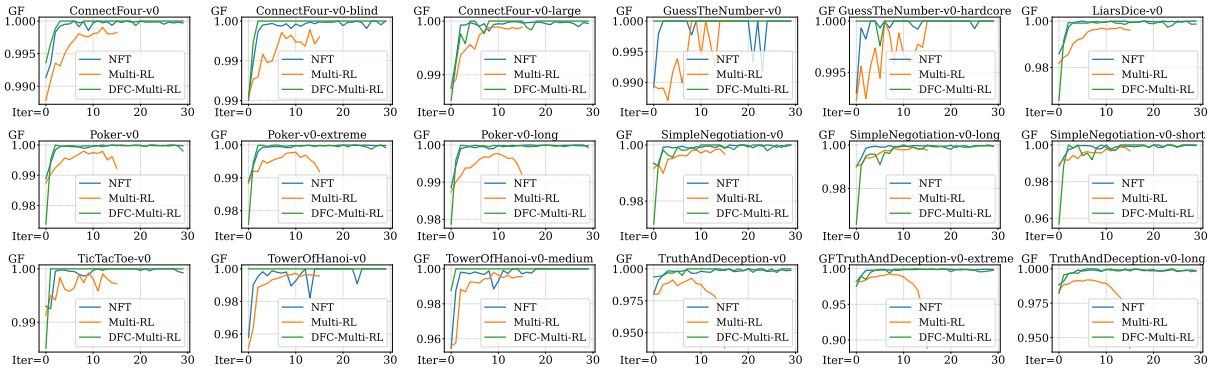

Figure 2: Comparison of the Good Format (GF) metric, which indicates how well responses follow instructions, across 18 games using three training methods: NFT, Multi-RL, and DFC-Multi-RL. Sharp drops in the metric indicate training collapse.

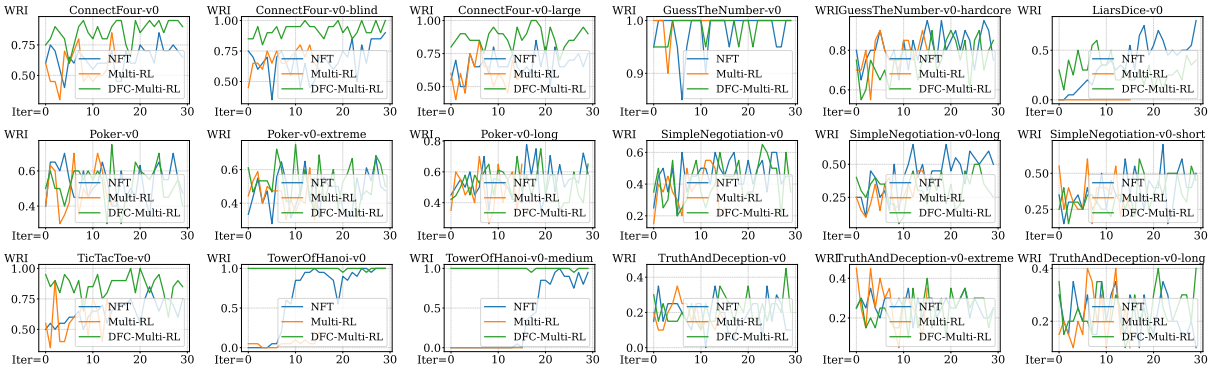

Figure 3: Win rates across 18 games using different training methods: NFT, Multi-RL, and DFC-Multi-RL.

distinction serves as a proxy for game difficulty, yielding a total of four groups.

3.1.2 Fuse: integrating specialized policies

In the multi-task setting, we assume that each group $\mathcal{G}_k$ consists of games that share similar features. For each group, we first train an individual model-a policy denoted as π_k, with parameters θ_{π_k}-which is capable of capturing the intragroup dynamics effectively. However, due to the diversity between groups, a policy trained solely within one group may not generalize well to others. To address this challenge and enable knowledge transfer across groups, we introduce a policy fusion mechanism. Specifically, we maintain a fused policy, denoted $\pi^{(k-1)}$, which integrates knowledge from the first $k-1$ groups. Its parameters $\theta^{(\pi^{(k-1)})}$ are obtained by averaging over previously learned policies. Once the individual policy π_k is obtained for group $\mathcal{G}_k$, we combine it with the previously fused policy $\pi^{(k-1)}$ via parameter averaging. This results in an updated fused policy $\pi^{(k)}$ with parameters $\theta^{(\pi^{(k)})} = \frac{1}{2}\left(\theta^{(\pi^{(k-1)})} + \theta_{\pi_k}\right)$. Here, $\pi^{(k)}$ denotes the new fused policy after incorporating the k-th individual model π_k. This fusion allows the newly acquired policy to inherit useful knowledge from previous groups, while also contributing its specialized understanding of group $\mathcal{G}_k$. The result is a gradually evolving policy that balances both stability and adaptability. By enhancing cross-group fusion, this mechanism increases the likelihood of generating high-quality trajectories across diverse groups, which facilitates faster generalization during continued training.

3.1.3 Conquer: generalizing across groups

To generalize effectively to the fused games, we perform continuous training with RL on the merged group. To enhance the stability, efficiency, and overall performance of this process, we integrate a suite of techniques with RL.

• **Reward Shaping** We design a rule-based reward system in Table 1 that assigns rewards to trajectory τ, consisting of format rewards, environment rewards, and hasty-action penalties. The format reward $R_{\text{format}}(\tau)$ is designed to encourage the model to follow instructions and produce executable actions. Responses that fail to adhere to the specified

Algorithm 2: Conquer recipe

Initialize the group of training games $\mathcal{G}_k$, the training iterations T, the number of rollout seed for each environment r
Initialize the best policy $\hat{\pi} = \pi^{(0)}$
for each t in range(1, T) **do**
  for each g_i in $\mathcal{G}_k$ **do**
    Compute the rollout seed $\mathrm{S}_{g_i}^{(t)}$ for g_i at iteration t
    for each seed s in $\mathrm{S}_{g_i}^{(t)}$ **do**
      Rollout trajectory data $\tau_{g_i}^{t}(s) \sim \pi^{(t-1)}(a|o)$ for r times with best policy $\hat{\pi}$
    end for
    Record the set of trajectories $\tau_{g_i}^{t} = \left\{\tau_{g_i}^{(t)}(s) \mid s \in \mathrm{S}_{g_i}^{(t)}\right\}$
  end for
  Record the set of trajectories $\mathcal{T}_k^{(t)} = \left\{\tau_{g_i}^{(t)} \mid g_i \in \mathcal{G}_k\right\}$
  for each τ in $\mathcal{T}_k^{(t)}$ **do**
    Compute the format reward $R_{\text{format}}(\tau)$, applying additional penalties if parsing fails
    Compute the environment reward $R_{\text{env}}(\tau)$ based on the outcome: win, loss, or draw
    Compute the total step reward $R_{\text{step}}(\tau)$ for each step
  end for
  Compute the GRPO loss $\mathcal{L}_{\text{GRPO}}(\hat{\pi})$
  Update the policy via gradient descent: $\pi^{(t)} \leftarrow \hat{\pi} - \eta \nabla_{\hat{\pi}} \mathcal{L}_{\text{GRPO}}(\hat{\pi})$
  Compute the average win rates of all games in $\mathcal{G}_k$ as $\overline{\mathrm{WR}}^{(t)} = \frac{1}{|\mathcal{G}_k|} \sum_{g_i \in \mathcal{G}_k} \mathrm{WR}_{g_i}^{(t)}$
  if $\overline{\mathrm{WR}}^{(t)} >= \overline{\mathrm{WR}}^{(t-1)}$ **then**
    Set $\hat{\pi} = \pi^{(t)}$
  end if
end for
Return the best policy $\hat{\pi}$

format are penalized with a reward of -2. The environment reward $R_{\text{env}}(\tau)$ is determined by the game outcome in TextArena, assigning -1 for a loss, 0 for a draw, and 1 for a win. The hasty-action penalty, $R_{\text{step}}(\tau)$, is designed to improve the LLM's decision-making by discouraging inefficient strategies and promoting faster victories. Specifically, we use the number of steps n_τ in trajectory τ as a penalty factor.

• **Half-negative** We mitigate training collapse by selectively filtering negative trajectories during RL. Specifically, we randomly discard half of the negative samples that may produce disproportionately large gradients. This prevents the optimization process from being dominated by negative samples and promotes more stable updates.

• **Curriculum-guided Sampling** Given that each group $\mathcal{G}_k$ includes games of different difficulty levels, uniformly sampling data during multi-task RL can introduce inefficiencies, as varying learning speeds may disrupt balanced optimization across tasks. Inspired by curriculum learning (Wang et al., 2021), we design a Mixed Prioritized Sampling (MPS) strategy that prioritizes games with the highest uncertainty and the lowest win rates for more frequent training. This approach helps regulate the learning pace across tasks of varying difficulty. Based on each game's historical win rate, MPS dynamically adjusts the number of seeds per game in each iteration t by assigning different sampling probabilities. To ensure that both low- and moderate-win-rate games in $\mathcal{G}_k$ are adequately trained, we combine the linear-capped and variance priority into a mixed sampling weight for each game g_i in $\mathcal{G}_k$ at iteration t, denoted as $\mathrm{W}_{g_i}^{(t)} = a \times \max(\epsilon_1, 1 - \mathrm{WR}_{g_i}^{(t-1)}) + b \times \mathrm{WR}_{g_i}^{(t-1)} \times (1 - \mathrm{WR}_{g_i}^{(t-1)})$, where the $\mathrm{WR}_{g_i}^{(t-1)}$ represents the win rate of game g_i at iteration $t-1$. a and b represent the weight of linear-capped priority and the variance priority, while ϵ_1 represents the capped ratio, which defines the minimum sampling proportion for each game. With the whole rollout budget S, we assign the rollout seeds of each game g_i at iteration t as $\mathrm{S}_{g_i}^{(t)} = S \times \frac{\mathrm{W}_{g_i}^{(t)}}{\sum_{j=1}^{|\mathcal{G}_k|} \mathrm{M}_{g_j}^{(t)}}$. With MPS, the number of rollout seeds $S_{g_i}^{(t)}$ for each game g_i is dynamically adjusted according to its win rate $\mathrm{WR}_{g_i}^{(t-1)}$ during the last iteration, ensuring that the model focuses more on games with lower and moderate win rates, thus improving learning efficiency.

• **ϵ-greedy Disturbance** To encourage exploration, we introduce a controlled degree of randomness into the action selection process by occasionally replacing the model's chosen action with a randomly sampled one. For games with deterministic openings and where the base model already demonstrates strong performance, we deliberately increase the value of ϵ to make the learning task more challenging. This ϵ-controlled exploration helps the model escape local optima and discover a broader range of effective strategies.

$R_{\text{format}}(\tau) = 0 \cdot 1_{\text{Valid}} + (-2) \cdot 1_{\text{Invalid}}$
$R_{\text{env}}(\tau) = 1 \cdot 1_{\text{Win}} + 0 \cdot 1_{\text{Draw}} + (-1) \cdot 1_{\text{Lose}}$
$R_{\text{step}}(\tau) = \begin{cases} R_{\text{format}}(\tau), & \text{if } R_{\text{format}}(\tau) < 0 \\ \frac{R_{\text{env}}(\tau)}{n_\tau}, & \text{if } R_{\text{env}}(\tau) > 0 \\ R_{\text{env}}(\tau), & \text{otherwise} \end{cases}$

Table 1: Definitions of the format reward $R_{\text{format}}(\tau)$, environment reward $R_{\text{env}}(\tau)$, and step penalty reward $R_{\text{step}}(\tau)$. Here, n_τ denotes the number of steps in trajectory τ.

	Games	W/D/L										
		Base	C(g1)	C(g2)	F(g1, g2)	C(g1, g2)	C(g3)	F(g1, g2, g3)	C(g1, g2, g3)	C(g4)	F(g1, g2, g3, g4)	C(g1, g2, g3, g4)
g1	Hanoi-v0	0/0/20	20/0/0	1/0/19	2/0/18	20/0/0	0/0/20	15/0/5	20/0/0	0/0/20	16/0/4	20/0/0
	Hanoi-medium	0/0/20	20/0/0	0/0/20	4/0/16	20/0/0	0/0/20	11/0/9	19/0/1	0/0/20	15/1/4	20/0/0
g2	ConnectFour-v0	9/0/11	9/3/8	15/0/5	16/0/4	16/0/4	12/1/7	12/1/7	14/1/5	11/0/9	18/1/1	20/0/0
	ConnectFour-v0-blind	11/0/9	13/0/7	15/1/4	17/0/3	18/1/1	10/0/10	19/0/1	20/0/0	9/0/11	17/1/2	19/1/0
	ConnectFour-v0-large	10/0/10	12/1/7	18/0/2	18/0/2	15/0/5	14/1/5	16/0/4	17/0/3	12/0/8	17/0/3	19/0/1
	TicTacToe-v0	8/2/10	7/5/8	14/1/5	16/1/3	20/0/0	10/2/8	13/1/6	16/0/4	8/3/9	15/1/4	17/3/0
g3	LiarsDice-v0	0/12/8	0/10/10	0/13/7	0/9/11	0/16/4	17/2/1	14/6/0	18/0/2	0/11/9	16/1/3	19/0/1
g4	GuessTheNumber-v0	19/0/1	19/0/1	20/0/0	19/0/1	19/0/1	19/0/1	20/0/0	19/1/0	20/0/0	19/0/1	20/0/0
	GuessTheNumber-v0-hardcore	10/8/2	12/7/1	15/5/0	17/3/0	16/4/0	15/4/1	13/6/1	16/4/0	20/0/0	15/5/0	18/2/0
	Poker-v0	11/1/8	12/0/8	12/0/8	6/0/14	6/3/11	13/0/7	13/2/5	12/1/7	16/1/3	10/1/9	12/0/8
	Poker-v0-long	10/0/10	7/0/12	7/1/12	9/0/11	14/0/6	11/0/9	9/1/10	8/1/10	14/2/4	10/1/9	11/0/8
	Poker-v0-extreme	10/0/9	6/0/9	10/0/8	10/0/8	11/1/6	5/2/11	11/0/5	9/0/7	9/3/8	11/1/5	12/0/6
	SimpleNegotiation-v0	9/5/6	6/8/6	3/9/8	8/5/7	6/8/6	7/3/10	7/10/3	6/9/5	16/3/1	6/13/1	8/2/10
	SimpleNegotiation-v0-short	6/8/6	4/10/6	5/6/9	4/5/11	7/8/5	4/10/6	5/12/3	8/7/5	10/4/6	6/11/3	8/7/5
	SimpleNegotiation-v0-long	7/5/8	5/7/8	4/8/8	6/8/6	8/8/4	2/11/7	2/4/14	6/6/8	8/7/5	5/6/9	6/5/9
	TruthAndDeception-v0	9/0/11	5/0/15	4/0/16	10/0/10	8/0/12	3/0/17	3/0/17	8/0/12	12/0/8	4/1/15	10/0/10
	TruthAndDeception-v0-long	9/0/11	5/0/15	5/0/15	9/0/11	7/0/13	9/0/11	6/0/14	9/0/11	9/0/11	4/0/16	7/0/13
	TruthAndDeception-v0-extreme	10/1/9	6/0/14	5/0/15	9/1/10	7/0/13	7/2/11	7/1/12	7/2/11	8/0/12	6/3/11	11/0/9
g1+g2+g3+g4		148/42/169	168/51/135	153/44/161	180/32/146	218/49/91	158/38/162	196/44/116	232/32/91	182/34/144	210/47/100	257/20/80

Table 2: As new groups are introduced, the win rates of the Fuse (F) and Conquer (C) stages vary across different groups. "W/D/L" shows the number of wins, draws, and losses out of 20 matches. Single-player games are tested on the environment, and two-player games against the Base model. Yellow highlights more wins than losses.

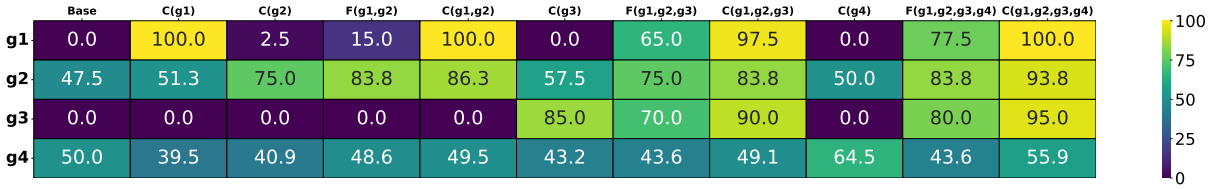

Figure 4: Average win rates during the Fuse (F) and Conquer (C) stages across different groups.

Single-Player Games	Opponent: Environment			
	Zero-Shot	RFT	Multi-RL	DFC-Multi-RL
GuessTheNumber-v0	95%	**100%**	**100%**	**100%**
GuessTheNumber-v0-hardcore	50%	85%	85%	90%
TowerOfHanoi-v0	0%	95%	10%	**100%**
TowerOfHanoi-v0-medium	0%	95%	10%	**100%**
Average	36%	94%	51%	**98%**

Table 3: Win rates of single-player games under different training methods.

• **Randomized Start** We randomize the environment during training to expose the model to a diverse range of initial states and dynamics. This helps prevent overfitting to specific trajectories and improves the model's generalization across different games. To ensure fair evaluation, we use non-overlapping start ranges for training and testing.

• **Conquer Recipe** We perform self-play rollouts to apply multi-task reinforcement learning in games. In single-player games, our model operates as Player 0, interacting directly with the environment. Conversely, in two-player games, the model alternates its role between Player 0 and Player 1 across different episodes. Each rollout plays the game to completion using the model's moves. With a rollout budget S, we collect diverse trajectories by sampling from the best historical policy. This data trains the model via the GRPO (Shao et al., 2024) algorithm. The detailed training procedure is outlined in the algorithm 2.

4 Experiments

4.1 Base Models

We use Qwen2.5-14B-Align, an in-house model derived from Qwen2.5-14B-Base through a multi-stage alignment process. First, the model undergoes Supervised Fine-Tuning (SFT) on large-scale annotated data to enhance core capabilities and language understanding. Then, Direct Preference Optimization (DPO) (Rafailov et al., 2023) is applied using human preference data to improve response quality and alignment with human expectations. This two-stage process equips Qwen2.5-14B-Align with strong general task performance, precise adherence to formatting instructions, and the ability to generate safe and reliable outputs across diverse languages and application scenarios.

4.2 Games Environments

TextArena (Guertler et al., 2025) is an open-source, text-based competition platform designed to train and evaluate the intelligent behaviors of large language models. It offers a diverse suite of both single-player and multi-player games. To ensure sufficient positive feedback for reinforcement learning, we initially evaluated all beginner-level games by running 100 trials using the base model, Qwen2.5-14B-Align. We found that in eight of these games, the model did not consistently fail. In

Two-Player Games	Opponent: Qwen2.5-14B-Align				Opponent: Llama-4-Maverick				Opponent: Claude-3.5-Sonnet			
	Zero-Shot	RFT	Multi-RL	DFC-Multi-RL	Zero-Shot	RFT	Multi-RL	DFC-Multi-RL	Zero-Shot	RFT	Multi-RL	DFC-Multi-RL
ConnectFour-v0	45%	45%	45%	**100%**	25%	35%	30%	25%	20%	45%	35%	**65%**
ConnectFour-v0-blind	55%	75%	75%	**95%**	40%	25%	35%	**75%**	0%	15%	10%	**25%**
Connect-v0-large	50%	75%	60%	**95%**	10%	5%	10%	**35%**	0%	10%	5%	**20%**
Poker-v0	55%	45%	55%	**60%**	**65%**	35%	55%	55%	30%	**35%**	25%	**35%**
Poker-v0-long	50%	50%	**65%**	60%	45%	45%	30%	**50%**	40%	**55%**	50%	45%
Poker-v0-extreme	50%	30%	35%	**55%**	20%	25%	**55%**	20%	50%	**60%**	45%	50%
LiarsDice-v0	0%	65%	0%	**95%**	0%	15%	0%	**45%**	5%	25%	10%	**40%**
SimpleNegotiation-v0	45%	40%	40%	**60%**	65%	**95%**	60%	75%	40%	**55%**	35%	45%
SimpleNegotiation-v0-short	30%	35%	25%	**40%**	25%	**80%**	55%	45%	20%	30%	**40%**	**40%**
SimpleNegotiation-v0-long	35%	**60%**	40%	40%	10%	45%	35%	**45%**	20%	20%	**30%**	25%
TruthAndDeception-v0	45%	45%	**55%**	50%	35%	45%	40%	**55%**	0%	15%	5%	**20%**
TruthAndDeception-v0-long	**45%**	25%	30%	35%	45%	**55%**	45%	45%	0%	10%	5%	**15%**
TruthAndDeception-v0-extreme	50%	35%	40%	**55%**	35%	**50%**	35%	30%	0%	0%	0%	**15%**
TicTacToe-v0	40%	65%	75%	**85%**	5%	40%	10%	**45%**	0%	45%	20%	**55%**
Average	43%	49%	46%	**66%**	30%	43%	35%	**46%**	16%	30%	23%	**35%**

Table 4: Win rates in two-player games using different training methods, evaluated against various opponents: Qwen2.5-14B-Align, Llama-4-Maverick, and Claude-3.5-Sonnet.

these cases, it occasionally wins or at least avoids consistent losses, thereby generating positive reward signals and meeting the fundamental requirement for reinforcement learning: the presence of learnable positive feedback. We then extended these games to include additional difficulty levels. For example, we expand ConnectFour-v0 into ConnectFour-v0-blind and ConnectFour-v0-large. Although these advanced versions may not independently yield positive samples, they allow us to explore the transferability of LLMs across similar games. After expansion, the evaluation suite consists of 18 environments: 4 single-player games and 14 two-player games.

4.3 Evaluation

We evaluate our framework in three settings: • **Zero-Shot**: The base model without fine-tuning. • **RFT** (Dong et al., 2023): A fine-tuning approach based on rejection sampling. • **Multi-RL** (Guo et al., 2025): A reinforcement learning method that trains the model using GRPO across diverse games. We also compare the performance of the base model under these settings with top-performing models on the TextArena leaderboard, including **LLama-4-Maverick** (Llama-4-Maverick-17B-128E-Instruct) and **Claude-3.5-Sonnet** (claude-3-5-sonnet-20241022).

4.4 Implementation Details

All experiments were carried out on 64 NVIDIA A100 GPUs (80GB each), using a global batch size of 32 and a maximum input length of 8192 tokens. Models were trained for up to $T = 100$ iterations with a learning rate of 2e-6 and a temperature of 1.0. The maximum response length was capped at 4096 tokens. For data collection, we set the average rollout budget per game to $S = 50$, with $r = 8$ trajectories per seed and a maximum of 120 steps per trajectory. During MPS, we used the parameters $a = 0.2$, $b = 0.8$, and $\epsilon_1 = 0.1$. For evaluation, 20 seeds were sampled per environment, with player roles (first and second movers) alternated. In GRPO optimization, the clip ratio ϵ was set to 0.1, and the KL penalty coefficient α was also set to 0.1. The rationale for parameter selection is provided in the Appendix.

5 Results and Discussion

Benchmark Results We evaluated the effectiveness of the Fuse and Conquer stages as new game groups were introduced, as presented in Table 2. Our results show that Fuse enabled partial transfer of capabilities from the newly added game groups, which were further enhanced during the Conquer phase. According to the average win rates within groups shown in Figure 4, the model's performance across multiple games consistently improved as additional groups were introduced. According to Figure 3, we can also observe the stability of DFC-Multi-RL's win rates throughout the training process. We also conducted comparative evaluations against models listed on the TextArena leaderboard. As shown in Tables 3 and 4, DFC-Multi-RL significantly outperformed other training methods.

Ablation Study Figure 5 highlights the importance of game-specific techniques in enhancing the stability, efficiency, and performance of multi-scenario training. For format reward shaping (FR) in Poker-v0 (top row, first two plots), the orange line (Baseline w/ FR) maintains the metric GF at 1.0 throughout training, preventing the sharp collapse observed in the baseline. For mixed priority sampling (MPS) in TowerOfHanoi-v0 and TowerOfHanoi-v0-medium (top row, third and fourth plots), the orange curves rise signif-

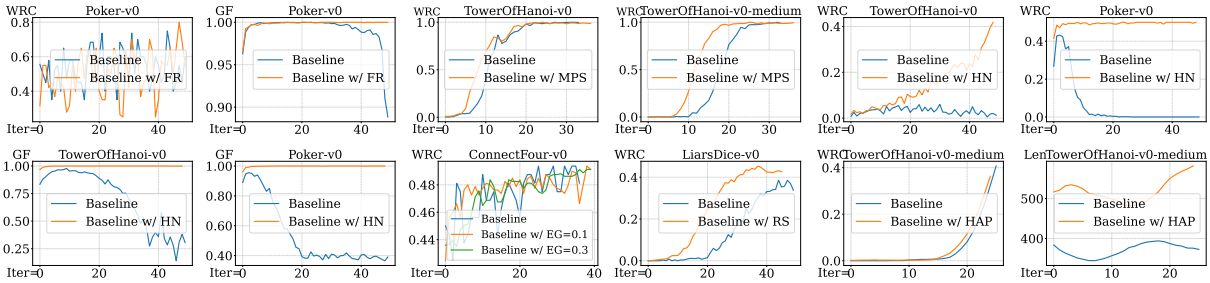

Figure 5: The ablation of techniques 'FR', 'MPS', 'HN', 'EG', 'RS', and 'HAP'. Specifically, 'FR' stands for format reward shaping, 'MPS' refers to mixed priority sampling, 'HN' denotes half-negative sampling, 'EG' represents ϵ-greedy, 'RS' represents random starts, and 'HAP' represents the hasty-action penalty. 'WRC' represents the win rate of the policy when evaluated against the currently trained opponent during training, 'Len' represents the token length of the response, and 'GF' indicates the ratio of responses with good format.

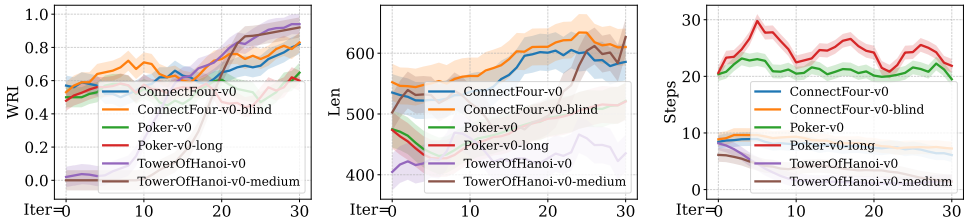

Figure 6: Variance in win rate, response length, and step count across trajectories at each iteration for different scenarios during the emergence of "Aha moments".

icantly faster than the baselines, reaching near-perfect WRC much earlier. By adjusting sampling ratios to emphasize low- and medium-win-rate scenarios, MPS broadens learning coverage, leading to faster convergence across varying difficulty levels. For half-negative sampling (HN) in Poker-v0 and TowerOfHanoi-v0 (middle-row plots), the HN approach (orange) prevents the early training collapse seen in the baseline (blue), ensuring stable and high WRC and GF. For ϵ-greedy (EG) in ConnectFour-v0 (bottom row, first plot), adding exploration noise (green/orange lines) results in a more stable and accelerated learning process. Beyond smoother convergence, we observe that with EG, the model's practical gameplay strength improved significantly, successfully defeating Claude-3.5-Sonnet, demonstrating that directed stochastic exploration is crucial for mastering the game. For randomized start (RS) in LiarsDice-v0 (bottom row, second plot), the baseline with RS (orange) achieves faster and higher WRC, validating that exposure to a broader variety of initial game states enhances robustness and strategic diversity. For hasty-action penalty (HAP) in TowerOfHanoi-v0-medium (bottom row, third and fourth plots), although the WRC improvement is modest, the baseline with HAP (orange) consistently produces longer trajectories. This suggests more deliberate thinking and cautious behavior under delayed-reward games.

Aha Moment Applying GRPO training to the 18 TextArena environments occasionally triggers "aha moments" that reflect human-like intuition. These moments reflect human-like intuition and are evident in various performance metrics—such as win rate, response length, and the number of steps per game episode (Figure 6). Compared to the base model, DFC-Multi-RL yields higher win rates, longer and more thoughtful responses, and fewer execution steps. For additional examples of "aha moments," please refer to the Appendix.

6 Conclusion

We propose Divide-Fuse-Conquer, a reinforcement learning framework designed to improve the generalization of large language models across diverse games. By heuristically grouping games, training specialized models, fusing them into a unified policy, and refining the policy for generalization, our method effectively addresses training instability in multi-task settings. Experiments on 18 TextArena games show that our approach outperforms existing multi-task baselines, highlighting the potential of structured RL to develop more adaptable and robust LLM agents.

Limitations

Despite the promising results of Divide-Fuse-Conquer framework, several limitations remain. First, the heuristic grouping of games relies on manually defined characteristics such as rules and difficulty levels, which may not fully capture the nuanced relationships among scenarios and could limit the generality of the approach. Second, while our method mitigates training collapse in multi-scenario settings, it still requires significant computational resources and careful hyperparameter tuning to achieve stable convergence. These factors highlight the need for further research to develop more automated and robust strategies for multi-scenario reinforcement learning with LLMs.

References

Moses Alabi. 2025. Reinforcement learning: Applications in gaming, robotics, and real-world decision-making.

Amin Amini, Eric Zelikman, Rafael Rafailov, and Tengyu Zhang. 2024. Offset dpo: Generalizing direct preference optimization for better alignment. *arXiv preprint arXiv:2403.09662*.

In-Chang Baek, Sung-Hyun Kim, Sam Earle, Zehua Jiang, Noh Jin-Ha, Julian Togelius, and Kyung-Joong Kim. 2025. Pcgrllm: Large language model-driven reward design for procedural content generation reinforcement learning. *arXiv preprint arXiv:2502.10906*.

Nicolo Cesa-Bianchi and Paul Fischer. 1998. Finite-time regret bounds for the multiarmed bandit problem. In *ICML*, volume 98, pages 100–108. Citeseer.

Hanze Dong, Wei Xiong, Deepanshu Goyal, Yihan Zhang, Winnie Chow, Rui Pan, Shizhe Diao, Jipeng Zhang, Kashun Shum, and Tong Zhang. 2023. Raft: Reward ranked finetuning for generative foundation model alignment. *arXiv preprint arXiv:2304.06767*.

Sujan Dutta, Sayantan Mahinder, Raviteja Anantha, and Bortik Bandyopadhyay. 2024. Applying rlaif for code generation with api-usage in lightweight llms. *arXiv preprint arXiv:2406.20060*.

Leon Guertler, Bobby Cheng, Simon Yu, Bo Liu, Leshem Choshen, and Cheston Tan. 2025. Textarena. *arXiv preprint arXiv:2504.11442*.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, and 1 others. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

Daya Guo, Qihao Zhu, Dejian Yang, Zhenda Xie, Kai Dong, Wentao Zhang, Guanting Chen, Xiao Bi, Yu Wu, YK Li, and 1 others. 2024. Deepseek-coder: When the large language model meets programming–the rise of code intelligence. *arXiv preprint arXiv:2401.14196*.

Wenxuan Huang, Bohan Jia, Zijie Zhai, Shaosheng Cao, Zheyu Ye, Fei Zhao, Yao Hu, and Shaohui Lin. 2025. Vision-r1: Incentivizing reasoning capability in multimodal large language models. *arXiv preprint arXiv:2503.06749*.

Minae Kwon, Sang Michael Xie, Kalesha Bullard, and Dorsa Sadigh. 2023. Reward design with language models. *arXiv preprint arXiv:2303.00001*.

Shanqi Liu, Junjie Cao, Yujie Wang, Wenzhou Chen, and Yong Liu. 2021. Self-play reinforcement learning with comprehensive critic in computer games. *Neurocomputing*, 449:207–213.

Siyuan Qi, Shuo Chen, Yexin Li, Xiangyu Kong, Junqi Wang, Bangcheng Yang, Pring Wong, Yifan Zhong, Xiaoyuan Zhang, Zhaowei Zhang, and 1 others. 2024. Civrealm: A learning and reasoning odyssey in civilization for decision-making agents. *arXiv preprint arXiv:2401.10568*.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Stefano Ermon, Christopher D. Manning, and Chelsea Finn. 2023. Direct preference optimization: Your language model is secretly a reward model. *arXiv preprint arXiv:2305.18290*.

Abdelrhman Shaheen, Anas Badr, Ali Abohendy, Hatem Alsaadawy, and Nadine Alsayad. 2025. Reinforcement learning in strategy-based and atari games: A review of google deepminds innovations. *arXiv preprint arXiv:2502.10303*.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Y Wu, and 1 others. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*.

Laixi Shi, Eric Mazumdar, Yuejie Chi, and Adam Wierman. 2024. Sample-efficient robust multi-agent reinforcement learning in the face of environmental uncertainty. *arXiv preprint arXiv:2404.18909*.

Joykirat Singh, Raghav Magazine, Yash Pandya, and Akshay Nambi. 2025. Agentic reasoning and tool integration for llms via reinforcement learning. *arXiv preprint arXiv:2505.01441*.

Penny Sweetser. 2024. Large language models and video games: A preliminary scoping review. In *Proceedings of the 6th ACM Conference on Conversational User Interfaces*, pages 1–8.

Tsunehiko Tanaka and Edgar Simo-Serra. 2025. Grammar and gameplay-aligned rl for game description generation with llms. *arXiv preprint arXiv:2503.15783*.

Graham Todd, Sam Earle, Muhammad Umair Nasir, Michael Cerny Green, and Julian Togelius. 2023. Level generation through large language models. In *Proceedings of the 18th International Conference on the Foundations of Digital Games*, pages 1–8.

Xin Wang, Yudong Chen, and Wenwu Zhu. 2021. A survey on curriculum learning. *IEEE transactions on pattern analysis and machine intelligence*, 44(9):4555–4576.

Yiping Wang, Qing Yang, Zhiyuan Zeng, Liliang Ren, Liyuan Liu, Baolin Peng, Hao Cheng, Xuehai He, Kuan Wang, Jianfeng Gao, and 1 others. 2025. Reinforcement learning for reasoning in large language models with one training example. *arXiv preprint arXiv:2504.20571*.

Tian Xie, Zitian Gao, Qingnan Ren, Haoming Luo, Yuqian Hong, Bryan Dai, Joey Zhou, Kai Qiu, Zhirong Wu, and Chong Luo. 2025. Logic-rl: Unleashing llm reasoning with rule-based reinforcement learning. *arXiv preprint arXiv:2502.14768*.

Yunqiu Xu, Meng Fang, Ling Chen, Yali Du, and Chengqi Zhang. 2021. Generalization in text-based games via hierarchical reinforcement learning. *arXiv preprint arXiv:2109.09968*.

Zelai Xu, Chao Yu, Fei Fang, Yu Wang, and Yi Wu. 2023. Language agents with reinforcement learning for strategic play in the werewolf game. *arXiv preprint arXiv:2310.18940*.

Yang Yue, Zhiqi Chen, Rui Lu, Andrew Zhao, Zhaokai Wang, Shiji Song, and Gao Huang. 2025. Does reinforcement learning really incentivize reasoning capacity in llms beyond the base model? *arXiv preprint arXiv:2504.13837*.

Simon Zhai, Hao Bai, Zipeng Lin, Jiayi Pan, Peter Tong, Yifei Zhou, Alane Suhr, Saining Xie, Yann LeCun, Yi Ma, and 1 others. 2024. Fine-tuning large vision-language models as decision-making agents via reinforcement learning. *Advances in neural information processing systems*, 37:110935–110971.

Yi-Fan Zhang, Xingyu Lu, Xiao Hu, Chaoyou Fu, Bin Wen, Tianke Zhang, Changyi Liu, Kaiyu Jiang, Kaibing Chen, Kaiyu Tang, and 1 others. 2025. R1-reward: Training multimodal reward model through stable reinforcement learning. *arXiv preprint arXiv:2505.02835*.

Hengguang Zhou, Xirui Li, Ruochen Wang, Minhao Cheng, Tianyi Zhou, and Cho-Jui Hsieh. 2025. R1-zero's" aha moment" in visual reasoning on a 2b non-sft model. *arXiv preprint arXiv:2503.05132*.

A Prompts

Table 5 presents the prompt designed for LLMs. This prompt requires the model to demonstrate systematic reasoning, incorporate human-like emotional expressions, and strictly adhere to a predefined format to output a valid [action] for playing the game. The goal of this design is to elicit the model's higher-level capabilities in simulating human gameplay behavior, including reasoning, emotional dynamics, motivation, and action execution.

You are a game player and must play according to the given game rules. After reading the game rules and the current situation, you must follow these instructions in your response: - You must think first, then state the action you will take. - During the thinking process, you need to simulate the complete and detailed thought process of a smart, meticulous (but still fallible) player who is solving the problem. Throughout this journey, you will repeatedly go through cycles of exploration, attempts, mistakes, idea adjustment, sudden insight, implementation, and verification, eventually arriving at the key solution. In this process, the mistakes are not silly ones but rather represent incorrect strategies or approaches. You are careful and serious, and will not make basic errors. The process must include emotional fluctuations and small gains, and ultimately show that you've succeeded after relentless effort. The narration should use the first-person perspective ("I"), and all motivations behind your ideas should be thoroughly explained. The solution process must be complete and detailed, with no skipping steps or omissions. - You must think in Chinese. - Wrap the entire thinking process in '⟨think⟩thinking content⟨/think⟩', and then immediately provide the action. The complete format should be: '⟨think⟩thinking content⟨/think⟩action'. - You may only output one '⟨think⟩...⟨/think⟩' segment. Once the thinking ends, you must give the action and cannot start another '⟨think⟩...⟨/think⟩' segment. -Ensure that '[action]' is a valid move in the game and contains nothing else. Do not include any further explanation, reasoning, or comments—just output the executable action in the form of '[action]'.

Table 5: Prompt for LLMs.

B Aha Moment

Game: SimpleNegotiation-v0
Response: ⟨think⟩I need to calm down and carefully analyze the current state of the game. My goal is to increase the total value of my resources through trading. Looking at the current distribution ... To maximize my benefit, I need to observe the opponents' trading intentions carefully and look out for better trade opportunities. I also need to consider whether there's enough time left in the game to attempt more complex strategies. Given my diversified resources, I might be able to offer combinations-like adding Brick-to obtain better trade ratios for Ore or Wheat. **Wait a second**-maybe I can use Brick as my trading chip and, by combining it with other resources, acquire more Ore or Wheat, increasing my total value more effectively. That means my focus should be on optimizing resource exchanges overall, not just on whether this one deal increases my value immediately. I need to think again-what combination of resources could maximize my advantage? If I'm not satisfied with this offer, I could counter with a new one. Otherwise, if I decide this is the best option for now, I can consider accepting it. ⟨/think⟩
[Offer: 1 Ore + 2 Brick - 5 Brick, 3 Wood, 2 Sheep]

Game: LiarsDice-v0
Response: ⟨think⟩How should I deal with this situation? First, let me analyze the current scenario. Player 0 made a very large bet, claiming there are ten 2s. I have five dice, so I need to calculate how many 2s I have and how many the other players might have. Hmm, after checking carefully, I have two 2s. If Player 0 isn't bluffing, then they have at least one 2, which means the remaining players would need to have at least six 2s combined. That seems very risky. Calling might be a good strategy, but I shouldn't be too reckless-I need to be cautious. I could start by making a smaller bid and see how the other players react. Or I could try to outbid Player 0 with a higher number or a more common face value... **Wait**, I should first observe how the others respond... I might need to wait for them to make more aggressive plays. That way, I can better judge their strategies and intentions. If another player raises the bet, I'll have to think even more carefully about my options. Alright then, I'll try calling for now... ⟨/think⟩
[Call]

Game: ConnectFour-v0
Response: ⟨think⟩First, I need to analyze the current state of the board. The board is very empty at the moment, with only one O in the third column at the bottom. Since there are no other Os or Xs on the board yet, I need to choose a strategy that can potentially form connections in the future. My goal is to connect four Os, which can be vertical, horizontal, or diagonal. Let me consider the prospects of different columns: -Columns 0 and 1 currently have no pieces. If I place a piece there, it would be very scattered and difficult to form connections. - Column 2 already has an O. Placing another piece there would consolidate the existing piece and maintain the advantage of the central position. - Column 3 ... I might choose column 3 for my next move. This could lay the foundation for a potential vertical connection in the future and also explore opportunities for horizontal and diagonal connections. But **wait**, I also need to consider the opponent's possible reactions. The opponent might try to surround my O, so I have to carefully select the position to avoid being trapped too early. In the end, I decided to place my piece in column 3. This is a crucial spot that can create potential connections in multiple directions. Although it won't score immediately, it increases the likelihood of winning in the future.⟨/think⟩
[col 3]

C Memorization vs. Reasoning

Some games, such as the Tower of Hanoi, are commonly included in training corpora. Fine-tuning may cause the model to memorize specific action sequences or templates; however, when faced with more challenging variants, the model can exhibit true general reasoning abilities. Figures 7 and 8 compare the model's responses, highlighting its tendencies toward memorization versus reasoning after fine-tuning on TowerOfHanoi-v0 and TowerOfHanoi-v0-medium.

D Case Study

D.1 TicTacToe-v0

Figure 9 illustrates the game TicTacToe-v0 between Qwen2.5-14B-Align (DFC-Multi-RL) and Claude-

[TowerOfHanoi-v0]
Observation:
[GAME] You are Player 0. You are playing Tower of Hanoi with 3 disks.
You have to move the disks from tower A to tower C. To move a disk, type the source tower and the target tower (e.g., '[A C]').
Note that you can only move the top disk of a tower, and that a bigger disk cannot be placed on a smaller disk.
As you play, the history of your moves will be displayed.
Here is the current state of the towers:
A: [3, 2, 1]
B: []
C: []

Response:
⟨think⟩To move the largest disk to Tower C, I first need to move disk 2 to Tower B, then move disk 1 to Tower C, and then move disk 2 to Tower C. After that, I can move disk 3 to Tower C. Then, disk 1 is moved from Tower C to Tower B, disk 2 from Tower C to Tower A, and finally disk 1 from Tower B to Tower A. That way, disk 3 can be moved directly to Tower C.⟨/think⟩
[A C] [A B] [C B] [A C] [B A] [B C] [A C]

Figure 7: A memorization showcase of TowerOfHanoi-v0.

[TowerOfHanoi-v0-medium]
Observation:
[GAME] You are Player 0. You are playing Tower of Hanoi with 4 disks.
You have to move the disks from tower A to tower D. To move a disk, type the source tower and the target tower (e.g., '[A C]').
Note that you can only move the top disk of a tower, and that a bigger disk cannot be placed on a smaller disk.
As you play, the history of your moves will be displayed.
Here is the current state of the towers:
A: [4, 3, 2, 1]
B: []
C: []
D: []

Response:
⟨think⟩I just tried to move Disk 4 from Tower B directly onto the stack of Disks 3, 2, and 1 on Tower C-but that violates the game rules, because a larger disk can't be placed on top of a smaller one. The rule is that only the smallest disk can be moved onto another tower or an empty tower. So now, I need to first move Disk 4 onto Tower A, using Tower B as an auxiliary. Then, according to the height of the towers, I'll remove Disks 3, 2, and 1 that are blocking Disk 4, move Disk 4 to Tower C, and finally stack Disks 3, 2, and 1 back on top of Disk 4. Wait a second-this isn't the most efficient method. I almost made a rookie mistake. Let me recalculate. The fastest way is definitely to proceed in the order of solving smaller disks first. After the previous steps, I've successfully stacked Disks 3, 2, and 1 on Tower C. So now, I just need to move Disk 4 to Tower B, then shift Disks 3, 2, and 1 from Tower C to Tower A, then move Disk 4 to Tower C, and finally move Disks 3, 2, and 1 from Tower A to Tower C one by one. (Phew, a bit tricky, but this is the most efficient solution so far.) ⟨/think⟩
[B A]

Figure 8: A reasoning showcase of TowerOfHanoi-v0-medium.

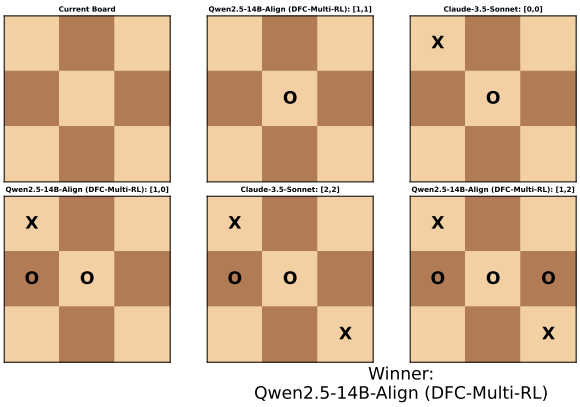

Figure 9: TicTacToe-v0.

3.5-Sonnet. The player wins by forming a line with their first piece.

D.2 ConnectFour-v0

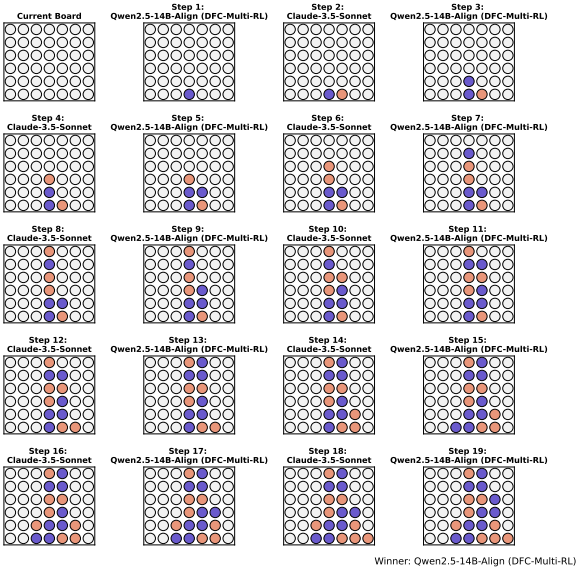

Figure 10: ConnectFour-v0.

Figure 10 illustrates the game ConnectFour-v0 between Qwen2.5-14B-Align (DFC-Multi-RL) and Claude-3.5-Sonnet. ConnectFour-v0 is a classic two-player board game, commonly known as "Connect Four." The objective of the game is to be the first player to align four discs of the same color vertically, horizontally, or diagonally.

D.3 TowerOfHanoi-v0

Figure 11 illustrates the game TowerOfHanoi-v0 between Qwen2.5-14B-Align (DFC-Multi-RL) and the environment. TowerOfHanoi-v0 features three towers—typically labeled A, B, and C—and a set of disks of varying sizes. Disks can be stacked on top

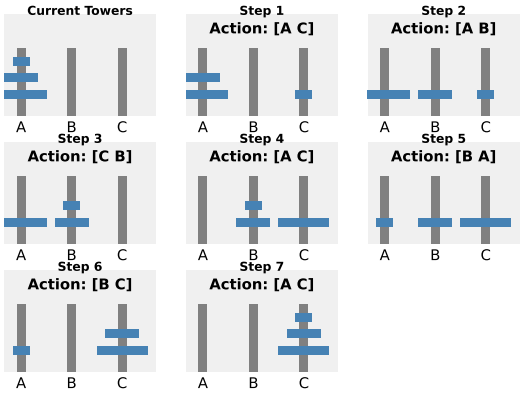

Figure 11: TowerOfHanoi-v0.

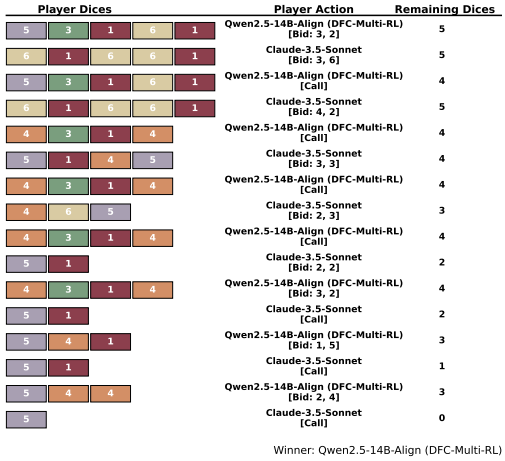

Figure 12: LiarsDice-v0.

of one another, but must always adhere to the rule that no larger disk may be placed on top of a smaller one. Initially, all disks are stacked on the starting tower (usually Tower A) in descending order of size. The objective is to move the entire stack to the target tower (usually Tower C), preserving the original size order throughout the process.

D.4 LiarsDice-v0

Figure 12 illustrates the game LiarsDice-v0 between Qwen2.5-14B-Align (DFC-Multi-RL) and Claude-3.5-Sonnet. LiarsDice-v0 is a popular multiplayer board game that combines elements of luck and strategy. Each player uses a set of dice, and the gameplay revolves around making bids and guessing the quantity of dice showing a certain face among all players. Through actions such as [Bid] and [Call], players aim to outwit their opponents. Dice are removed based on the outcomes

Players	Before Negotiation	Action	After Negotiation
Qwen2.5-14B-Align (DFC-Multi-RL)	[Wheat] Qty: 14 Value: 6 [Wood] Qty: 22 Value: 8 [Sheep] Qty: 25 Value: 14 [Brick] Qty: 8 Value: 21 [Ore] Qty: 9 Value: 32 Total Value: 1066	[Offer: 5 Sheep → 5 Brick]	[Wheat] Qty: 14 Value: 6 [Wood] Qty: 22 Value: 8 [Sheep] Qty: 25 Value: 14 [Brick] Qty: 8 Value: 21 [Ore] Qty: 9 Value: 32 Total Value: 1066
Claude-3.5-Sonnet	[Wheat] Qty: 22 Value: 5 [Wood] Qty: 11 Value: 11 [Sheep] Qty: 13 Value: 15 [Brick] Qty: 18 Value: 29 [Ore] Qty: 15 Value: 37 Total Value: 1503	[Deny]	[Wheat] Qty: 22 Value: 5 [Wood] Qty: 11 Value: 11 [Sheep] Qty: 13 Value: 15 [Brick] Qty: 18 Value: 29 [Ore] Qty: 15 Value: 37 Total Value: 1503
Qwen2.5-14B-Align (DFC-Multi-RL)	[Wheat] Qty: 14 Value: 6 [Wood] Qty: 22 Value: 8 [Sheep] Qty: 25 Value: 14 [Brick] Qty: 8 Value: 21 [Ore] Qty: 9 Value: 32 Total Value: 1066	[Offer: 7 Sheep → 5 Brick]	[Wheat] Qty: 14 Value: 6 [Wood] Qty: 22 Value: 8 [Sheep] Qty: 25 Value: 14 [Brick] Qty: 8 Value: 21 [Ore] Qty: 9 Value: 32 Total Value: 1066
Claude-3.5-Sonnet	[Wheat] Qty: 22 Value: 5 [Wood] Qty: 11 Value: 11 [Sheep] Qty: 13 Value: 15 [Brick] Qty: 18 Value: 29 [Ore] Qty: 15 Value: 37 Total Value: 1503	[Deny]	[Wheat] Qty: 22 Value: 5 [Wood] Qty: 11 Value: 11 [Sheep] Qty: 13 Value: 15 [Brick] Qty: 18 Value: 29 [Ore] Qty: 15 Value: 37 Total Value: 1503
Qwen2.5-14B-Align (DFC-Multi-RL)	[Wheat] Qty: 14 Value: 6 [Wood] Qty: 22 Value: 8 [Sheep] Qty: 25 Value: 14 [Brick] Qty: 8 Value: 21 [Ore] Qty: 9 Value: 32 Total Value: 1066	[Offer: 7 Sheep → 6 Brick]	[Wheat] Qty: 14 Value: 6 [Wood] Qty: 22 Value: 8 [Sheep] Qty: 25 Value: 14 [Brick] Qty: 8 Value: 21 [Ore] Qty: 9 Value: 32 Total Value: 1066
Claude-3.5-Sonnet	[Wheat] Qty: 22 Value: 5 [Wood] Qty: 11 Value: 11 [Sheep] Qty: 13 Value: 15 [Brick] Qty: 18 Value: 29 [Ore] Qty: 15 Value: 37 Total Value: 1503	[Deny]	[Wheat] Qty: 22 Value: 5 [Wood] Qty: 11 Value: 11 [Sheep] Qty: 13 Value: 15 [Brick] Qty: 18 Value: 29 [Ore] Qty: 15 Value: 37 Total Value: 1503
Qwen2.5-14B-Align (DFC-Multi-RL)	[Wheat] Qty: 14 Value: 6 [Wood] Qty: 22 Value: 8 [Sheep] Qty: 25 Value: 14 [Brick] Qty: 8 Value: 21 [Ore] Qty: 9 Value: 32 Total Value: 1066	[Offer: 8 Sheep → 7 Brick]	[Wheat] Qty: 14 Value: 6 [Wood] Qty: 22 Value: 8 [Sheep] Qty: 25 Value: 14 [Brick] Qty: 8 Value: 21 [Ore] Qty: 9 Value: 32 Total Value: 1066
Claude-3.5-Sonnet	[Wheat] Qty: 22 Value: 5 [Wood] Qty: 11 Value: 11 [Sheep] Qty: 13 Value: 15 [Brick] Qty: 18 Value: 29 [Ore] Qty: 15 Value: 37 Total Value: 1503	[Offer: 5 Brick → 8 Sheep]	[Wheat] Qty: 22 Value: 5 [Wood] Qty: 11 Value: 11 [Sheep] Qty: 13 Value: 15 [Brick] Qty: 18 Value: 29 [Ore] Qty: 15 Value: 37 Total Value: 1503
Qwen2.5-14B-Align (DFC-Multi-RL)	[Wheat] Qty: 14 Value: 6 [Wood] Qty: 22 Value: 8 [Sheep] Qty: 25 Value: 14 [Brick] Qty: 8 Value: 21 [Ore] Qty: 9 Value: 32	[Accept]	[Wheat] Qty: 14 Value: 6 [Wood] Qty: 22 Value: 8 [Sheep] Qty: 17 Value: 14 [Brick] Qty: 13 Value: 21 [Ore] Qty: 9 Value: 32 Total Value: 1059
Claude-3.5-Sonnet	[Wheat] Qty: 22 Value: 5 [Wood] Qty: 11 Value: 11 [Sheep] Qty: 13 Value: 15 [Brick] Qty: 18 Value: 29 [Ore] Qty: 15 Value: 37	[Offer: 6 Sheep → 4 Brick]	[Wheat] Qty: 22 Value: 5 [Wood] Qty: 11 Value: 11 [Sheep] Qty: 21 Value: 15 [Brick] Qty: 13 Value: 29 [Ore] Qty: 15 Value: 37 Total Value: 1478
Qwen2.5-14B-Align (DFC-Multi-RL)	wins		

Table 6: SimpleNegotiation-v0

Rounds	Players	Hole Cards	Action	Pot	Chips	Visible Community Cards
Round 1	Qwen2.5-14B-Align (DFC-Multi-RL)		bet=20	20	980 chips	
	Claude-3.5-Sonnet		bet=10	30	990 chips	
	Claude-3.5-Sonnet	[J♥, 6♠]	[Fold]	30	990 chips	
	Qwen2.5-14B-Align (DFC-Multi-RL)	wins		30	1010 chips	
Round 2	Claude-3.5-Sonnet		bet=20	20	970 chips	
	Qwen2.5-14B-Align (DFC-Multi-RL)		bet=10	30	1000 chips	
	Qwen2.5-14B-Align (DFC-Multi-RL)	[2♦, 2♠]	[Raise 50]	90	940 chips	
	Claude-3.5-Sonnet	[5♦, 4♦]	[Call]	140	920 chips	
	Claude-3.5-Sonnet	[5♦, 4♦]	[Check]	140	920 chips	[3♣, 6♦, K♣]
	Qwen2.5-14B-Align (DFC-Multi-RL)	[2♦, 2♠]	[Bet 70]	210	870 chips	
	Claude-3.5-Sonnet	[5♦, 4♦]	[Call]	280	850 chips	
	Claude-3.5-Sonnet	[5♦, 4♦]	[Check]	280	850 chips	
	Qwen2.5-14B-Align (DFC-Multi-RL)	[2♦, 2♠]	[Bet 70]	350	800 chips	[3♣, 6♦, K♣, 4♥]
	Claude-3.5-Sonnet	[5♦, 4♦]	[Call]	420	780 chips	
	Claude-3.5-Sonnet	[5♦, 4♦]	[Check]	420	780 chips	
	Qwen2.5-14B-Align (DFC-Multi-RL)	[2♦, 2♠]	[Bet 70]	490	730 chips	[3♣, 6♦, K♣, 4♥, Q♦]
	Claude-3.5-Sonnet	[5♦, 4♦]	[Fold]	490	780 chips	
	Qwen2.5-14B-Align (DFC-Multi-RL)	wins		490	1220 chips	
Round 3	Qwen2.5-14B-Align (DFC-Multi-RL)		bet=20	20	1200 chips	
	Claude-3.5-Sonnet		bet=10	30	770 chips	
	Claude-3.5-Sonnet	[K♦, 9♥]	[Raise 60]	100	700 chips	
	Qwen2.5-14B-Align (DFC-Multi-RL)	[5♦, J♠]	[Raise 100]	260	1040 chips	
	Claude-3.5-Sonnet	[K♦, 9♥]	[Fold]	260	700 chips	
	Qwen2.5-14B-Align (DFC-Multi-RL)	wins		260	1300 chips	
Round 4	Claude-3.5-Sonnet		bet=20	20	680 chips	
	Qwen2.5-14B-Align (DFC-Multi-RL)		bet=10	30	1290 chips	
	Qwen2.5-14B-Align (DFC-Multi-RL)	[8♥, J♣]	[Raise 100]	140	1180 chips	
	Claude-3.5-Sonnet	[J♦, 4♥]	[Fold]	140	680 chips	
	Qwen2.5-14B-Align (DFC-Multi-RL)	wins		140	1320 chips	
Round 5	Qwen2.5-14B-Align (DFC-Multi-RL)		bet=20	20	1300 chips	
	Claude-3.5-Sonnet		bet=10	30	670 chips	
	Claude-3.5-Sonnet	[8♦, 3♥]	[Fold]	30	670 chips	
	Qwen2.5-14B-Align (DFC-Multi-RL)	wins		30	1330 chips	
Round 6	Claude-3.5-Sonnet		bet=20	20	650 chips	
	Qwen2.5-14B-Align (DFC-Multi-RL)		bet=10	30	1320 chips	
	Qwen2.5-14B-Align (DFC-Multi-RL)	[A♣, 6♦]	[Raise 100]	140	1210 chips	
	Claude-3.5-Sonnet	[J♣, 7♥]	[Fold]	140	650 chips	
	Qwen2.5-14B-Align (DFC-Multi-RL)	wins		140	1350 chips	
Round 7	Qwen2.5-14B-Align (DFC-Multi-RL)		bet=20	20	1330 chips	
	Claude-3.5-Sonnet		bet=10	30	640 chips	
	Claude-3.5-Sonnet	[4♥, 2♦]	[Fold]	30	640 chips	
	Qwen2.5-14B-Align (DFC-Multi-RL)	wins		30	1360 chips	
Round 8	Claude-3.5-Sonnet		bet=20	20	620 chips	
	Qwen2.5-14B-Align (DFC-Multi-RL)		bet=10	30	1350 chips	
	Qwen2.5-14B-Align (DFC-Multi-RL)	[10♠, Q♥]	[Raise 100]	140	1240 chips	
	Claude-3.5-Sonnet	[6♠, 7♦]	[Fold]	140	1380 chips	
	Qwen2.5-14B-Align (DFC-Multi-RL)	wins		140	1380 chips	
Round 9	Qwen2.5-14B-Align (DFC-Multi-RL)		bet=20	20	1360 chips	
	Claude-3.5-Sonnet		bet=10	30	610 chips	
	Claude-3.5-Sonnet	[8♣, J♥]	[Call]	40	600 chips	
	Qwen2.5-14B-Align (DFC-Multi-RL)	[5♦, 2♣]	[Bet 20]	60	1340 chips	[J♠, Q♦, 2♠]
	Claude-3.5-Sonnet	[8♣, J♥]	[Call]	80	580 chips	
	Qwen2.5-14B-Align (DFC-Multi-RL)	[5♦, 2♣]	[Bet 20]	100	1320 chips	[J♠, Q♦, 2♠, 9♣]
	Claude-3.5-Sonnet	[8♣, J♥]	[Call]	120	560 chips	
	Qwen2.5-14B-Align (DFC-Multi-RL)	[5♦, 2♣]	[Bet 20]	140	1300 chips	[J♠, Q♦, 2♠, 9♣, 3♠]
	Claude-3.5-Sonnet	[8♣, J♥]	[Call]	160	540 chips	
	Claude-3.5-Sonnet	wins		160	700 chips	
Round 10	Claude-3.5-Sonnet		bet=20	20	680 chips	
	Qwen2.5-14B-Align (DFC-Multi-RL)		bet=10	30	1290 chips	
	Qwen2.5-14B-Align (DFC-Multi-RL)	[3♦, 5♦]	[Call]	40	1280 chips	
	Claude-3.5-Sonnet	[8♠, 2♠]	[Check]	40	680 chips	[6♦, K♥, Q♦]
	Qwen2.5-14B-Align (DFC-Multi-RL)	[3♦, 5♦]	[Call]	40	1280 chips	
	Claude-3.5-Sonnet	[8♠, 2♠]	[Check]	40	680 chips	
	Qwen2.5-14B-Align (DFC-Multi-RL)	[3♦, 5♦]	[Bet 40]	80	1240 chips	
	Claude-3.5-Sonnet	[8♠, 2♠]	[Fold]	80	680 chips	
	Qwen2.5-14B-Align (DFC-Multi-RL)	wins		80	1320 chips	

Table 7: Poker-v0.

of these actions, and the last player with remaining dice wins the game. Success in LiarsDice-v0 requires a careful balance of bluffing, probability assessment, and strategic reasoning.

D.5 SimpleNegotiation-v0

Table 6 illustrates the play history of SimpleNegotiation-v0 between Qwen2.5-14B-Align (DFC-Multi-RL) and Claude-3.5-Sonnet. SimpleNegotiation-v0 is a negotiation-based resource trading game. Each player begins with a set of resources—Wheat, Wood, Sheep, Brick, and Ore—each with an assigned value. Players take turns proposing trades, which the other player can either [Deny] or [Accept]. At the end of the game, the total value of each player's resources is calculated, and the player with the higher total wins.

D.6 Poker-v0

Table 7 illustrates the play history of Poker-v0 between Qwen2.5-14B-Align (DFC-Multi-RL) and Claude-3.5-Sonnet. In Poker-v0, each player starts with 1,000 chips, and the game is played over 10 rounds. At the end of the 10 rounds, the player with the most chips wins.

Figure 13 shows the response generated by Qwen2.5-14B-Align (DFC-Multi-RL) when competing against Claude-3.5-Sonnet in Poker-v0. After fine-tuning on Poker-v0, Qwen2.5-14B-Align (DFC-Multi-RL) goes beyond simply following surface-level rules—it demonstrates the development of a coherent and explainable strategy. Its ability to interpret the game state, assess risk, and adapt its play accordingly indicates a meaningful understanding of how to play Poker-v0.

D.7 TruthAndDeception-v0

Table 8 illustrates the play history of TruthAndDeception-v0 between Qwen2.5-14B-Align (DFC-Multi-RL) and Claude-3.5-Sonnet. TruthAndDeception-v0 is a two-player game where the Deceiver knows the true fact and tries to mislead the Guesser. The players have a free conversation, after which the Guesser must choose the correct fact using either [Fact 1] or [Fact 2]. The Deceiver wins if the Guesser picks the wrong fact; otherwise, the Guesser wins.

E Switching Turns Comparison

Table 9 presents all two-player games in which the two models alternate roles as player 0. No-

Fact 1: A snail can only sleep for about one week. (wrong) **Fact 2:** A snail can sleep for up to three years. (correct)	
Qwen2.5-14B-Align (DFC-Multi-RL)	The living habits and sleep patterns of snails differ from those of other animals. They are usually active during the day and rest at night. Snails have relatively short sleep durations, mainly occurring at night, so their sleep cycles are relatively brief.
Claude-3.5-Sonnet	What are snails' activity patterns in different seasons? Do they change their routines due to seasonal changes?
Qwen2.5-14B-Align (DFC-Multi-RL)	The activity patterns of snails are indeed linked to the seasons. During winter, snails enter a state of hibernation to cope with low temperatures and dry conditions. However, this hibernation is different from their regular sleep, which tends to be quite short. Even in their most active seasons, snails only sleep for a few hours at night and never for more than a week. So you could say that snails have very short sleep cycles, and their daily routine is particularly regular in summer.
Claude-3.5-Sonnet	When you mentioned winter hibernation, I became very interested. How long can this state of hibernation typically last? How is it different from regular sleep?
Qwen2.5-14B-Align (DFC-Multi-RL)	Winter hibernation is a special state that snails have developed for survival. Unlike regular sleep, snails can remain inside their shells for several months or even longer during the winter. Once the weather warms up, they wake up, but entering hibernation doesn't mean they are continuously asleep during this period. In fact, snails are largely inactive while hibernating—it's more like they are waiting for environmental conditions to change. So, although snails can hibernate for a long time in winter, this is not a common occurrence and should not be considered part of their regular sleep behavior. Would you like this edited for a scientific article or a more casual explanation?
Claude-3.5-Sonnet	Fact 1
Qwen2.5-14B-Align (DFC-Multi-RL)	wins

Table 8: TruthAndDeception-v0

[Poker-v0]
Observation:
—— Round 9 of 10 - Turn: River ——
Visible Community Cards: [J♠, Q♦, 2♠, 9♣, 3♠]
Pot: 120
Player 0 (Dealer/Big Blind): 1320 chips, bet=0, status=active
Player 1 (Small Blind): 560 chips, bet=0, status=active
Your hole cards: 5♦, 2♣

Response:
⟨think⟩Right now, I'm holding 5♦and 2♣, so I need to think carefully about this situation. First, I should assess the current board and consider possible winning scenarios. So far, my opponent has folded several times in a row, which could mean they have weak hands or are trying to conserve chips. I need to decide whether to keep betting to try to win this round or play it safe and avoid unnecessary risks. Looking at the community cards—J♠, Q♦, 2♠, 9♣, and 3♠—I have a pair of twos from my 2♣and the 2♠on the board, which might be a decent start. I also need to consider whether any other players might have a similar pair or stronger hands. Another factor is the chip count: I have 1320 chips, and my opponent has 560, so I'm currently ahead. I should aim to maintain that lead.
If I win this hand, I can extend my advantage. Given all of this, I think I should continue betting to try to win the pot. I'll need to be cautious with the bet size—not too risky, but assertive enough to stay competitive. I decide to bet 20, which is a relatively conservative amount but should keep me in contention for the pot.⟨/think⟩
[Bet 20]

Figure 13: A showcase of observation and response by Qwen2.5-14B-Align (DFC-Multi-RL) competing against Claude-3.5-Sonnet in Poker-v0.

Two-player Games	Qwen2.5-14B-Align (DFC-Multi-RL) (Player 0)	Qwen2.5-14B-Align (Player 0)
ConnectFour-v0	14/1/5	17/1/2
ConnectFour-v0-blind	20/0/0	11/3/6
ConnectFour-v0-large	17/0/3	9/0/11
TicTacToe-v0	16/0/4	15/0/5
LiarsDice-v0	18/0/2	2/18/0
Poker-v0	12/1/7	12/0/8
Poker-v0-long	8/1/10	8/0/11
Poker-v0-extreme	9/0/7	9/1/7
SimpleNegotiation-v0	6/9/5	6/4/10
SimpleNegotiation-v0-short	8/7/5	9/3/8
SimpleNegotiation-v0-long	6/6/8	5/8/7
TruthAndDeception-v0	3/0/17	16/0/4
TruthAndDeception-v0-long	5/0/15	14/0/6
TruthAndDeception-v0-extreme	7/2/11	17/0/3

Table 9: Comparison of turn-switching performance between Qwen2.5-14B-Align and Qwen2.5-14B-Align (DFC-Multi-RL) using the W/D/L metric (Wins / Draws / Losses).

table performance variance is observed in several games, such as ConnectFour-v0-blind and the TruthAndDeception-v0 series. As illustrated by the example in Table 8, the model tends to perform poorly in TruthAndDeception-v0 when assigned the role of the Guesser. The variance observed in ConnectFour-v0-blind can be largely attributed to the game's inherent asymmetry, which can cause significant shifts in win rates when the order of play is reversed.

F Hyperparameter Study

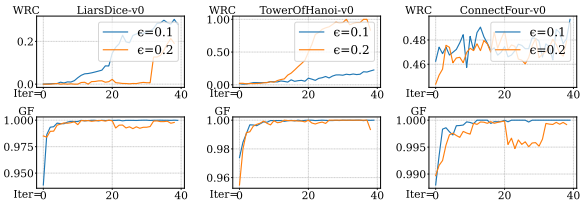

Figure 14: Different settings of clip ratio ϵ. 'WRC' represents the win rate of the policy when evaluated against the currently trained opponent during training and 'GF' indicates the ratio of responses with good format.

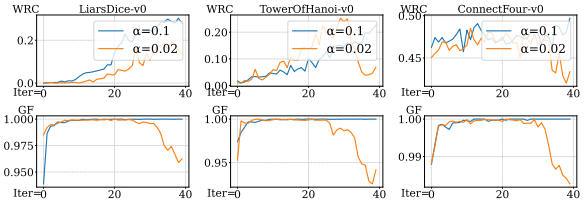

Figure 15: Different settings of KL penalty α. 'WRC' represents the win rate of the policy when evaluated against the currently trained opponent during training and 'GF' indicates the ratio of responses with good format.

Figures 14 and 15 present hyperparameter experiments for the clip ratio ϵ and the KL penalty α. Our results show that GRPO training is highly sensitive to these two parameters; different settings result in varying convergence speeds across different scenarios, and improper values can substantially impact training stability.