Bridging Heuristic Experience and Structural Signals: A Dual-Driven Framework for LLM Reasoning

Anonymous ACL submission

Abstract

Although large language models (LLMs) perform strongly across diverse tasks, they still struggle in complex multi-agent interaction scenarios characterized by information asymmetry, strategic communication, and evolving relations. Reasoning in such environments requires not only semantic understanding but also the ability to capture hidden relational structures and reusable strategic experience. However, existing reasoning enhancement methods rarely model these two aspects jointly. To address this limitation, we propose *SHIELD*, a dual-driven framework that enhances LLM reasoning with structural interaction signals and distilled heuristic experience. Leveraging multi-agent simulation to alleviate data scarcity, *SHIELD* models relational dependencies through interaction graphs and distills reusable insights via difficulty-aware experience abstraction. These complementary signals seamlessly integrate into existing reasoning paradigms, such as Chain-of-Thought (CoT) and ReAct. Experiments on two multi-agent reasoning tasks demonstrate that *SHIELD* consistently achieves superior performance over vanilla prompting and single-signal baselines, effectively balancing error correction and predictive stability across varying difficulty levels.

1 Introduction

Large Language Models (LLMs) have demonstrated remarkable performance across reasoning tasks, including mathematics (Ahn et al., 2024; Wang et al., 2026), coding (Jiang et al., 2026; Zhang et al., 2026a), and logical inference (Plaat et al., 2025; Liu et al., 2025; Ma, 2025). However, reasoning in complex human-centric interaction environments, which requires inferring the latent intents of participants under uncertainty and deceptive contexts, poses a significant challenge. Such scenarios are characterized by information asymmetry, strategic communication, and evolving relational dynamics. This complex context fundamentally presents a heterogeneous evidence integration problem. Successful inference requires a model to not only comprehend local semantic information but also aggregate fragmented, partially observable, and potentially deceptive evidence, including linguistic clues, behavioral signals, and structural relationships among participants.

Existing reasoning enhancement methods remain limited in complex interaction environments. Semantic prompting paradigms, such as Chain-of-Thought (CoT) (Wei et al., 2022) and ReAct (Yao et al., 2022b), exhibit strong semantic reasoning and commonsense capabilities. Knowledge-driven methods further augment LLM reasoning with external knowledge sources, such as Retrieval-Augmented Generation (RAG) (Lewis et al., 2020), improving knowledge utilization and decision support. However, these methods often lack a global view to capture high-order relational dependencies among participants. To address this issue, structure-driven approaches such as MEOW (Wang et al., 2024) leverage graph-based models to capture interaction and relational patterns. Although effective at modeling interaction structures, these methods still struggle to fully capture the dynamic and strategic nature of human interactions.

In human cognition, robust decision-making heavily relies on experience, where individuals distill reusable heuristic rules by reflecting on past successes and failures (Ellis and Davidi, 2005). However, extending such experience mechanisms to strategic interaction environments for LLMs faces a fundamental bottleneck in data acquisition, as authentic interaction data is scarce due to privacy constraints and limited observability. Existing experience-driven frameworks, which typically rely on abundant and easily synthesized single-agent reasoning trajectories (Alazraki et al., 2025; Zhao et al., 2024; Gao et al., 2024), struggle to generalize to complex interactive settings. Consequently,

these methods remain largely restricted to well-structured and non-adversarial reasoning domains.

To bridge these gaps, we propose *SHIELD* (**S**tructural and **H**euristic **I**ntegration for **E**nhanced **L**LM Reasoning and **D**ecision), a dual-driven framework that unifies structural interaction modeling with distilled heuristic experience. To alleviate real-world data scarcity, we leverage multi-agent simulation to construct diverse interaction trajectories. Based on these simulated trajectories, *SHIELD* extracts structural signals to capture latent relational dependencies through dynamic interaction graphs, and distills reusable heuristic insights through a hierarchical, difficulty-aware abstraction process. These two guidances are seamlessly integrated into a staged reasoning pipeline, enabling experience-driven inference followed by structure-informed refinement. We evaluate *SHIELD* on two representative multi-agent reasoning tasks, *Find the Spy* and *Say and Guess*. Experimental results show that *SHIELD* consistently outperforms vanilla prompting baselines and single-signal variants across various LLM backbones. Further analyses on prediction transitions and difficulty levels demonstrate that integrating structural signals and heuristic experience improves both error correction capability and prediction stability.

Our contributions are summarized as follows:

1. To address data scarcity in acquiring reusable reasoning experience, we leverage simulation and propose a hierarchical, difficulty-aware insight extraction method.

2. We propose *SHIELD*, which integrates structural interaction modeling with distilled heuristic experience for improved reasoning in complex interactive environments.

3. Experiments on two multi-agent reasoning tasks show that *SHIELD* consistently outperforms strong baselines, confirming the complementarity of structural and experiential guidance.

2 Related Work

Standard Reasoning Paradigms. The rapid development of LLMs has led to diverse prompting paradigms to elicit complex reasoning. CoT (Wei et al., 2022) and its variants (Kojima et al., 2022; Lyu et al., 2023; Zhang et al., 2023) encourage step-by-step problem decomposition. To improve robustness, Self-consistency (Wang et al., 2023b) aggregates multiple reasoning paths, while Tree-of-Thoughts (ToT) (Yao et al., 2023) enables structured exploration over intermediate reasoning states. Furthermore, ReAct (Yao et al., 2022b) and Plan-and-Solve (Wang et al., 2023a) integrate reasoning with environment interaction and explicit planning. These methods are often enhanced by In-Context Learning (ICL) (Dong et al., 2024) via few-shot demonstrations. Rather than replacing these paradigms, our framework complements and enhances them by providing additional structural and heuristic guidance.

Reasoning with External Guidance. Beyond prompting, LLMs can be enhanced via parameter adaptation or knowledge augmentation. The former applies Fine-tuning (FT), such as RLHF (Ouyang et al., 2022), instruction tuning (Zhang et al., 2026b), and parameter-efficient fine-tuning (PEFT) (Wang et al., 2025b), to internalize reasoning patterns, though constrained by heavy compute and data demands. The latter integrates external resources during inference. RAG (Lewis et al., 2020) grounds models on retrieved knowledge (Fan et al., 2024; Huang and Huang, 2026), while tool learning (Qu et al., 2025b) enables dynamic interaction with APIs (Basu et al., 2024; Qin et al., 2024), solvers (Gou et al., 2024), and code assistants (Wang et al., 2025a). Recent standards like Model Context Protocol (MCP) (Hou et al., 2026) regularize these workflows. However, these methods depend heavily on rich external ecosystems or scarce high-quality data. To alleviate this, MEOW (Wang et al., 2024) leverages simulated data to train expert models that assist reasoning. Inspired by this, we utilize simulated environments to distill structural guidance and reusable heuristic experience, providing a more interpretable and data-efficient enhancement framework.

Experience-driven Reasoning. Recent studies improve LLM reasoning via self-reflection and trajectory-based experience accumulation. Reflexion (Shinn et al., 2023), Self-Refine (Madaan et al., 2023), and S^3C-MATH (Yan et al., 2025) use self-generated feedback for iterative correction. Beyond immediate reflection, ExpeL (Zhao et al., 2024) and SEGPT (Gao et al., 2024) learn reusable reasoning strategies from trial-and-error trajectories, while ExpeTrans (Gao et al., 2025) explores the transferability of such experience across domains. Song et al. (2024) further refines agent trajectories to improve exploration efficiency. Moreover, mistake-centric learning emphasizes learning from

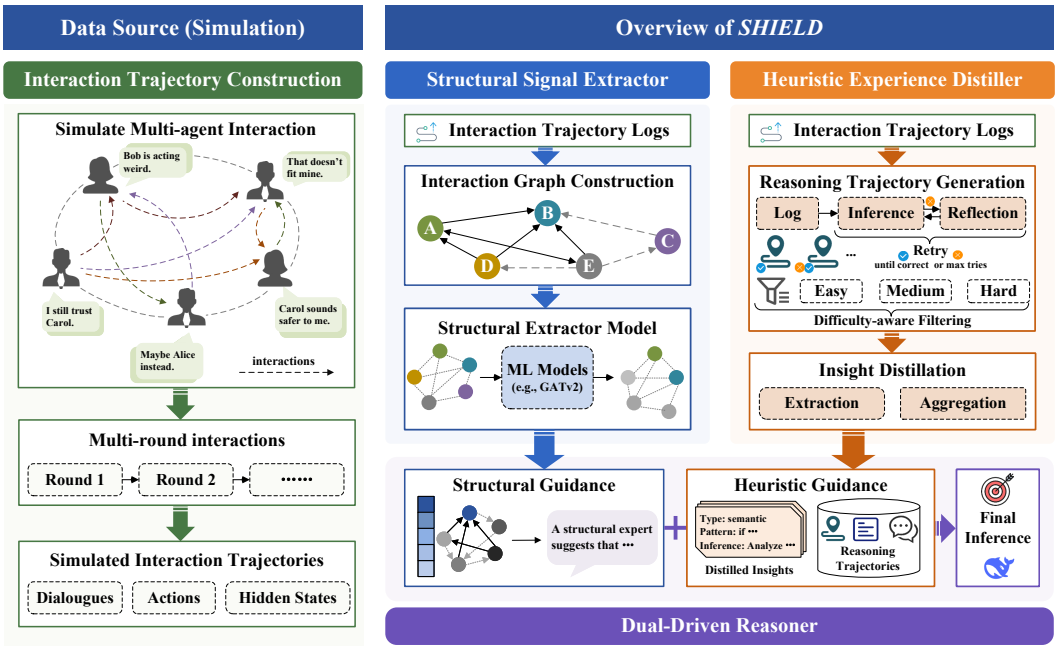

Figure 1: Overview of *SHIELD*. Since real-world interaction data is scarce, we construct interaction trajectories via simulation. *SHIELD* comprises three modules: a structural signal extractor that captures task-specific topological patterns, a heuristic experience distiller that summarizes reusable high-level strategies, and a dual-driven reasoner that integrates both signals for accurate and robust LLM reasoning.

failures. Empirical studies (Tong et al., 2024; An et al., 2024b; Alazraki et al., 2025) show that analyzing prior mistakes improves reasoning, while TRAN (Yang et al., 2023) and LEAP (Zhang et al., 2024) extract generalized rules from negative examples. These paradigms even extend to alignment (Chen et al., 2024), fine-tuning (An et al., 2024a), and tool-use optimization (Qu et al., 2025a).

Despite their success, most methods are evaluated in structured settings such as mathematical reasoning (Cobbe et al., 2021), textual inference (Yang et al., 2018), and interactive decision-making (Yao et al., 2022a), where inputs are well-defined, feedback is deterministic, and trajectories are easy to construct. In contrast, complex interaction scenarios involve higher uncertainty, deception, and relational dependency, making realistic trajectory construction more challenging. Our framework addresses this gap by leveraging simulation to jointly model structural signals and heuristic experience for better reasoning.

3 Approach

In this section, we introduce our proposed method, *SHIELD*, a dual-driven framework designed to enhance LLM reasoning with both structural and experiential knowledge. The overall architecture of *SHIELD* is illustrated in Figure 1.

3.1 Problem Definition

Reasoning in complex human-centric interaction systems is challenging due to participants' latent intents and deceptive behaviors. To make the problem tractable, we represent interactions as observable trajectories and formulate reasoning as inference over them. Let $\mathcal{U} = \{u_1, u_2, \ldots, u_M\}$ denote the set of M participants. An interaction trajectory under a specific task instance is defined as $\mathcal{T} = (t_1, t_2, \cdots, t_N)$, where each interaction event $t_i = \langle r_i, u_i, R_i, s_i, a_i \rangle$ comprises the round index r_i, the acting participant $u_i \in \mathcal{U}$, the set of involved counterparts $R_i \subseteq \mathcal{U}$, the generated utterance s_i, and the corresponding observable action a_i.

Given $\mathcal{T}$, the objective is to infer its corresponding ground-truth label $\mathbf{y} \in \mathcal{Y}$, where $\mathcal{Y}$ denotes the task label space. Effective reasoning requires integrating complementary evidence: semantic clues extracted from utterances and actions, structural signals capturing latent interaction patterns like alliances or conflicts, and heuristic experience providing reusable strategies distilled from historical reasoning trajectories. To address this challenge, we propose *SHIELD*, a dual-driven framework that enhances LLM reasoning through complementary

structural and experiential guidance.

3.2 Interaction Trajectory Construction

Structural and heuristic learning requires large-scale, fully labeled interaction trajectories. Although such data is ideally derived from real-world complex human systems, data acquisition faces privacy and observability barriers. Following Wang et al. (2024), we build interaction datasets via multi-agent simulation covering two typical reasoning tasks: *Find the Spy*, which requires identifying hidden adversary agents, and *Say and Guess*, which predicts whether guessing players can infer target words from descriptions. Both tasks demand reasoning over partial, sequential, and strategic interactions. During simulation, we vary game configurations to generate diverse interaction trajectories. We record observable events, including utterances and actions, while latent states such as role assignments and final outcomes provide ground-truth supervision signals. Compared to Wang et al. (2024), which trains standalone expert models, our framework leverages these trajectories to jointly drive structural modeling and heuristic experience distillation. Further details of the simulation setup are provided in Appendix A.

3.3 Structural Signal Extractor

Agent interactions naturally induce rich relational structures hard to capture from sequential text alone. Therefore, we convert each interaction trajectory into a behavioral topology to explicitly model agent dependencies. Graph Neural Networks (GNNs) (Wu et al., 2021) offer a powerful paradigm for learning from such relational data.

Graph Construction. Given an interaction trajectory $\mathcal{T}$ over the agent set $\mathcal{U}$, we construct a heterogeneous interaction graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where each node $v_m \in \mathcal{V}$ corresponds to an agent $u_m \in \mathcal{U}$ and is initialized with a one-hot identity vector $\mathbf{x}_m \in \mathbb{R}^M$. For each interaction event t_i ($i \in \{1, 2, \cdots, N\}$), a directed edge $(v_i, v_j) \in \mathcal{E}$ is instantiated if $u_j \in R_i$, representing an interaction from u_i to u_j. To preserve task-specific semantics and temporal dynamics, each edge is further assigned an attribute $\tau = (a_i, r_i)$ that encodes the action type and its temporal stage. The constructed graph $\mathcal{G}$ serves as a dynamic representation of interpersonal behaviors and inherits the ground-truth label $\mathbf{y}$ from its source trajectory.

Structural Extraction. The proposed structural extractor is model-independent and can flexibly incorporate various architectures according to the target task and graph schema. During training, the extractor learns a mapping from the interaction graph $\mathcal{G}$ to the ground-truth label $\mathbf{y}$ by optimizing a task-specific objective. Specifically, *Find the Spy* is formulated as a multi-class classification problem, where each class corresponds to a candidate player being the hidden spy. While *Say and Guess* is modeled as a node-level binary outcome prediction problem, indicating whether guessing players can successfully infer the target word from the describer's description. Formally, let $f(\cdot; \Theta)$ denote the structural extractor. Its prediction over the label space is defined as: $P(\tilde{\mathbf{y}} \mid \mathcal{G}) = f(\mathcal{G}; \Theta)$. In this work, we instantiate the extractor with Graph Attention Network v2 (GATv2) (Brody et al., 2022), leveraging its ability to dynamically weight different strategic interactions. Further architectural details are provided in Appendix B.2.

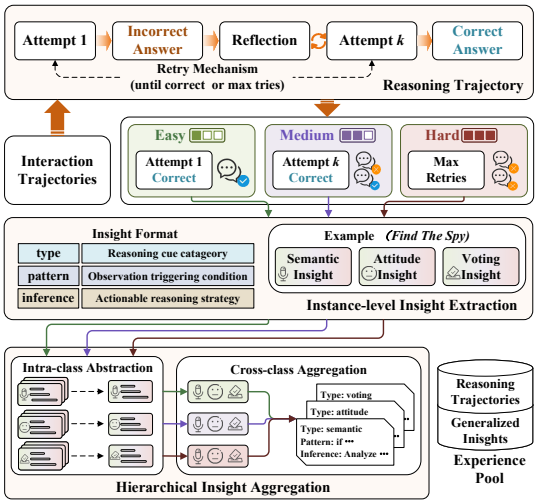

Figure 2: Heuristic experience distillation in *SHIELD*. Reasoning trajectories from simulated interactions are partitioned by difficulty and distilled into reusable insights through extraction and aggregation.

3.4 Heuristic Experience Distiller

Evidence shows that historical reasoning experience enhances LLM reasoning in single-agent settings (Zhao et al., 2024; Gao et al., 2024). As shown in Figure 2, we extend this idea to multi-agent systems by distilling reusable heuristics from simulated interaction trajectories.

Reasoning Trajectory Generation. Let $\mathcal{D} = \{(\mathcal{T}_i, \mathbf{y}_i)\}_{i=1}^{L}$ denote the set of labeled interaction trajectories. For each $\mathcal{T}_i$, the LLM solves the task in an iterative manner. Its k-th attempt is

denoted as $r_i^{(k)} = (c_i^{(k)}, \hat{\mathbf{y}}_i^{(k)})$, where $c_i^{(k)}$ and $\hat{\mathbf{y}}_i^{(k)}$ denote the reasoning chain and predicted label, respectively. If $\hat{\mathbf{y}}_i^{(k)} \neq \mathbf{y}_i$, the model reflects to revise its reasoning (Shinn et al., 2023). The process continues until a correct prediction is obtained or a maximum retry budget K is reached. The resulting reasoning trajectory is defined as $\mathcal{R}_i = (\mathcal{T}_i, r_i^{(1)}, \cdots, r_i^{(\min(o_i,K))})$, where o_i is the first successful attempt index ($o_i = K+1$ indicates failure). We partition these trajectories by difficulty into $\mathcal{C}_{\text{easy}} = \{\mathcal{R}_i \mid o_i = 1\}$, $\mathcal{C}_{\text{medium}} = \{\mathcal{R}_i \mid 1 < o_i \leq K\}$, and $\mathcal{C}_{\text{hard}} = \{\mathcal{R}_i \mid o_i = K+1\}$.

Instance-level Insight Extraction. To categorize heterogeneous reasoning signals across tasks, we define a task-specific observation type space, for example, semantic consistency, interactive attitudes, and voting dynamics in *Find the Spy*. For each trajectory $\mathcal{R}_i \in \mathcal{C}_d$ ($d \in \{\text{easy}, \text{medium}, \text{hard}\}$), an LLM extracts an insight $z_i = (\text{type}_i, \text{pattern}_i, \text{inference}_i)$, where $\text{type}_i \in S$ is the signal category, pattern_i is the triggering condition, and inference_i is the actionable reasoning strategy. The extracted insight set is denoted as $\mathcal{I}_d = \{z_i \mid \mathcal{R}_i \in \mathcal{C}_d\}$. To reduce redundancy and computation, stratified sampling can be applied to $\mathcal{C}_d$ before extraction.

Hierarchical Insight Aggregation. To filter redundant insights, we perform a two-stage aggregation. First, intra-class abstraction groups insights in $\mathcal{I}_d$ by type $s \in S$, yielding representative heuristics $\tilde{\mathcal{I}}_d = \bigcup_{s \in S} \text{Abstract}(\{z_i \mid z_i \in \mathcal{I}_d,\ \text{type}_i = s\})$. Subsequently, cross-class aggregation combines all difficulty-level sets to perform a global abstraction, obtaining a compact global heuristic set $\mathcal{I}^* = \text{Abstract}(\tilde{\mathcal{I}}_{\text{easy}} \cup \tilde{\mathcal{I}}_{\text{medium}} \cup \tilde{\mathcal{I}}_{\text{hard}})$, which ultimately forms a unified experience pool together with $\mathcal{C}_d$.

3.5 Dual-driven Reasoner

To combine these complementary guidance sources, we propose a staged, inference-then-refinement framework. Given an interaction trajectory $\mathcal{T}$, the distilled insight set $\mathcal{I}^*$ provides high-level reasoning guidelines, while k sampled reasoning trajectories from $\mathcal{C}_d$ serve as few-shot demonstrations to anchor the reasoning process. This experience-driven inference yields an initial prediction $\hat{\mathbf{y}}^{(0)}$. Subsequently, the structural extractor's output $P(\tilde{\mathbf{y}} \mid \mathcal{G})$ is converted into a natural language form, such as "A structural expert suggests that <agent> is the most likely spy based on interaction topology". Conditioned on both the initial reasoning trace and the structural signal, the LLM refines its judgment to obtain the final prediction $\hat{\mathbf{y}}^{(1)}$. By sequentially integrating bottom-up heuristic intuition and top-down structural signals, the framework mitigates the risks of semantic hallucination or blind adherence to structural signals, leading to more accurate and robust reasoning.

4 Experimental Setup

4.1 Tasks and Datasets

We evaluate our approach on two simulated multi-agent reasoning tasks: *Find the Spy* and *Say and Guess*.

***Find the Spy* (4-player Setting).** One player is assigned the hidden role of "spy". All Players interact over multiple rounds by providing descriptive clues, expressing attitudes (support and suspicion), and casting votes. The spy aims to mislead others, making the task inherently deceptive. The objective is to identify the spy from the observed trajectory. To assess reasoning under varying evidence volumes, we evaluate models across different observed interaction prefixes T: Spy($T = 1$) and Spy($T = 2$).

***Say and Guess* (5-player Setting).** Players take turns as describers while others guess a target word. The game rules penalize the describer if guessers unanimously succeed or fail, encouraging controlled ambiguity in descriptions. The task is to predict each guesser's outcome. We use the first 5 rounds as a shared contextual behavior baseline, and evaluate on subsequent rounds, i.e., Guess($T = 6$) to Guess($T = 10$), where each setting corresponds to a different describer.

4.2 Evaluation Metrics

We evaluate model performance using Accuracy (Acc.) and weighted F1 score (WA-F1). For the *Say and Guess* task, predictions are treated as multi-label outcomes over guessing agents, and metrics are computed in an element-wise manner. To ensure robustness, we construct multiple data splits with different partition strategies and report the average results. Specifically, for each setting, we discard the highest and lowest scores across splits and compute the mean of the remaining results.

4.3 Baselines

We compare *SHIELD* with baselines representing different reasoning paradigms:

Table 1: Main results on *Find the Spy* tasks. Best results in **bold** and second best in underline.

Method	Spy($T=1$)				Spy($T=2$)			
	Deepseek-v3		Deepseek-v3.2		Deepseek-v3		Deepseek-v3.2	
	Acc./%	WA-F1/%	Acc./%	WA-F1/%	Acc./%	WA-F1/%	Acc./%	WA-F1/%
Struct-only	72.62	72.31	72.62	72.31	58.06	57.80	58.06	57.80
CoT	51.88	51.44	59.38	59.23	43.95	40.82	45.97	45.10
CoT+Struct	64.00	63.71	70.62	70.43	58.06	56.34	55.65	55.64
CoT+Exp	65.38	65.26	66.50	66.63	56.05	55.58	59.27	59.22
SHIELD(CoT)	**74.00**	**73.93**	**75.88**	**75.83**	**74.60**	**74.79**	74.19	74.12
ReAct	47.00	46.06	60.38	60.45	37.10	36.77	41.13	40.57
ReAct+Struct	59.13	59.03	71.12	70.85	57.26	56.27	52.42	52.37
ReAct+Exp	63.38	63.07	69.25	69.15	53.23	53.60	58.47	58.42
SHIELD(ReAct)	72.25	72.08	75.00	74.87	70.97	71.06	**76.61**	**76.99**

Table 2: Main results on *Say and Guess* tasks. Best results in **bold** and second best in underline.

Model	Method	Guess ($T=6$)		Guess ($T=7$)		Guess ($T=8$)		Guess ($T=9$)		Guess ($T=10$)	
		Acc./%	WA-F1/%	Acc./%	WA-F1/%	Acc./%	WA-F1/%	Acc./%	WA-F1/%	Acc./%	WA-F1/%
-	Struct-only	93.14	89.83	84.80	77.83	91.67	87.68	88.24	82.72	84.80	77.83
Deepseek-v3	CoT	88.24	89.18	82.35	80.53	86.76	86.94	84.31	82.46	84.80	81.84
	CoT+Struct	93.63	92.62	84.31	77.59	91.18	88.25	88.24	83.60	85.78	80.82
	CoT+Exp	90.69	89.22	85.29	82.19	90.69	88.59	86.27	81.73	86.76	82.14
	SHIELD(CoT)	**94.61**	**92.94**	86.27	81.80	91.67	87.68	89.22	84.96	87.75	84.96
	ReAct	85.78	87.07	85.78	84.16	83.33	84.74	86.27	84.21	85.78	83.81
	ReAct+Struct	93.14	90.67	87.25	83.64	90.20	88.79	88.24	82.72	**88.73**	**85.75**
	ReAct+Exp	90.20	90.20	86.27	84.19	89.22	**88.91**	**89.71**	**87.97**	86.27	83.38
	SHIELD(ReAct)	93.14	89.83	**87.75**	**84.51**	**92.16**	88.83	**89.71**	85.99	**88.73**	**85.75**
Deepseek-v3.2	CoT	88.24	88.60	81.37	79.02	85.29	84.39	86.27	83.71	85.29	81.13
	CoT+Struct	92.65	90.99	84.80	77.83	91.67	87.68	88.24	82.72	86.27	81.14
	CoT+Exp	90.20	91.35	87.25	85.94	88.73	87.37	89.22	**89.40**	88.24	**87.71**
	SHIELD(CoT)	**95.10**	**93.80**	88.24	85.35	**92.65**	89.89	**91.18**	88.74	**89.71**	87.36
	ReAct	90.69	90.16	83.82	80.67	88.73	88.22	87.25	86.45	87.25	84.96
	ReAct+Struct	93.63	91.66	84.80	77.83	91.67	87.68	88.24	82.72	86.27	81.14
	ReAct+Exp	91.18	90.86	89.22	88.54	91.18	**91.18**	88.73	86.83	88.24	86.12
	SHIELD(ReAct)	94.12	92.56	**91.18**	**89.59**	92.16	88.83	89.22	84.96	89.22	86.57

Reasoning-augmented Methods. We include **CoT** (Wei et al., 2022) and **ReAct** (Yao et al., 2022b) with few-shot prompting, which rely on the internal reasoning capabilities of LLMs.

Experience-enhanced Methods. To isolate the impact of heuristic knowledge, **CoT+Exp** and **ReAct+Exp** provide LLMs with the full set of distilled heuristic insights utilized by *SHIELD*.

Structure-driven Methods. To assess relational modeling, **CoT+Struct** and **ReAct+Struct** inject structural signals into the prompt. We also evaluate **Struct-only**, a GATv2 model operating exclusively on interaction graphs.

These baselines correspond to semantic-only, experience-enhanced, and structure-aware reasoning, enabling a systematic analysis of their individual and combined effects. Notably, the experience- and structure-enhanced variants serve as controlled ablation of *SHIELD*, where our dual-driven reasoner degenerates into a single-guidance variant. All LLM-based models use Deepseek-v3 (DeepSeek-AI et al., 2025a) and Deepseek-v3.2 (DeepSeek-AI et al., 2025b) as backbones. Implementation details are provided in Appendix C.

5 Results

5.1 Overall Performance

Main Results on *Find the Spy*. Table 1 presents the performance on *Find the Spy*. Most baselines show clear degradation from Spy($T=1$) to Spy($T=2$), suggesting that extended interactions introduce additional deceptive noise. While Struct-only achieves competitive results on Spy($T=1$), it is less robust on Spy($T=2$), with accuracy dropping by 14.56%. Augmenting LLMs with either structure or experience consistently outperforms vanilla baselines. For example, CoT+Struct and CoT+Exp outperform CoT by 11.24% and 7.12% accuracy on Spy($T=1$) under Deepseek-v3.2. Overall, *SHIELD* achieves the best performance in nearly all settings. The only exception is Spy($T=1$) with Deepseek-v3, where *SHIELD*(ReAct) is slightly lower than Struct-only

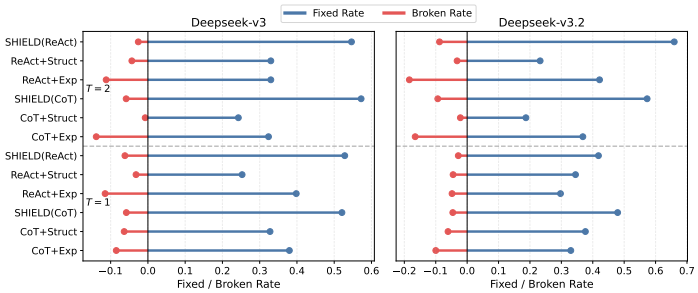

Figure 3: Prediction transition analysis on *Find the Spy*. *Fixed rate* and *broken rate* denote the proportions of incorrect predictions corrected and correct ones flipped after incorporating guidance.

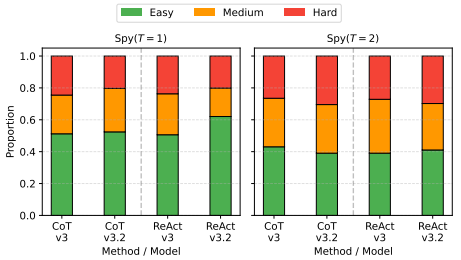

Figure 4: Difficulty distribution of trajectories in *Find the Spy*, with 'v3' and 'v3.2' referring to Deepseek-v3 and Deepseek-v3.2, respectively.

by 0.37% accuracy and 0.21% WA-F1. Notably, on the challenging Spy($T = 2$) setting with Deepseek-v3.2, *SHIELD*(ReAct) improves accuracy from 41.13% to 76.61% (+25.48%), highlighting the advantage of integrating structural and heuristic guidance.

Main Results on *Say and Guess*. Table 2 presents the performance on *Say and Guess*. Compared to *Find the Spy*, this task exhibits smaller performance variations across different settings, indicating more stable reasoning patterns. Augmenting CoT or ReAct with either structural or heuristic guidance consistently ybrings performance gains. For example, CoT+Struct and ReAct+Exp outperform their vanilla counterparts by 5.39% and 4.42% accuracy on Guess($T = 6$) with Deepseek-v3. A rare exception occurs on Guess($T = 10$) under Deepseek-v3.2, where ReAct+Struct slightly underperforms vanilla ReAct, suggesting that excessive reliance on structural signals may mislead the model and distort initially correct semantic intuitions. Overall, *SHIELD* achieves either the best or highly competitive accuracy and WA-F1 in nearly all settings. Particularly under Deepseek-v3.2, *SHIELD*(CoT) maintains dominant performance, demonstrating its superiority.

Key Observations. First, external guidance brings higher relative gains for weaker backbones (Deepseek-v3), while stronger models (Deepseek-v3.2) exhibit higher absolute performance ceilings. Second, although the competitiveness of Struct-only validates that interaction topology carries rich predictive cues, its black-box nature limits interpretability and heightens sensitivity to training quality. Third, structural signals empower LLMs to achieve performance comparable to or even better than specialized graph models. Fourth, distilled heuristics provide consistent gains, confirming the value of reusable reasoning knowledge. Crucially, while standalone structural or heuristic guidance may suffer from over-correction or over-generalization, *SHIELD* harmonizes these components. Its superior performance confirms that structural signals and heuristic experience are complementary rather than redundant, making *SHIELD* a flexible plug-and-play enhancement module for existing LLM reasoning frameworks.

5.2 Component-wise Analysis

Contribution of Structural and Heuristic Signals. While Table 1 and Table 2 demonstrate the superiority of *SHIELD*, these metrics obscure the component-level impact on individual predictions. Therefore, we conduct a prediction transition analysis using two fine-grained metrics: the *fixed rate*, measuring how often external guidance corrects originally incorrect predictions, and the *broken rate*, measuring how often it flips originally correct predictions to incorrect ones.

Using *Find the Spy* as a representative case, as shown in Figure 3, *SHIELD* achieves a better balance between error correction and prediction stability. Although all methods exhibit a double-edged effect, simultaneously fixing and disrupting predictions, the *fixed rate* consistently exceeds the *broken rate*, explaining the overall gains in the main results. We further observe distinct impacts of the two signals. Heuristic experience drives aggressive correction, yielding a higher *fixed rate* but also a higher *broken rate*, indicating a more aggressive correction behavior. For example, CoT+Exp achieves 39.81% *fixed rate* and 11.49% *broken rate* on Spy($T = 1$) with Deepseek-V3. Conversely, structural signals are more conservative, with CoT+Struct achieving a near-zero *broken rate* (0.73%) on Spy($T = 2$). By combining both signals, *SHIELD* maximizes correction while re-

Method	Insight Type	Spy($T=1$)		Spy($T=2$)	
		Acc./%	WA-F1/%	Acc./%	WA-F1/%
CoT	All	**75.88**	**75.83**	**74.19**	**74.12**
	Semantic	71.88	71.69	70.56	70.78
	Attitude	73.12	72.84	72.18	72.42
	Voting	72.62	72.44	69.35	69.45
ReAct	All	**75.00**	**74.87**	**76.61**	**76.99**
	Semantic	69.75	69.32	68.95	68.71
	Attitude	71.00	70.73	70.97	70.45
	Voting	72.00	71.67	73.39	73.48

Table 3: Performance of using different types of insights in *Find the Spy* (Deepseek-v3.2).

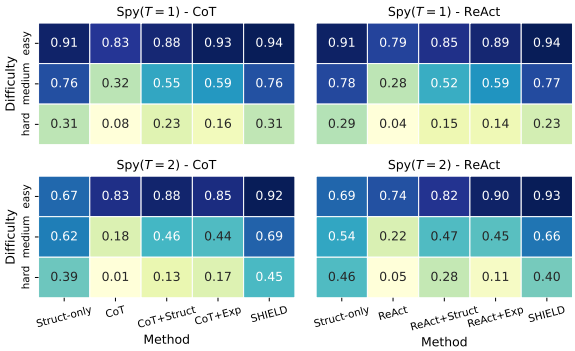

Figure 5: Accuracy of different methods with Deepseek-v3 across different difficulty levels in *Find the Spy*.

straining disruption. For example, on Spy($T = 2$) with Deepseek-V3.2, *SHIELD*(ReAct) peaks at a 65.95% *fixed rate* while restricting the *broken rate* to 8.80%, suppressing the high disruption of ReAct+Exp (18.40%) while far exceeding the correction capacity of ReAct+Struct (23.24%).

Effect of Different Insight Types. We further study the impact of different insight sources. Using *Find the Spy* as a representative case, we decompose the insights into 'semantic', 'attitude', and 'voting' dimensions. Table 3 presents the results of Deepseek-v3.2 when each insight type is utilized in isolation. While all three types of insights consistently outperform the vanilla baseline, none can achieve the peak performance of the full *SHIELD*. This confirms that each dimension captures useful but incomplete reasoning signals. Notably, the integration of all insight types yields the best performance across both CoT and ReAct, indicating that these signals provide complementary cues rather than redundant information.

5.3 Difficulty-wise Analysis

Difficulty Perception. Figure 4 illustrates the difficulty distributions in trajectory collection for *Find the Spy*. While different reasoning paradigms and backbones exhibit slight variations, the overall difficulty perception remains broadly consistent. On Spy($T = 1$), both models classify approximately half of the instances as easy. However, as the interaction proceeds to Spy($T = 2$), a clear difficulty migration occurs: the proportion of easy cases decreases, for example, from over 60% to around 40% for ReAct with Deepseek-v3.2, while medium and hard cases increase accordingly. This indicates that multi-round interactions introduce additional ambiguity and strategic complexity, necessitating more robust reasoning frameworks.

Performance across Difficulty Levels. Figure 5 presents the accuracy across easy, medium, and hard cases using Deepseek-v3 for *Find the Spy*. On easy cases, vanilla CoT and ReAct already achieve decent performance, yet *SHIELD* further improves accuracy, exceeding 92% in all settings. The most pronounced gains emerge in medium and hard cases, where vanilla LLMs suffer severe degradation. For example, in hard Spy($T = 2$) cases, vanilla CoT drops to 1.1% accuracy, whereas *SHIELD*(CoT) recovers it to 45%, a nearly 40-fold relative improvement. Single-guidance variants provide consistent improvement. While Struct-only sometimes performs *SHIELD* on hard cases, it often underperforms on easy and medium cases. Overall, *SHIELD* achieves the most balanced performance, maintaining high accuracy on easy cases while substantially improving performance on medium and hard cases.

6 Conclusion

In this paper, we introduce *SHIELD*, a dual-driven framework for enhancing LLM reasoning in complex multi-agent interaction environments by jointly modeling structural interaction signals and distilled heuristic experience. Experimental results on two multi-agent reasoning tasks show that *SHIELD* consistently improves reasoning performance across different LLM backbones and prompting paradigms. Further analysis shows that structural and heuristic signals exhibit a dual effect on prediction behavior, enabling both error correction and prediction disruption. Their integration in *SHIELD* leads to more stable performance and consistent gains across difficulty levels, particularly in the hardest cases. These results highlight the complementarity of structural signals and heuristic experience for enhancing reasoning in complex multi-agent environments.

Limitations

While *SHIELD* demonstrates strong performance on simulated multi-agent reasoning tasks, it still has several limitations.

First, although simulation enables scalable data construction, it may still introduce distribution shifts or systematic bias compared to real-world human interactions (Ranjan et al., 2024), and cannot fully cover the diversity of open-world scenarios. In particular, our experiments are conducted on two representative but still closed-form game settings. While they successfully mirror multi-round interaction, strategic behavior, and intent deception in real-world interactions, future work is required to validate the generalizability of *SHIELD* to open-domain, messy, and highly implicit real-world social interactions.

Second, our heuristic insights are obtained through abstraction and hierarchical aggregation, which may inevitably lead to information loss when compressing fine-grained reasoning traces. Furthermore, while the distillation process can be optimized via selective sampling to avoid the need for per-instance trajectory processing, this introduces a coverage-efficiency trade-off. Specifically, sparse sampling may fail to capture the full spectrum of heuristic knowledge, whereas exhaustive distillation remains computationally intensive even with abundant data.

Third, the effectiveness of our structural extractor heavily depends on the topological fidelity of the constructed interaction graphs. In some interactive environments, agent interactions may be sparse, implicit, or corrupted by adversarial noise, which potentially undermines the reliability of the extracted structural signals. To mitigate the sensitivity to graph topology in complex real-world scenarios, the simple structural extractor instance currently deployed in our experiments could be upgraded to integrate advanced robust graph learning paradigms (Dong and Kluger, 2023; Ju et al., 2026), which offer a potential path to alleviate topological noise.

Finally, integrating structural and heuristic signals introduces additional computational cost, as experience distillation relies on LLM inference and structural modeling requires graph-model training. Moreover, due to the inherently black-box nature of LLMs (Somvanshi et al., 2026), the internal coordination between bottom-up heuristic intuition and top-down structural guidance remains difficult to explicitly control or interpret. While their complementary effects can be observed from the final outputs and empirical performance, we cannot guarantee an ideal or fully predictable integration mechanism, which may lead to task-dependent variance across different settings.

Ethical Considerations. This work involves simulated multi-agent interaction environments constructed using LLMs. Since authentic interaction data in complex strategic environments is often difficult to obtain due to privacy and observability constraints, all interaction trajectories in our study are generated through controlled simulations rather than collected from real human participants. Therefore, our study does not involve human subjects, personal data, or private information, and does not pose direct risks related to privacy, consent, or human harm. We also acknowledge that the use of LLMs as simulated agents may inherit biases present in pretraining data and could generate unintended or biased outputs during simulation. However, since our focus is on structural and heuristic reasoning, these risks are limited to controlled experimental settings. If future applications involve real-world interaction data, appropriate preprocessing procedures, such as anonymization and removal of sensitive information, should be conducted before data usage. We encourage responsible use of our reasoning framework and caution against deploying it in critical environments without further validation.

References

Janice Ahn, Rishu Verma, Renze Lou, Di Liu, Rui Zhang, and Wenpeng Yin. 2024. Large language models for mathematical reasoning: Progresses and challenges. In *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics: Student Research Workshop*, pages 225–237, St. Julian's, Malta. Association for Computational Linguistics.

Lisa Alazraki, Maximilian Mozes, Jon Ander Campos, Tan Yi-Chern, Marek Rei, and Max Bartolo. 2025. No need for explanations: LLMs can implicitly learn from mistakes in-context. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 33191–33215, Suzhou, China. Association for Computational Linguistics.

Chenyang An, Zhibo Chen, Qihao Ye, Emily First, Letian Peng, Jiayun Zhang, Zihan Wang, Sorin Lerner, and Jingbo Shang. 2024a. Learn from failure: Fine-tuning LLMs with trial-and-error data for intuitionistic propositional logic proving. In *Proceedings*

of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 776–790, Bangkok, Thailand. Association for Computational Linguistics.

Shengnan An, Zexiong Ma, Siqi Cai, Zeqi Lin, Nanning Zheng, Jian-Guang Lou, and Weizhu Chen. 2024b. Can LLMs learn from mistakes? an empirical study on reasoning tasks. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 833–854, Miami, Florida, USA. Association for Computational Linguistics.

Kinjal Basu, Ibrahim Abdelaziz, Subhajit Chaudhury, Soham Dan, Maxwell Crouse, Asim Munawar, Vernon Austel, Sadhana Kumaravel, Vinod Muthusamy, Pavan Kapanipathi, and Luis Lastras. 2024. API-BLEND: A comprehensive corpora for training and benchmarking API LLMs. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 12859–12870, Bangkok, Thailand. Association for Computational Linguistics.

Shaked Brody, Uri Alon, and Eran Yahav. 2022. How attentive are graph attention networks? In *International Conference on Learning Representations*.

Kai Chen, Chunwei Wang, Kuo Yang, Jianhua Han, Lanqing HONG, Fei Mi, Hang Xu, Zhengying Liu, Wenyong Huang, Zhenguo Li, Dit-Yan Yeung, and Lifeng Shang. 2024. Gaining wisdom from setbacks: Aligning large language models via mistake analysis. In *The Twelfth International Conference on Learning Representations*.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. Training verifiers to solve math word problems. *Preprint*, arXiv:2110.14168.

DeepSeek-AI, Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, and et al. 2025a. Deepseek-v3 technical report. *Preprint*, arXiv:2412.19437.

DeepSeek-AI, Aixin Liu, Aoxue Mei, Bangcai Lin, Bing Xue, Bingxuan Wang, Bingzheng Xu, Bochao Wu, Bowei Zhang, Chaofan Lin, Chen Dong, and et al. 2025b. Deepseek-v3.2: Pushing the frontier of open large language models. *Preprint*, arXiv:2512.02556.

Mingze Dong and Yuval Kluger. 2023. Towards understanding and reducing graph structural noise for GNNs. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 8202–8226. PMLR.

Qingxiu Dong, Lei Li, Damai Dai, Ce Zheng, Jingyuan Ma, Rui Li, Heming Xia, Jingjing Xu, Zhiyong Wu, Baobao Chang, Xu Sun, Lei Li, and Zhifang Sui. 2024. A survey on in-context learning. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 1107–1128, Miami, Florida, USA. Association for Computational Linguistics.

Shmuel Ellis and Inbar Davidi. 2005. After-event reviews: drawing lessons from successful and failed experience. *Journal of applied psychology*, 90(5):857.

Wenqi Fan, Yujuan Ding, Liangbo Ning, Shijie Wang, Hengyun Li, Dawei Yin, Tat-Seng Chua, and Qing Li. 2024. A survey on rag meeting llms: Towards retrieval-augmented large language models. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, KDD '24, page 6491–6501, New York, NY, USA. Association for Computing Machinery.

Jinglong Gao, Xiao Ding, Yiming Cui, Jianbai Zhao, Hepeng Wang, Ting Liu, and Bing Qin. 2024. Self-evolving GPT: A lifelong autonomous experiential learner. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6385–6432, Bangkok, Thailand. Association for Computational Linguistics.

Jinglong Gao, Xiao Ding, Lingxiao Zou, Bibo Cai, Bing Qin, and Ting Liu. 2025. ExpeTrans: LLMs are experiential transfer learners. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 10577–10616, Vienna, Austria. Association for Computational Linguistics.

Zhibin Gou, Zhihong Shao, Yeyun Gong, yelong shen, Yujiu Yang, Minlie Huang, Nan Duan, and Weizhu Chen. 2024. ToRA: A tool-integrated reasoning agent for mathematical problem solving. In *The Twelfth International Conference on Learning Representations*.

Xinyi Hou, Yanjie Zhao, Shenao Wang, and Haoyu Wang. 2026. Model context protocol (mcp): Landscape, security threats, and future research directions. *ACM Transactions on Software Engineering and Methodology*.

Yizheng Huang and Jimmy Xiangji Huang. 2026. A survey on retrieval-augmented text generation for large language models. *ACM Computing Surveys*, 58(12).

Juyong Jiang, Fan Wang, Jiasi Shen, Sungju Kim, and Sunghun Kim. 2026. A survey on large language models for code generation. *ACM Transactions on Software Engineering and Methodology*, 35(2).

Wei Ju, Siyu Yi, Yifan Wang, Zhiping Xiao, Zhengyang Mao, Hourun Li, Yiyang Gu, Yifang Qin, Nan Yin, Senzhang Wang, Xinwang Liu, Philip S. Yu, and Ming Zhang. 2026. A survey of graph neural networks in real world: Imbalance, noise, privacy and ood challenges. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 48(3):3036–3055.

Takeshi Kojima, Shixiang (Shane) Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. Large language models are zero-shot reasoners. In *Advances in Neural Information Processing Systems*, volume 35, pages 22199–22213. Curran Associates, Inc.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. In *Advances in Neural Information Processing Systems*, volume 33, pages 9459–9474. Curran Associates, Inc.

Xiaoyu Liu, Paiheng Xu, Junda Wu, Jiaxin Yuan, Yifan Yang, Yuhang Zhou, Fuxiao Liu, Tianrui Guan, Haoliang Wang, Tong Yu, Julian McAuley, Wei Ai, and Furong Huang. 2025. Large language models and causal inference in collaboration: A comprehensive survey. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pages 7683–7699, Albuquerque, New Mexico. Association for Computational Linguistics.

Qing Lyu, Shreya Havaldar, Adam Stein, Li Zhang, Delip Rao, Eric Wong, Marianna Apidianaki, and Chris Callison-Burch. 2023. Faithful chain-of-thought reasoning. In *Proceedings of the 13th International Joint Conference on Natural Language Processing and the 3rd Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 305–329, Nusa Dua, Bali. Association for Computational Linguistics.

Jing Ma. 2025. Causal inference with large language model: A survey. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pages 5901–5913, Albuquerque, New Mexico. Association for Computational Linguistics.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. 2023. Self-refine: Iterative refinement with self-feedback. In *Advances in Neural Information Processing Systems*, volume 36, pages 46534–46594. Curran Associates, Inc.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul F Christiano, Jan Leike, and Ryan Lowe. 2022. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems*, volume 35, pages 27730–27744. Curran Associates, Inc.

Aske Plaat, Annie Wong, Suzan Verberne, Joost Broekens, Niki Van Stein, and Thomas Bäck. 2025. Multi-step reasoning with large language models, a survey. *ACM Computing Surveys*, 58(6).

Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, Sihan Zhao, Lauren Hong, Runchu Tian, Ruobing Xie, Jie Zhou, Mark Gerstein, dahai li, Zhiyuan Liu, and Maosong Sun. 2024. ToolLLM: Facilitating large language models to master 16000+ real-world APIs. In *The Twelfth International Conference on Learning Representations*.

Changle Qu, Sunhao Dai, Xiaochi Wei, Hengyi Cai, Shuaiqiang Wang, Dawei Yin, Jun Xu, and Ji-Rong Wen. 2025a. From exploration to mastery: Enabling LLMs to master tools via self-driven interactions. In *The Thirteenth International Conference on Learning Representations*.

Changle Qu, Sunhao Dai, Xiaochi Wei, Hengyi Cai, Shuaiqiang Wang, Dawei Yin, Jun Xu, and Ji-Rong Wen. 2025b. Tool learning with large language models: A survey. *Frontiers of Computer Science*, 19(8):198343.

Rajesh Ranjan, Shailja Gupta, and Surya Narayan Singh. 2024. A comprehensive survey of bias in llms: Current landscape and future directions. *Preprint*, arXiv:2409.16430.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. 2023. Reflexion: language agents with verbal reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 36, pages 8634–8652. Curran Associates, Inc.

Shriyank Somvanshi, Md Monzurul Islam, Amir Rafe, Anannya Ghosh Tusti, Arka Chakraborty, Anika Baitullah, Tausif Islam Chowdhury, Nawaf Alnawmasi, Anandi Dutta, and Subasish Das. 2026. Bridging the black box: A survey on mechanistic interpretability in ai. *ACM Computing Surveys*, 58(8).

Yifan Song, Da Yin, Xiang Yue, Jie Huang, Sujian Li, and Bill Yuchen Lin. 2024. Trial and error: Exploration-based trajectory optimization of LLM agents. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7584–7600, Bangkok, Thailand. Association for Computational Linguistics.

Yongqi Tong, Dawei Li, Sizhe Wang, Yujia Wang, Fei Teng, and Jingbo Shang. 2024. Can LLMs learn from previous mistakes? investigating LLMs' errors to boost for reasoning. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3065–3080, Bangkok, Thailand. Association for Computational Linguistics.

Chong Wang, Jian Zhang, Yebo Feng, Tianlin Li, Weisong Sun, Yang Liu, and Xin Peng. 2025a. Teaching code llms to use autocompletion tools in repository-level code generation. *ACM Transactions on Software Engineering and Methodology*, 34(7).

Chuwen Wang, Shirong Zeng, and Cheng Wang. 2024. Large language models need consultants for reasoning: Becoming an expert in a complex human system through behavior simulation. *Preprint*, arXiv:2403.18230.

Lei Wang, Wanyu Xu, Yihuai Lan, Zhiqiang Hu, Yunshi Lan, Roy Ka-Wei Lee, and Ee-Peng Lim. 2023a. Plan-and-solve prompting: Improving zero-shot chain-of-thought reasoning by large language models. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2609–2634, Toronto, Canada. Association for Computational Linguistics.

Luping Wang, Sheng Chen, Linnan Jiang, Shu Pan, Runze Cai, Sen Yang, and Fei Yang. 2025b. Parameter-efficient fine-tuning in large language models: a survey of methodologies. *Artificial Intelligence Review*, 58(8):227.

Peng-Yuan Wang, Tian-Shuo Liu, Chenyang Wang, Ziniu Li, Yidi Wang, Shu Yan, Chengxing Jia, Xu-Hui Liu, Xinwei Chen, Jiacheng Xu, and Yang Yu. 2026. A survey on large language models for mathematical reasoning. *ACM Computing Surveys*, 58(8).

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2023b. Self-consistency improves chain of thought reasoning in language models. In *The Eleventh International Conference on Learning Representations*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, brian ichter, Fei Xia, Ed Chi, Quoc V Le, and Denny Zhou. 2022. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*, volume 35, pages 24824–24837. Curran Associates, Inc.

Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S. Yu. 2021. A comprehensive survey on graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 32(1):4–24.

Yuchen Yan, Jin Jiang, Yang Liu, Yixin Cao, Xin Xu, Mengdi Zhang, Xunliang Cai, and Jian Shao. 2025. Sˆ 3cmath: Spontaneous step-level self-correction makes large language models better mathematical reasoners. *Proceedings of the AAAI Conference on Artificial Intelligence*, 39(24):25588–25596.

Zeyuan Yang, Peng Li, and Yang Liu. 2023. Failures pave the way: Enhancing large language models through tuning-free rule accumulation. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 1751–1777, Singapore. Association for Computational Linguistics.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. 2018. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2369–2380, Brussels, Belgium. Association for Computational Linguistics.

Shunyu Yao, Howard Chen, John Yang, and Karthik Narasimhan. 2022a. Webshop: Towards scalable real-world web interaction with grounded language agents. In *Advances in Neural Information Processing Systems*, volume 35, pages 20744–20757. Curran Associates, Inc.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. 2023. Tree of thoughts: Deliberate problem solving with large language models. In *Advances in Neural Information Processing Systems*, volume 36, pages 11809–11822. Curran Associates, Inc.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Izhak Shafran, Karthik R Narasimhan, and Yuan Cao. 2022b. React: Synergizing reasoning and acting in language models. In *NeurIPS 2022 Foundation Models for Decision Making Workshop*.

Quanjun Zhang, Chunrong Fang, Yang Xie, Yaxin Zhang, Shengcheng Yu, Weisong Sun, Yun Yang, and Zhenyu Chen. 2026a. A survey on large language models for software engineering. *Science China Information Sciences*, 69(4):141102.

Shengyu Zhang, Linfeng Dong, Xiaoya Li, Sen Zhang, Xiaofei Sun, Shuhe Wang, Jiwei Li, Runyi Hu, Tianwei Zhang, Guoyin Wang, and Fei Wu. 2026b. Instruction tuning for large language models: A survey. *ACM Computing Surveys*, 58(7).

Tianjun Zhang, Aman Madaan, Luyu Gao, Steven Zheng, Swaroop Mishra, Yiming Yang, Niket Tandon, and Uri Alon. 2024. In-context principle learning from mistakes. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24. JMLR.org.

Zhuosheng Zhang, Aston Zhang, Mu Li, and Alex Smola. 2023. Automatic chain of thought prompting in large language models. In *The Eleventh International Conference on Learning Representations*.

Andrew Zhao, Daniel Huang, Quentin Xu, Matthieu Lin, Yong-Jin Liu, and Gao Huang. 2024. Expel: Llm agents are experiential learners. *Proceedings of the AAAI Conference on Artificial Intelligence*, 38(17):19632–19642.

A Simulation Environments and Data Representation

A.1 Task Definition

In this study, we utilize *Find the Spy* and *Say and Guess* as our evaluation environments, adopting the simulation protocol established by Wang et al. (2024). These environments serve as complex

testbeds for evaluating LLMs' ability to navigate asymmetric information and strategic discourse in multi-agent settings.

Find the Spy is a competitive game involving four players, where three players belong to the folk group and receive the same description word, while one player is assigned the role of the spy and receives a similar but distinct word. Players do not know either their own identities or those of others. The game proceeds through multiple rounds in which players provide descriptions, express attitudes (support or suspicion) toward others, and participate in a voting process where the player with the most votes is eliminated. The game terminates when the spy is eliminated or when only one folk player and the spy remain. Under the four-player setting, the voting phase can last for at most two rounds. The game terminates when the spy is eliminated or when only one folk player and the spy remain.

In our setting, the LLM acts as an external observer and must identify the spy solely from partial interaction trajectories containing linguistic descriptions, interpersonal attitudes, and voting records. Therefore, the model needs to reason under epistemic uncertainty while recognizing potentially deceptive or strategically ambiguous behaviors embedded in the interactions. To ensure consistency, each simulated session follows a fixed interaction protocol consisting of sequential stages of description, attitude expression, and voting. The resulting interaction trajectories, composed of player utterances and behavioral records, form the raw data used for both heterogeneous interaction graph construction and heuristic experience distillation.

Say and Guess is a more cooperative multi-agent game in which players take turns acting as the describer while the remaining participants serve as guessers. In each round, the describer provides a description of a designated target word, and the guessers attempt to infer the correct answer based on the description. A guesser receives one point for each correct guess. The describer receives a score equal to the number of successful guessers unless all guessers answer correctly, in which case the describer receives no points. This mechanism discourages overly explicit descriptions and encourages strategically balanced communication.

Unlike *Find the Spy*, the game contains no elimination process and can therefore continue indefinitely. In our experiments, the game terminates after each player has acted as the describer twice. The task of the LLM is to predict the guessing outcomes of all guessers based on the historical interaction trajectory and the description provided in the current prediction round. To ensure consistency, each simulated session follows a fixed interaction protocol consisting of sequential stages of description, guessing, and score settlement.

A.2 Edge Types in Interaction Graphs

In *Find the Spy*, the interaction graph contains three types of edges: *support*, *suspicion*, and *vote*. The first two correspond to attitude-based interactions, while the last represents explicit voting behavior. Since interactions occurring at different stages may carry different strategic implications, edges of the same interaction category are further distinguished by their temporal stage. Therefore, an interaction event t_i is associated with an edge attribute $\tau = (a_i, r_i)$, where a_i denotes the interaction action type and r_i denotes the interaction round, as described in Section 3.3. In *Say and Guess*, the interaction graph contains two types of edges. We use self-loop edges to identify the describer in a specific round. Additionally, if a guesser successfully infers the target word, a directed edge is constructed from the guesser node to the describer node to represent a successful guessing interaction.

A.3 Insight Taxonomy

In *Find the Spy*, the interaction signals are derived from three primary sources: descriptions, attitudes, and votes. Accordingly, we define three corresponding insight categories: *semantic*, which captures how semantic relationships among description words provide cues for distinguishing the spy from the folk players, *attitude*, which models how inter-player relations (support and suspicion) reflect latent alliances and adversarial behavior, and *voting*, which reflects how collective voting patterns reveal inconsistencies in behavioral coordination and potential deceptive strategies.

In *Say and Guess*, we define a different set of insight categories tailored to cooperative guessing behavior. Specifically, *description* captures the intrinsic properties of utterances, including information content, abstraction level, and ambiguity, and their influence on downstream understanding, *player_model* captures how historical behavioral tendencies of players affect their guessing behavior, and *outcome* summarizes the relationship between description patterns and resulting guessing outcomes.

To ensure interpretability and avoid information leakage, each distilled insight is constrained to involve only a single information source. Insights are not allowed to mix multiple categories, and they are prohibited from containing explicit player identities, target words, or category-specific labels.

B Additional Method Details

B.1 Three heterogeneous Signals in Interaction Trajectories

Effective reasoning in complex interaction environments requires integrating heterogeneous evidence from multiple sources:

1. **Semantic Clues**: The interaction trajectory contains surface-level signals, such as textual utterances and discrete actions. These cues can be partially understood through the intrinsic language understanding and commonsense reasoning abilities of LLMs.

2. **Structural Signals**: Interactions among agents lead to latent dependency patterns, such as alliances, conflicts, consistency relations, and influence structures. Exploiting such relational evidence requires modeling the global interaction structure across the trajectory.

3. **Heuristic Experience**: Given a historical interaction trajectory, its corresponding reasoning trajectory can be derived by prompting an LLM to solve the task. When a sufficient number of reasoning trajectories are available, reusable high-level insights can be further distilled from these reasoning processes. Such accumulated experience can provide valuable guidance for future inference.

B.2 Structural Extractor Architecture

The structural extractor of *SHIELD* is modular and can be instantiated with different graph encoders. In our experiments, we adopt a relational GATv2-based architecture.

Given the interaction graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ constructed in Section 3.3, each node $v_i \in \mathcal{V}$ is initialized with an identity embedding $\mathbf{h}_i^{(0)} = \mathbf{x}_i$. We perform type-conditioned message passing over the interaction graph. For each edge type τ, the message aggregation at layer l is defined as:

$$\mathbf{h}_{i,\tau}^{(l)} = \sigma\left(\sum_{j \in \mathcal{N}_{i,\tau}} \alpha_{ij,\tau}^{(l)} \cdot \mathbf{W}_\tau^{(l)} \mathbf{h}_j^{(l)}\right), \quad (1)$$

Task	Graphs	Nodes	Edges
Spy($T = 1$)	498	1992	6284
Spy($T = 2$)	151	604	3074
Guess($T = 6$)	255	1275	5449
Guess($T = 7$)	255	1275	6799
Guess($T = 8$)	255	1275	7942
Guess($T = 9$)	255	1275	9129
Guess($T = 10$)	255	1275	10316

Table 4: Statistics of constructed datasets for different tasks.

$$e_{ij,\tau}^{(l)} = a_\tau^{(l)\top} \mathrm{LeakyReLU}\left(\mathbf{W}_\tau^{(l)}[\mathbf{h}_i^{(l)} \| \mathbf{h}_j^{(l)}]\right), \quad (2)$$

$$\alpha_{ij,\tau}^{(l)} = \mathrm{softmax}_{j \in \mathcal{N}_{i,\tau}}(e_{ij,\tau}^{(l)}), \quad (3)$$

where $\mathbf{W}_\tau^{(l)}$ is a learnable type-specific projection matrix that maps node features into a relation-aware latent space for edge type τ, and $a_\tau^{(l)}$ is a learnable attention vector that measures the importance of interactions under type τ. The representations from different interaction types are then aggregated as:

$$\mathbf{h}_i^{(l+1)} = \sum_{\tau \in \mathcal{T}} \mathbf{h}_{i,\tau}^{(l)}. \quad (4)$$

After L layers, we obtain $\mathbf{H}^{(L)} = \{\mathbf{h}_i^{(L)}\}_{i=1}^{|\mathcal{V}|}$. Task-specific predictions are derived from the final node representations $\mathbf{H}^{(L)}$ as follows.

For *Find the Spy*, we apply mean pooling over node representations, followed by a softmax classifier trained with standard multi-class cross-entropy loss. For *Say and Guess*, we perform node-level binary prediction using a sigmoid output layer with binary cross-entropy loss, where each node prediction corresponds to an individual candidate's outcome.

C Implementation Details

C.1 Dataset

The constructed interaction datasets exhibit diverse characteristics across tasks and observation lengths, as summarized in Table 4. For the *Find the Spy* task, the interaction graphs contain 498 and 151 graph instances under Spy($T = 1$) and Spy($T = 2$), respectively, with corresponding edge counts of 6,284 and 3,074. The substantially smaller number of graph instances on Spy($T = 2$) arises because

Method	Spy($T=1$)				Spy($T=2$)			
	sem.	att.	vot.	total	sem.	att.	vot.	total
CoT (v3)	2	4	1	7	6	1	2	9
CoT (v3.2)	3	4	2	9	2	3	2	7
ReAct (v3)	4	4	2	10	9	3	1	13
ReAct (v3.2)	3	3	2	8	3	2	2	7

Table 5: Distribution of extracted insights across different categories on *Find the Spy*. Here, 'sem.', 'att.', and 'vot.' denote *semantic*, *attitude*, and *voting*, respectively.

Method	Metric	Guess($T=6$)	Guess($T=7$)	Guess($T=8$)	Guess($T=9$)	Guess($T=10$)
CoT (v3)	*description*	4	4	4	4	9
	player_model	1	1	1	2	1
	outcome	2	1	3	0	2
	total	7	6	8	6	12
CoT (v3.2)	*description*	3	2	2	3	1
	player_model	1	1	2	1	2
	outcome	1	0	1	1	2
	total	5	3	5	5	5
ReAct (v3)	*description*	5	9	10	5	6
	player_model	1	2	1	2	1
	outcome	1	3	0	0	0
	total	7	14	11	7	7
ReAct (v3.2)	*description*	3	3	2	3	2
	player_model	1	1	1	1	2
	outcome	1	1	1	0	1
	total	5	5	4	4	5

Table 6: Distribution of extracted insights across different categories on *Say and Guess*.

the game terminates early once the spy is successfully eliminated in the first round. In contrast, the *Say and Guess* task maintains a fixed number of graph instances and nodes across different stages, while the number of edges increases steadily from 5,449 to 10,316 as the interaction progresses from Guess($T = 6$) to Guess($T = 10$). This trend reflects the accumulation of increasingly dense interaction dependencies and more complex relational structures over longer reasoning horizons.

C.2 Hyperparameter Settings

Structural Extractor Training. For each dataset, we construct S independent training/test splits with an 8:2 ratio, and train a separate structural extractor on each split. To improve robustness and reduce variance across partitions, we further apply F-fold cross-validation within each split. Early stopping is performed with a patience of K epochs based on validation loss. The structural extractor consists of a 2-layer GATv2 network, where H_l and O_l denote the number of attention heads and output channels in layer l, respectively. Optimization uses learning rate α, weight decay λ, and dropout rate p.

We summarize the hyperparameters using the tuple $(S, F, p, K, H_1, O_1, H_2, O_2, \alpha, \lambda,)$. For *Find the Spy*, we use $(10, 4, 0.1, 30, 8, 32, 8, 16, 0.0005, 0.00001)$ when $T = 1$ and $(10, 4, 0.1, 50, 2, 16, 2, 8, 0.001, 0.0005)$ when $T = 2$. For *Say and Guess*, all models use a same configuration $(1, 4, 0.2, 30, 4, 32, 4, 16, 0.0001, 0.0005)$.

Distilled Insights. The extracted insights are designed to be global and shared across different data splits, ensuring consistency of the distilled reasoning patterns. Different splits are used when constructing few-shot demonstrations during inference, while the distilled insights themselves remain invariant across splits. Due to dataset scale considerations, we apply different sampling strategies across settings. For Spy($T = 1$) and Guess($T = 6, \cdots, 10$), the number of simple cases is relatively large. We therefore uniformly sample 60 instances from the simple subset for insight distillation, while fully utilizing all medium and hard cases without additional sampling. In contrast, for Spy($T = 2$), the dataset size is moderate, and thus no sampling is applied.

The distribution of the final distilled insights is summarized in Table 5 and Table 6. Representative examples of the distilled insights are illustrated in Figure 6 and Figure 7. Overall, the number of extracted insights remains relatively compact, while the insight distributions under Deepseek-v3.2 appear noticeably more stable than those under Deepseek-v3. For the Spy($T = 1$) task, *semantic* and *attitude* insights constitute the dominant categories. In contrast, under Guess($T = 6, \cdots, 10$), *description* insights account for the overwhelming majority of extracted experiences. In several settings with Deepseek-v3 under *Say and Guess*, the number of *outcome* insights becomes extremely limited or even zero. This phenomenon mainly arises from the final cross-category aggregation stage, where all insights from difficult categories are jointly aggregated by the LLMs. Since *description* and *outcome* insights often exhibit strong semantic overlap and are both grounded in descriptive semantic patterns, the model tends to classify the aggregated results into the broader *description* category.

Inference Setting. During interaction trajectory simulation described in Section 3.2, we use Deepseek-v3 as the backbone model for all agents, with temperature set to 0.9 and top-p set to 0.85.

For both inference and heuristic experience distillation, we employ models from both Deepseek-v3 and Deepseek-v3.2 families to ensure robustness across different model capacities. The decoding hyperparameters remain fixed at temperature = 0.9 and top_p = 0.85 across these stages. During inference, we additionally adopt a few-shot prompting strategy by randomly sampling two simple reasoning cases from the training set as demonstrations.

(Insight) Type: Semantic
Pattern: The descriptive words of the majority of players converge on a specific entity, category, or concept, whereas the description of an individual player points toward a closely related but distinct entity.
Inference: High priority should be assigned to suspecting the individual player whose description deviates toward the related yet distinct concept.
...
(Insight) Type: Attitude
Pattern: A prominent faction consisting of the majority or mutually supportive players emerges to collectively suspect or target a single player or a small minority, establishing a clear majority-versus-minority isolation or confrontation.
Inference: Strategic focus should be directed toward investigating the isolated player who faces collective opposition from the majority faction.
...
(Insight) Type: Voting
Pattern: During the voting phase, the ballot behaviors of the majority of players are highly synchronized (e.g., heavily concentrated on a single target), forming a dominant voting bloc, while a minority of players distinctly deviate from this consensus.
Inference: High suspicion should be cast upon players whose voting behaviors explicitly diverge from the majority voting consensus.
...

Figure 6: Examples of different types of insights on *Find the Spy*.

All distilled generalized insights are incorporated into the prompts.

Language Setting. All prompts and interaction trajectories used in our framework are originally constructed in Chinese, motivated by the strong Chinese-language reasoning and dialogue capabilities of the Deepseek model family. For presentation consistency, all examples and cases discussed in this paper are translated into English from their original Chinese versions.

D Additional Experimental Results

D.1 *Fixed rate* and *Broken Rate*

Taking the original predictions of CoT or ReAct as the reference, we examine how the introduction of structural signals, heuristic experience, or the full *SHIELD* framework changes the prediction outcome for each instance.

Specifically, each prediction transition is categorized into four cases: **CC** (correct → correct), **CW** (correct → incorrect), **WC** (incorrect → correct), and **WW** (incorrect → incorrect). Based on these transitions, we define two metrics: the *fixed rate*, measuring the proportion of originally incorrect predictions that are corrected after introducing external guidance,

$$fixed\,rate = \frac{\mathrm{WC}}{\mathrm{WC} + \mathrm{WW}},$$

and the *broken rate*, measuring the proportion of originally correct predictions that are corrupted,

$$broken\,rate = \frac{\mathrm{CW}}{\mathrm{CC} + \mathrm{CW}}.$$

(Insight) Type: Description
Pattern: Multiple highly plausible and reasonable candidate answers exist under the category targeted by the description, or the core features themselves are inherently ambiguous, triggering distinct yet equally logical associative paths.
Inference: You should predict a divergence in collective guessing, where answers are likely to be scattered rather than converging on a unanimous consensus.
...
(Insight) Type: Player_model
Pattern: The describer exhibits a historical record of systemic bias (e.g., their prior descriptions consistently led the group into collective errors), demonstrating that their descriptive style, cognitive bias, or communication modality maintains a stable deviation from the group consensus.
Inference: When predicting subsequent behaviors of this specific describer, the confidence weight assigned to the surface clarity of their descriptions should be discounted, prioritizing the likelihood that their descriptions will cause some or all guessers to deviate from common or pan-expected answers.
...
(Insight) Type: Outcome
Pattern: A player has demonstrated systemic misjudgments within specific contextual scenarios (e.g., festival associations, unique cultural backgrounds, or highly abstract concepts).
Inference:You should strictly isolate the scene-specificity of historical failures to avoid over-generalizing specific contextual biases to all descriptions; instead, evaluations should be performed independently based on the concrete type and cue structure of the current description.
...

Figure 7: Examples of different types of insights on *Say and Guess*.

D.2 Prediction Transition Analysis on *Say and Guess*

To complement the findings from *Find the Spy* in Section 5.2, we extend our prediction transition analysis to the *Say and Guess* task. Table 7 illustrates the *fixed rate* and *broken rate* across various settings for Deepseek-v3 and Deepseek-V3.2 backbones. Consistent with our findings in the main text, heuristic experience exhibits aggressive correction behavior, while structural signals maintain high stability. Notably, in several settings, experience-driven variants may achieve slightly higher *fixed rates* than *SHIELD*. For example, on Guess($T = 7$) with Deepseek-v3, ReAct+Exp attains a fixed rate of 48.3%, exceeding the 44.8% achieved by *SHIELD*(ReAct). However, this does not necessarily indicate superior overall repair quality. The *fixed rate* only measures the proportion of originally incorrect predictions that are successfully corrected, and its denominator differs across settings and methods. Therefore, a higher *fixed rate* mainly reflects more aggressive modifications to previously incorrect predictions. In contrast, such

Table 7: Prediction transition analysis on *Say and Guess*. Each cell reports *fixed rate* ↑ / *broken rate* ↓ (%).

Method	DeepSeek-V3					DeepSeek-V3.2				
	Guess($T=6$)	Guess($T=7$)	Guess($T=8$)	Guess($T=9$)	Guess($T=10$)	Guess($T=6$)	Guess($T=7$)	Guess($T=8$)	Guess($T=9$)	Guess($T=10$)
CoT+Exp	54.2 / 4.4	38.9 / 4.8	37.0 / 1.1	28.1 / 2.9	22.6 / 1.7	41.7 / 3.3	44.7 / 3.0	36.7 / 2.3	60.7 / 6.2	50.0 / 5.2
CoT+Struct	54.2 / 1.1	33.3 / 4.8	48.1 / 2.3	31.2 / 1.2	16.1 / 1.7	41.7 / 0.6	31.6 / 3.0	43.3 / 0.0	25.0 / 1.7	10.0 / 0.6
SHIELD(CoT)	66.7 / 1.7	36.1 / 3.0	51.9 / 2.3	37.5 / 1.2	38.7 / 3.5	66.7 / 1.1	47.4 / 2.4	50.0 / 0.0	46.4 / 1.7	30.0 / 0.0
ReAct+Exp	44.8 / 2.3	48.3 / 7.4	47.1 / 2.4	42.9 / 2.8	41.4 / 6.3	47.4 / 4.3	60.6 / 5.3	43.5 / 2.8	46.2 / 5.1	42.3 / 5.1
ReAct+Struct	58.6 / 1.1	24.1 / 2.3	44.1 / 0.6	28.6 / 2.3	24.1 / 0.6	36.8 / 0.5	21.2 / 2.9	43.5 / 2.2	38.5 / 4.5	15.4 / 3.4
SHIELD(ReAct)	62.1 / 1.7	44.8 / 5.1	61.8 / 1.8	35.7 / 1.7	44.8 / 4.0	47.4 / 1.1	51.5 / 1.2	43.5 / 1.7	46.2 / 4.5	46.2 / 4.5

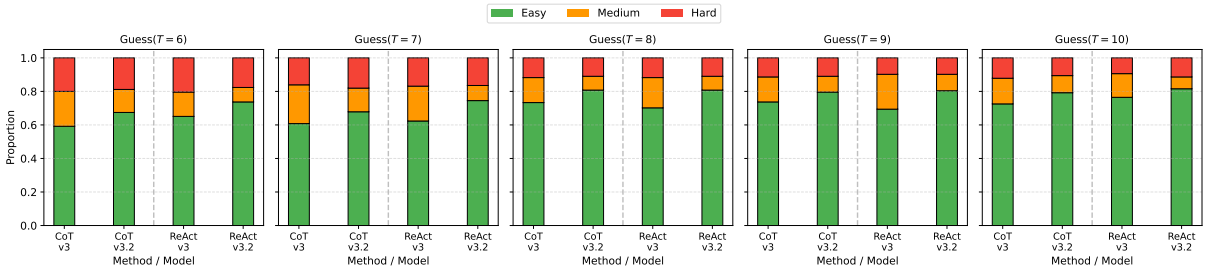

Figure 8: Difficulty distribution of trajectories in *Say and Guess*, with 'v3' and 'v3.2' referring to Deepseek-v3 and Deepseek-v3.2, respectively.

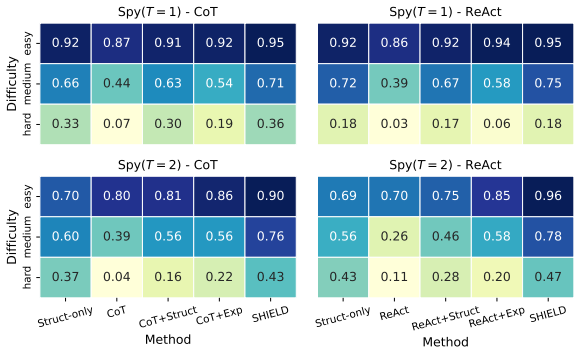

Figure 9: Accuracy of different methods with Deepseek-v3.2 across different difficulty levels in *Find the Spy*.

aggressive correction behavior is often accompanied by substantially larger *broken rate*. Overall, *SHIELD* effectively leverages both signals to maximize the *fixed rate* while keeping the *broken rate* within a stable range.

D.3 Difficulty-aware Analysis

Difficulty Distribution on *Say and Guess*. Figure 8 illustrates the difficulty distributions in trajectory collection for *Say and Guess*. The distribution exhibits a pattern consistent with *Find the Spy*, while shifting toward easier trajectories, indicating that *Say and Guess* presents relatively lower reasoning complexity.

Performance Across Difficulty Levels. Figure 9 presents the accuracy across easy, medium, and hard cases using DeepSeek-v3.2 on *Find the Spy*. Compared with DeepSeek-v3, DeepSeek-v3.2 achieves consistently better performance across most settings, obtaining the highest accuracy in nearly all cases. It maintains strong performance on easy cases while substantially improving accuracy on medium and hard cases. In particular, it achieves close to 40% accuracy on hard cases in most settings, with only a slight degradation on Spy($T = 1$), indicating robust reasoning ability under challenging conditions.

Figure 10 shows the accuracy across easy, medium, and hard cases using DeepSeek-v3.2 on *Say and Guess*. Overall, the performance trends are largely consistent with those observed in *Find the Spy*. However, compared with *Find the Spy*, *Say and Guess* contains substantially fewer samples and exhibits a much larger proportion of easy cases due to its lower overall task difficulty. As a result, the medium and hard subsets in certain settings are relatively small, which may amplify apparent performance fluctuations. For example, under Guess($T = 9$), vanilla ReAct outperforms *SHIELD*(ReAct) on medium cases by more than 30%. Nevertheless, this subset contains only 26 samples, compared with over 200 easy cases, indicating that such large numerical differences should be interpreted cautiously and have a limited impact on the overall conclusions.

Consensus Difficulty Evaluation. To reduce individual model bias and establish a more reliable difficulty benchmark, we conduct a consensus difficulty evaluation. Consensus easy, medium, and hard subsets are defined as the intersection

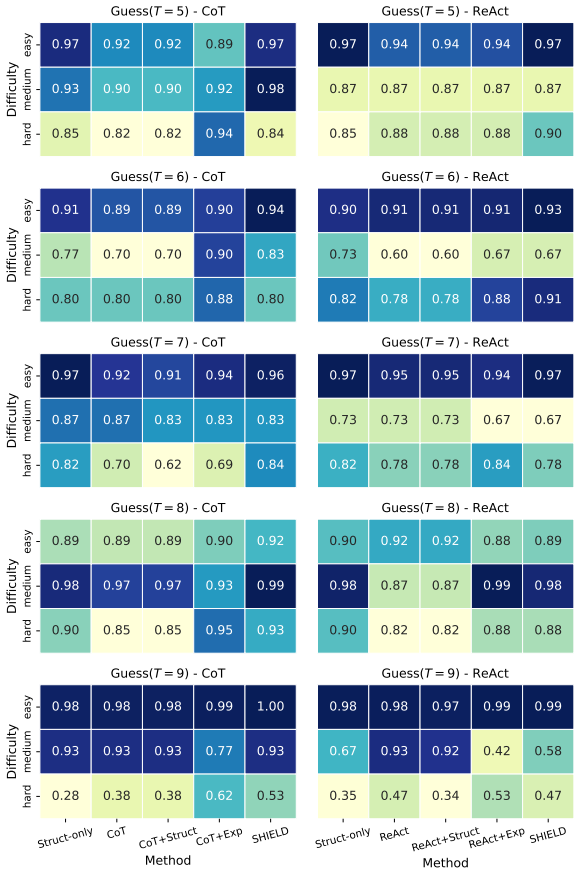

Figure 10: Accuracy of different methods with Deepseek-v3.2 across different difficulty levels in *Say and Guess*.

of difficulty perceptions between DeepSeek-V3 and DeepSeek-V3.2 under the same reasoning paradigm, while samples outside these intersections are categorized as 'out'. We additionally explore cross-paradigm intersections between CoT and ReAct, but the resulting medium subset became excessively small due to substantial disagreement in difficulty perception. Therefore, we focus on intra-paradigm consensus results.

Using *Find the Spy* as the representative task, Figure 11 shows that the superiority of *SHIELD* consistently holds across the entire consensus spectrum. The performance trends on consensus easy, medium, and hard subsets largely align with those observed under the original difficulty partitions. Notably, even on out cases, where difficulty estimates differ across models, *SHIELD* still maintains strong performance, typically exceeding 70% accuracy. These results further demonstrate the robustness and generalizability of *SHIELD* across different LLM backbones and reasoning paradigms.

E Prompts

As noted previously, all prompt examples presented here are translated from their original Chinese versions. The full prompts and implementation code are provided in the supplementary materials as a compressed archive.

From Figure 12 to Figure 21, we use *Find the Spy* as a representative case and present the core prompt templates. In these templates, symbols enclosed in "<>" indicate placeholders to be filled with task-specific content.

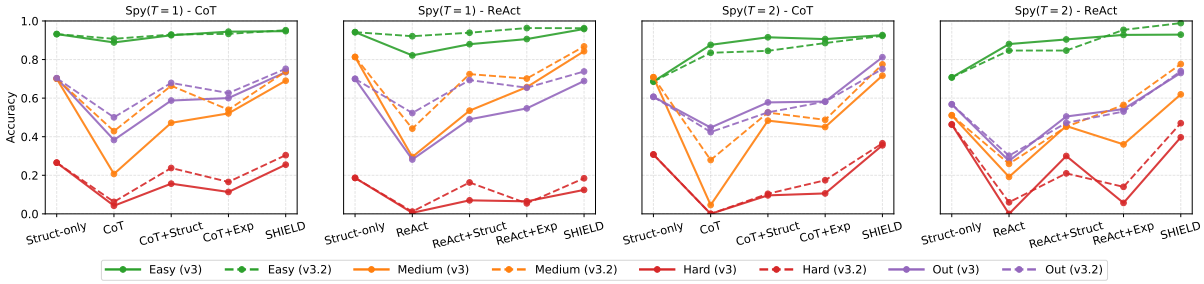

Figure 11: Performance on consensus difficulty partitions of *Find the Spy*. Easy/medium/hard subsets correspond to samples with identical difficulty classification across two models, and out represents discrepant difficulty judgments.

// The following instructions define the global simulation setting and are shared across the simulation, experience distillation, and reasoning stages.
Game Rules: The game consists of <num_of_players> players:
- Folks (<num_of_folks>): receive the same secret word.
- Spy (1): receives a different but semantically related word.
Each player only knows their own word and has no access to the words or identities of others.
Objectives: Identify adversarial opponents through multi-round descriptions and voting.
- Folks: identify and eliminate the spy through voting.
- Spy: conceal identity and avoid elimination.
Global Constraints:
- Never explicitly mention your assigned word.
- Avoid repeating descriptions used by previous players.
- Reason carefully before answering, but only output the required JSON object.

// Agent role prompt.
You are player <name> participating in a game with players: <players>. You are a regular player, NOT a referee, host, or AI assistant.
Your assigned word is <my_word>. You are the <statement_number>-th player to provide a description in the current round.

// Description phase.
To convince others that you are a folk player, provide a natural and informative description of your word without revealing it directly.
If you suspect that you may be the spy, you may strategically describe the potential folk word instead of your own word to improve credibility.
Current round: <round>
Respond strictly in the following JSON format:
{"description": "your descriptio"}

// Attitude update phase.
Player <description_player> described their word as: <description>.
Analyze whether the player's description is semantically aligned with yours.
Given that there is only one spy and all folks share the same word, infer your own possible identity by comparing all previous descriptions.
If you suspect that you are the spy, infer the most likely folk word.
Update your attitudes toward other remaining players in the following JSON format:
{"for": [...], "against": [...], "no comment": [...]}
Constraints:
1. Each player must appear in exactly one category ("for" = trust/support, "against" = suspect/oppose, "no comment" = neutral).
2. Do NOT include yourself in any list.

// Voting Phase.
The player receiving the highest number of votes will be eliminated.
Based on previous descriptions, your strategic reasoning, and your current attitudes toward others, select the player most likely to be the spy.
Respond strictly in the following JSON format:
{"vote": "player_name"}

Figure 12: Examples of simulation prompt templates for *Find the Spy*.

// The following global reasoning context is shared across the reasoning trajectory generation stage and inference stage.
<background>
In this task, you are given partial logs from a single game match. The logs contain only three types of behavioral signals:
1. Descriptions: semantic clues provided by players about their assigned words.
2. Attitudes: expressed support, suspicion, or neutrality toward other players.
3. Voting Ballots: players' final elimination choices.
Crucial Constraint: The provided logs cover only Round 1. Do NOT assume or infer any future rounds, hidden outcomes, later discussions, or whether the game continues. Your judgment must rely solely on the observed evidence.

// Demonstration examples. For interaction trajectory collection and vanilla reasoning settings without the experience pool, the demonstration examples are manually constructed.
Two diagnostic examples are provided below to illustrate how to identify the Spy from historical game logs:
<examples>

// Current game state.
Analyze the following game configuration and interaction logs:
- Total Players: <players>
- Global Category: <category>
Round 1 Logs:
- Player Descriptions: <descriptions>
- Attitudes: <tendencies>
- Voting Ballots: <votes>

// Reasoning instruction.
You are an independent external analyst of this match, NOT a participating player or the game host.
Reason carefully by jointly analyzing semantic consistency, behavioral alignment, and voting patterns. Avoid introducing unsupported assumptions.
First, provide a concise analytical summary covering the following aspects:
- Semantic Deviation: identify descriptions that diverge from the shared semantic pattern.
- Attitude Consistency: analyze whether players' attitudes align with collective behavior or reveal strategic inconsistencies.
- Voting Logic: identify key motivations and anomalous voting behaviors.
// If this is not the first attempt, incorporate the prior reflection.
During analysis, consider the following advice: "<advice>".
Finally, output your definitive judgment strictly in the following JSON format:
{"spy": "player_name"}

Figure 13: Examples of prompt templates used during reasoning trajectory generation stage for Spy($T = 1$). This prompt is also used for vanilla CoT or ReAct reasoning.

. . .
Crucial Constraint: Currently, logs for the first TWO rounds are provided. Do NOT assume, extrapolate, or hypothesize about future rounds, potential game outcomes, subsequent statements, or whether the match continues. Your judgment must rely entirely on the observed evidence.
You MUST strictly follow the following deterministic rules:
- The player eliminated in Round 1 is always a folk.
- The eliminated player is the one with the highest vote count in Round 1. In case of a tie, cross-reference with the Round 2 speaker list; the missing player is the one eliminated.
- The eliminated player's description belongs to a shared semantic space and can be used to infer the global semantic direction of the game. It is not an exact word but is semantically aligned with the common word.
. . .
Reason carefully by jointly analyzing semantic consistency, behavioral alignment, and voting patterns. Avoid introducing unsupported assumptions.
Provide a concise analytical summary addressing the following dimensions:
- Consensus Base: infer the shared semantic direction from the eliminated player's description.
- Semantic Deviation: identify players whose descriptions deviate from this semantic space.
- Round 2 Dynamics: describe players who strengthen, deviate, or remain stable in semantics.
- Behavioral Anomalies: highlight any abnormal voting or attitude patterns (if any).
- Definitive Evidence: identify the most likely spy based on available evidence.
// If this is not the first attempt, incorporate the prior reflection.
During analysis, consider the following advice: "<advice>".
Finally, output your definitive judgment strictly in the following JSON format:
{"spy": "player_name"}

Figure 14: Examples of prompt templates used during reasoning trajectory generation stage for Spy($T = 2$).

——Example 1——
—-Input—-
Players: ["Alice", "Bob", "Carol", "Daniel"]
Global theme: Natural phenomena
Round 1 Game Logs:
Descriptions:
- Alice: "Afterglow"
- Bob: "The East"
- Carol: "Twilight"
- Daniel: "Sunset Clouds"
Attitudes:
- Alice: supports Carol, Daniel; suspects Bob
- Bob: supports Carol; suspects Alice
- Carol: supports Alice, Daniel; suspects Bob
- Daniel: supports Alice, Carol; suspects Bob
Voting Ballots:
- Alice votes for Bob
- Bob votes for Alice
- Carol votes for Bob
- Daniel votes for Bob
—- Analysis —-
- Semantic Deviation: The descriptions "screen", "call", "portable", and "charging" all relate to electronic products, though "portable" is comparatively vague.
- Attitude & Voting Dynamics: Alice and Bob consistently suspect Daniel, while Daniel votes defensively for Bob. Carol's voting behavior deviates from the group consensus.
- Definitive Evidence: Considering both semantic and behavioral signals, Carol is the most likely spy.
—— End of Example 1 ——

Figure 15: A manually constructed CoT reasoning example for Spy($T = 1$).

——Example 1——
—-Input—-
Players: ["Alice", "Bob", "Carol", "Daniel"]
Global theme: Audiovisual entertainment
Round 1 Game Logs:
Descriptions:
- Alice: "Broadcast"
- Bob: "Binge-watching"
- Carol: "Anime series"
- Daniel: "Premiere"
Attitudes:
- Alice: supports Bob; suspects Daniel,Carol
- Bob: supports Alicke; suspects Daniel
- Carol: supports Alice; suspects Bob
- Daniel: supports Carol; suspects Bob
Voting Ballots:
- Alice votes for Bob
- Bob votes for Carol
- Carol votes for Bob
- Daniel votes for Bob
Round 2 Game Logs:
Descriptions:
- Alice: "Satellite TV"
- Carol: "Product placement"
- Daniel: "Film reel"
Attitudes:
- Alice: suspects Daniel
- Carol: suspects Daniel
- Daniel: suspects Alice
Voting Ballots:
- Alice votes for Daniel
- Carol votes for Daniel
- Daniel votes for Alice
—- Analysis —-
thought1: (Semantic clustering in Round 1) Group descriptions to identify potential semantic majority–minority structure.
action1: Perform semantic grouping.
observation1: "Broadcast", "Binge-watching", and "Anime series" strongly converge toward episodic/TV show content. Daniel's "Premiere" distinctly aligns with a cinematic/movie context, forming a clear 3:1 semantic structure.
thought2: (Validation using elimination information) Use the eliminated player's identity to validate semantic grouping.
action2: Incorporate elimination signal for verification.
observation2: Bob is eliminated and confirmed as a folk; his description "Binge-watching" aligns with the television semantic cluster, confirming a valid 3-person folk group. Daniel remains the only consistent outlier.
thought3: (Semantic evolution analysis) Check for persistence or correction of semantic deviation across rounds.
action3: Compare cross-round semantic transitions.
observation3: Remaining players actively reinforce the TV broadcasting theme: Alice transitions to "Satellite TV" and Carol to "Product Placement". Meanwhile, Daniel doubles down on his cinematic theme with "Film Reel", demonstrating continuous semantic deviation.
thought4: (Behavioral validation) Use behavioral signals as secondary evidence when semantic structure is clear.
action4: Analyze attitude and voting consistency.
observation4: Daniel is consistently suspected and fails to integrate into the dominant semantic group; voting behavior remains defensive.
thought5: (Final decision) With a stable semantic cluster and persistent deviation, directly identify the spy.
Final answer: Daniel is the most likely spy.
—— End of Example 1 ——

Figure 16: A manually constructed ReAct reasoning example for Spy($T = 2$).

You are analyzing the reasoning trajectory of a "Find the Spy" instance (a 4-player game with exactly one Spy).
The game configuration includes:
- Player descriptions
- Expressed attitudes (suspicion or support) among players
- Final voting ballots
- Step-by-step reasoning logs and the definitive judgment
—— Reasoning Trajectory ——
<reasoning_trajectories>
—— End of Trajectory ——
Your task is to extract a single reusable reasoning experience from this trajectory.
The extracted experience should capture an observable and generalizable pattern, together with a corresponding inference strategy that can be reused in future games.
Requirements:
1. The experience must belong to exactly one of the following types:
- semantic: based on relationships among descriptions
- attitude: based on support/suspicion relations
- voting: based on voting behavior patterns
2. Each experience must rely on only one type of information source (no mixing).
3. Do NOT mention any player names, specific words, or concrete categories.
4. Do NOT summarize the game; instead, extract a reusable reasoning rule.
5. The "pattern" must describe a clear and observable structure, not vague intuition.
6. The "inference" must specify an actionable decision rule.
7. Avoid vague statements (e.g., "complex situation", "consider multiple factors").
8. Keep the output concise ($\leq$40 words per field).
9. Output exactly ONE most important experience.
Output format (JSON):
{"type": "semantic" / "attitude" / "voting",
"pattern": "observable structural pattern",
"inference": "decision rule for identifying suspicious players"}
Good examples: {"type": "semantic",
"pattern": "Most players show similar descriptions while one deviates significantly",
"inference": "Prioritize the outlier as the suspect"}
{"type": "voting",
"pattern": "A player consistently follows majority voting without deviation",
"inference": "This player is likely concealing identity"}
Bad examples (avoid):
- pattern: "some descriptions are different" (too vague)
- pattern: "mentions food-related words" (not generalized)
- inference: "need comprehensive analysis" (not actionable)

Figure 17: Examples of prompt templates used during insight extraction for *Find the Spy*.

You will receive multiple heuristic rules of the same modality (where the "type" is identical) as formatted below:
<same_type_insights>
Your task is NOT to summarize them, but to perform **deduplication and compression** under the following rules:
1. Merge experiences that are semantically equivalent or only differ in wording.
2. Preserve the underlying "structural pattern" rather than surface expressions.
3. Remove duplicate or redundant experiences.
4. Do NOT add new information or extend reasoning beyond the input.
5. The number of outputs must be $\leq$ 30% of the original number of experiences.
6. Do NOT generate overly general or abstract "meta-experiences".
Output format (JSON):
{
"experience1":{"pattern":"observable structural pattern",
"inference":"decision rule for identifying the target player type"},
"experience2":{...},
...}

Figure 18: Examples of prompt templates used during insight intra-class abstraction for *Find the Spy*.

You will receive multiple pre-abstracted experiences:
<all_insights>
Your task is to further perform **high-level abstraction and unification** according to the following rules:
1. If multiple experiences share the same core structural logic (e.g., "majority agreement + single outlier"), they must be merged into one.
2. Differences in surface attributes (e.g., function, wording, abstraction level) should be treated as variations of the same underlying pattern, not separate rules.
3. The "pattern" should be elevated to a higher level of abstraction that can generalize across different scenarios, but must remain concrete and interpretable (not overly vague).
4. The "inference" must remain a single, unified decision rule without redundancy.
5. After merging, assign each rule to one of the following categories:
- semantic: based on relationships among descriptions
- attitude: based on support/suspicion relations
- voting: based on voting behavior patterns
Maximize generalization capability while minimizing the total entry count of the output rule set.
Output format(JSON):
{
"experience1":{"type": "semantic / attitude / voting",
"pattern":"observable structural pattern",
"inference":"decision rule for identifying the target player type"},
"experience2":{...},
...}

Figure 19: Examples of prompt templates used during insight cross-category aggregation for *Find the Spy*.

. . .
// The examples are randomly sampled from simple reasoning trajectories. In our experiments, we use two such examples as demonstrations.
Some diagnostic examples are provided below to illustrate how to identify the spy from historical game logs:
<example_reasoning_trajectories>
The following are reasoning experience guidelines that can help improve your accuracy in identifying the spy:
<insights>
These experiences summarize common and reusable reasoning patterns. During analysis, if the current game state matches any of these patterns, prioritize applying the corresponding experience when making your judgment.
. . .

// The following prompt incorporates structural knowledge and is used as a general template on Spy ($T = 1$).
A structural expert has analyzed the current state and concluded that the spy is "<agent:>". Its judgment typically carries an exceptionally high reference value.
You are required to perform an **consistency check** based on your current analysis, following the rules below strictly.
Step 1: Consistency Check
Determine whether the expert's prediction is consistent with your current conclusion:
- If consistent: the result is mutually reinforced and can be used as the final answer.
- If inconsistent: proceed to Step 2.
Step 2: Conflict Analysis
When there is a disagreement, evaluate credibility from the following three aspects only:
1. Semantic evidence strength
- Are there clear semantic clusters among descriptions?
- Is there a single obvious outlier?
- If there is a clear semantic anomaly → your current reasoning is more reliable.
- If semantics are ambiguous or multiple interpretations exist → expert is more reliable.
2. Behavioral consistency
- Is there a stable majority pattern in attitudes or votes?
- If majority behavior clearly converges on a candidate consistent with your reasoning → your reasoning is more reliable.
- If behavior is fragmented or lacks a clear trend → expert is more reliable.
3. Conflict complexity
- Are there multiple conflicting signals or multiple deviating players?
- If only one clear anomaly exists → your reasoning is more reliable.
- If multiple conflicting signals exist → expert is more reliable.
Decision rule (must follow):
- If both semantic and behavioral evidence strongly support your current conclusion → keep your prediction.
- Otherwise (uncertainty or conflict) → adopt the expert's prediction.
Finally, output your final decision in the following format:
{"spy": "player_name"}

Figure 20: Examples of prompt templates used in *SHIELD* reasoning for Spy($T = 1$).

. . .
// The examples are randomly sampled from simple reasoning trajectories. In our experiments, we use two such examples as demonstrations.
Some diagnostic examples are provided below to illustrate how to identify the spy from historical game logs:
<example_reasoning_trajectories>
The following are reasoning experience guidelines that can help improve your accuracy in identifying the spy:
<insights>
These experiences summarize common and reusable reasoning patterns. During analysis, if the current game state matches any of these patterns, prioritize applying the corresponding experience when making your judgment.
. . .

// The following prompt incorporates structural knowledge and is used as a general template on Spy ($T = 2$).
A structural expert has analyzed the current state and concluded that the spy is "<agent:>". Its judgment typically carries an exceptionally high reference value.
You are required to perform an **consistency check** based on your current analysis, following the rules below strictly.
Step 1: Consistency Check
Determine whether the expert's prediction is consistent with your current conclusion:
- If consistent: the result is mutually reinforced and can be used as the final answer.
- If inconsistent: proceed to Step 2.
Step 2: Conflict Analysis
When there is a disagreement, evaluate credibility from the following four aspects only:
1. Semantic community structure
- Do descriptions across both rounds form a stable majority semantic cluster (3:1)?
- If a clear majority cluster exists and your prediction points to the minority → your reasoning is more reliable.
- If the semantic structure is unclear → the expert is more reliable.
2. Semantic anomaly strength
- Is there a clearly deviating description from the shared semantic space?
- If there is a single strong anomaly → your reasoning is more reliable.
- If multiple players are ambiguous or deviations are weak → the expert is more reliable.
3. Semantic evolution pattern
- Do players show persistent deviation, sudden correction, or forced alignment with the majority across rounds?
- If a player shows persistent or abnormal evolution → your reasoning is more reliable.
- Otherwise → the expert is more reliable.
4. Behavioral consistency
- Do suspicion and voting patterns converge toward a stable target?
- If behavior consistently supports your prediction → your reasoning is more reliable.
- If behavior is fragmented or unstable → the expert is more reliable.
Decision rule (must follow):
- If at least two of semantic structure, anomaly strength, evolution pattern support your conclusion → keep your prediction.
- Otherwise (uncertainty or conflict) → adopt the expert's prediction.
Finally, output your final decision in the following format:
{"spy": "player_name"}

Figure 21: Examples of prompt templates used in *SHIELD* reasoning for Spy($T = 2$).