Playing with Fire: What Transfers When RL Trains a Language Agent?

Mahesh Ramesh[1]* **Kaousheik Jayakumar**[2]* **Hemanth Ram**[3]
Pavan Thodima[1] **Ramani Duraiswami**[2] **Dinesh Manocha**[2]
Aniket Rege[1] **Emmanouil-Vasileios Vlatakis-Gkaragkounis**[1]
[1]University of Wisconsin–Madison [2]University of Maryland, College Park
[3]The University of Texas at Austin

Abstract

What does reinforcement learning (RL) teach an agentic language model—*task-transferable competence* or *scaffold-specific behavior*? We study this question in *Hanabi*, a cooperative hidden-information card game, by training and evaluating models across three Holmesian scaffolds that present the same game with different prompt support: WATSON supplies the public game state, visible hands, clue-derived own-card information, and legal moves; SHERLOCK adds Hanabi rules and engine-computed deductions over possible card identities; and MYCROFT requires the model to update per-player belief states from intervening actions before acting. Our results reveal that a QWEN3-4B model with weak scaffold-independent Hanabi competence, trained with RL on the supported SHERLOCK and MYCROFT scaffolds, achieves large in-domain gains but fails to transfer to held-out WATSON, where rules are not provided in context. In contrast, GPT-OSS-120B, which starts with stronger scaffold-independent Hanabi competence, improves on both held-out WATSON and SHERLOCK after RL on MYCROFT alone. To evaluate long-horizon state tracking, we construct FIREWORKS, a 10K benchmark requiring exact reconstruction of per-player hidden card beliefs across 2–5 player games, where QWEN3-4B has near-zero Pass@1. Training on FIREWORKS yields QWEN-ST, which improves SHERLOCK performance over QWEN3-4B despite using the same 4B backbone, showing that the 4B model can benefit from RL when the scaffold supplies the missing Hanabi rules and deductions. These results suggest that cross-scaffold transfer is gated by pre-RL task competence: playing Hanabi well without prompt-supplied rules rather than by model scale alone. QWEN-ST also produces longer, more persistent rollouts and improves held-out non-Hanabi PI-LOGIC exact-answer Avg@8/Pass@8 from $0.03/0.15$ to $0.13/0.37$, with continued Hanabi RL raising them to $0.20/0.61$, even though the exact belief-reconstruction skill does not transfer zero-shot to MYCROFT. When MYCROFT state-tracking and move-rating rewards are optimized jointly, cold-start training is slowed by reward interference, while initialization from FIREWORKS reaches the same state-tracking reward level in roughly half as many RL steps. Together, our results separate RL post-training into three outcomes: *scaffold-local gains*, *competence-gated transfer*, and *computation-style/trainability transfer*, where hard-task RL changes rollout behavior and improves downstream optimization even when the trained zero-shot skill does not transfer across scaffolds.

*Equal contribution.

ICML 2026 Workshop: RLxF: Reinforcement Learning from World Feedback

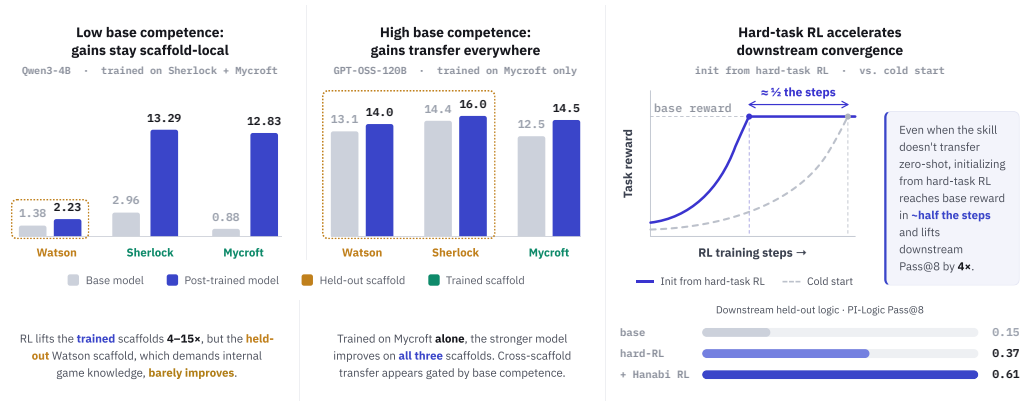

Figure 1: Benefits of RL post-training of LLMs to play Hanabi do not always transfer across scaffolds. Transferability depends on the base model's pretrained, scaffold-independent task competence. **Left:** LLMs with low base Hanabi competence improve only on the trained scaffold. **Center:** LLMs with high base competence transfer post-training gains across scaffolds. **Right:** Hard-task state-tracking RL does not transfer its state-tracking skill zero-shot to a different scaffold (Section 3.3), but the resulting checkpoint produces longer, more persistent rollouts and helps downstream in two distinct ways. First, using it to initialize subsequent state-tracking RL, rather than starting from the base model (a cold start), reaches the same state-tracking reward in substantially fewer RL steps. Second, the same checkpoint improves performance on a held-out, non-Hanabi logic task, with continued Hanabi RL improving it further.

1 Introduction

Post-training with reinforcement learning (RL) has become the default recipe to improve modern language model (LM) reasoning and agentic capabilities [42, 29], where the learning algorithm aims to maximize a scalar reward during training. However, a higher reward or downstream benchmark score does not guarantee that the LM has learned robust and generalizable competence; it may have latched on to idiosyncrasies of the data or training setup [58].

This distinction matters because many popular agentic LM setups for tasks like coding [13] have "scaffolds" built around them [44, 45], *i.e.* a suite of software tools and/or prompt templates that augment the underlying language model. For example, in software engineering and terminal control benchmarks like SWE-bench [19] and Terminal-Bench [27], reported scores reflect not only on the underlying LM's learned knowledge, but also the scaffold that surrounds it: context selection, tool access, command execution, observation formatting, retry logic, and error recovery [13, 53]. Practitioner reports on SWE-bench Pro [1, 37] and related coding-agent leaderboards [18] similarly emphasize that different scaffolds around the same model can produce substantially different scores, sometimes comparable to or larger than differences between model families.

This scaffold sensitivity is increasingly consequential as agent benchmarks move beyond static question answering toward interactive harnesses [35]. The same underlying environment can be presented with different observations, tools, action formats, and reasoning instructions. If RL training merely teaches the LM how to exploit one scaffold, downstream LM evaluation [28, 15, 20] will likely overstate the model's deployable competence [26]. If RL is instead able to elicit latent task competence that the model already possesses or learns new skills, downstream performance gains are more likely to robustly transfer when the task is presented through a different observation format, or tool interface, even when the underlying environment is unchanged [8].

This leads to the central question of our paper:

$$\textit{When RL improves an LM agent in one scaffold, what actually transfers to another?} \tag{1}$$

We answer this question through *Hanabi*, a game-theoretic multi-agent testbed for cooperative reasoning under hidden information. Hanabi is a cooperative card game that requires belief tracking, coordination, and theory-of-mind reasoning [9]. This setting is especially useful as a scientific testbed because the exactly same game can be exposed through multiple scaffolds, while models must act reliably across 60+ turns to achieve high scores. Since early mistakes can propagate through the trajectory, Hanabi provides a stern test of sustained agent competence. For our study, we

adopt the WATSON, SHERLOCK, and MYCROFT scaffolds[2] introduced in Ramesh et al. [33]. These scaffolds form a spectrum of support: the underlying game remains fixed, but the model receives different state information and must shoulder different degrees of state maintenance before acting. Concretely, the scaffolds differ along four axes (supplied game rules, candidate-move enumeration, engine-computed state deductions, and state formatting) so that future ablations can isolate which component gates transfer. We survey related work on RLVR generalization, scaffold sensitivity, inference-time computation, and cascaded RL in Appendix A.

The contribution of this paper is a decomposition of RL post-training effects in agentic language models. Using Hanabi as a controlled case study, we isolate four claims:

1. **Scaffold-local gains.** When the base model lacks usable scaffold-independent task competence, RL can produce large gains on the trained scaffold without reliable cross-scaffold transfer.
2. **Competence-gated transfer.** When the base model already has task-relevant latent competence, the same style of RL can improve behavior across scaffolds.
3. **Computation-style and trainability transfer.** Even when direct scaffold transfer is imperfect, RL on hard structured tasks can shape generation-time behavior: it produces longer and more persistent rollouts, improves search efficiency by reducing degenerate long failures, and yields initializations that converge faster under subsequent RL.
4. **A strict state-tracking benchmark.** We construct FIREWORKS, a state-transition dataset of 10K examples across 2–5 player Hanabi games. We also collect 18K MYCROFT-scaffold training data via Grok-4.1 Fast Reasoning, enabling verifiable state-tracking rewards during Hanabi gameplay.

2 Problem Setup

Why Hanabi? Hanabi is a useful testbed [33] for behaviors induced in a language model by RL training because the LM must track hidden information (cards) over a long horizon (turns) and choose actions under limited communication with the environment (other players). Hanabi also allows us to examine the impact of the information in the prompt we provide to the LM (see scaffold details below). Player count adds a second axis of variation, since 2P games require different strategic behavior from 4P or 5P games. These properties make Hanabi resemble many practical agentic settings: the LM must maintain state, infer hidden information and act reliably in a dynamic environment, which we exploit to evaluate whether Hanabi score improvements offered by RL training reflect true game competence or narrower adaptation to the scaffold(s) used during training.

Scaffolds. We follow the prompt template scaffold from [33] that vary in how much game knowledge, state information, and deductive support are supplied to the LM:

- **WATSON** provides minimal context: the game state, visible hands, and explicit clue-derived knowledge with no information about Hanabi or game state deductions. WATSON serves as a competence test; strong performance indicates that the LM has internalized Hanabi knowledge.
- **SHERLOCK** provides the richest context to the LM: candidate moves, Hanabi rules and engine-computed deductive information from the Hanabi Learning Environment [14].
- **MYCROFT** provides the LM with the game state at its previous turn and all intervening moves, and the LM must mirror how a human player would reason and use a structured scratch pad to reconstruct the current belief state on its own. Because MYCROFT requires the model to maintain beliefs explicitly, it also serves as a direct test of state-tracking capability.

We use the WATSON and SHERLOCK prompts from [33] without modification. We slightly modify the MYCROFT prompt from [33] to elicit theory-of-mind (ToM) reasoning. Full prompts for all three scaffolds are provided in Appendix K.

Training dataset and weighted rewards. We use QWEN3-4B-INSTRUCT-2507 (henceforth called QWEN3-4B for brevity) and GPT-OSS-120B as base models in our experiments. Our Hanabi RL data contains fixed game states with precomputed reward labels. We use two reward types. The

[2]Full scaffold details are provided in Section 2 & Appendix H. The names are mnemonic and follow the Holmes-family intuition: WATSON receives the least deductive support, SHERLOCK is provided with rich game-state and deduction information, and his brother MYCROFT faces the most demanding interface, requiring belief reconstruction and theory-of-mind reasoning before action.

first is a move-rating based reward, where an LLM judge scores each candidate legal action in $[-1, 1]$. For Sherlock, we use the dataset from [33], which provides move-rating labels generated by OpenAI o3 and Grok-3-mini [31, 49]. For Mycroft, we generate move-rating labels using Grok-4.1 Fast Reasoning [50]. The second is a verifiable state-tracking reward, used in Mycroft, which scores whether the model correctly updates hidden card beliefs between turns. Thus, move-rating is evaluated by an LLM-as-judge reward, while state tracking is evaluated by a deterministic verifier.[3]

For MYCROFT state tracking, we use a weighted verifiable reward that emphasizes changed cards. For card slots $c \in C$, prediction $\hat{s}_c$, target s_c, and weights w_c:

$$R_{\text{st}} = \frac{\sum_{c \in C} w_c \, \mathbf{1}[\hat{s}_c = s_c]}{\sum_{c \in C} w_c}, \qquad w_c = \begin{cases} 1.0 & \text{if the slot changed since the previous state,} \\ 0.1 & \text{if the slot was unchanged.} \end{cases} \tag{2}$$

The unequal weighting is essential: under equal weighting, the model can achieve high reward by copying stale beliefs, since most card slots are unchanged between turns. In all RL experiments, the model trains on fixed datasets of game states and candidate actions with precomputed reward labels.

FIREWORKS state-tracking data. Because gameplay rewards do not allow us to cleanly test whether state-tracking as a standalone skill transfers across scaffolds, we construct a separate, strict state-transition reconstruction dataset, which we call FIREWORKS. In FIREWORKS, the model receives previous and current public state information and must reconstruct the updated hidden-belief state. The reward is strict and verifiable: the model receives reward 1 only if the entire reconstructed hidden-belief state is correct, and reward 0 otherwise. We train QWEN3-4B on 10K FIREWORKS samples to obtain QWEN-ST. We use QWEN-ST to test two additional questions: whether hard state-tracking RL elicits inference-time computation that transfers to tasks where the model already has relevant ability, and whether such hard-task training improves convergence in subsequent Hanabi RL. Example FIREWORKS training instances are shown in Appendix K.4, and data curation recipe in Appendix H.

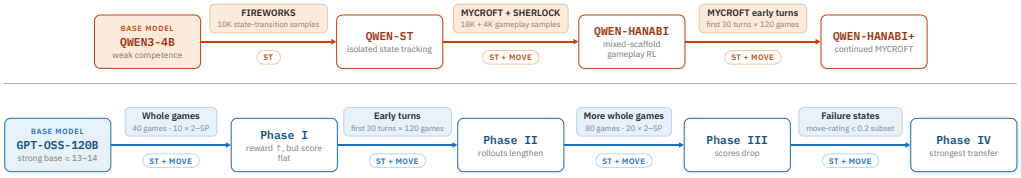

Figure 2: **Multi-stage training.** We RL post-train the QWEN3-4B base model in three stages, yielding three checkpoints: FIREWORKS state-tracking (→ QWEN-ST), mixed SHERLOCK + MYCROFT gameplay data (→ QWEN-HANABI), and continued MYCROFT-only training (→ QWEN-HANABI+). We train GPT-OSS-120B across four phases of MYCROFT data, progressively expanding game coverage (I, III), incorporating early-turn slices (II), and finally targeting low-reward failure cases (IV), yielding four checkpoints. All checkpoints are evaluated for scaffold transfer and deliberation. We train all checkpoints with both state tracking (ST) and move rating (MOVE) rewards, except QWEN-ST (only ST)

We summarize our multi-stage training pipeline in Figure 2. Before training directly for Hanabi gameplay, we target improving the base LLM's (QWEN3-4B) general state-tracking ability, as we observed this improved downstream RL training with MYCROFT (Section 3.4). We thus first train QWEN3-4B on 10K state-transition samples from FIREWORKS to obtain QWEN-ST. We then continue training on mixed SHERLOCK/MYCROFT gameplay data to obtain QWEN-HANABI, to examine whether RL training on Hanabi gameplay rewards (move ratings) transfers to held-out WATSON (Section 3.1). Finally, QWEN-HANABI+ continues training from the QWEN-HANABI checkpoint on the initial 30 turns of 120 MYCROFT games. We use early turns because early belief-state errors propagate through the rest of the game, making this data slice a direct stress test for state tracking and early-game reasoning. This continuation tests whether prolonged optimization toward one trained scaffold improves that scaffold while degrading another (Table 1).

With GPT-OSS-120B (Hanabi score: 13.1 / 25), our overall goal is to examine the degree to which base QWEN3-4B's poor held-out scaffold transfer is due to low pretrained Hanabi playing compe-

[3]This distinction matters for interpreting our claims. The scaffold-transfer and trainability-transfer results (Sections 3.1–3.4) rest on the deterministic state-tracking reward and on held-out Hanabi game scores, not on the judge-based move-rating signal. The move-rating reward is used only to train and evaluate action-selection quality within gameplay; conclusions about cross-scaffold transfer do not depend on its accuracy.

tence (Hanabi score: 1.4 / 25). We train GPT-OSS-120B with only the MYCROFT scaffold and evaluate transfer to held-out WATSON and SHERLOCK (Section 3.2). We train in four phases, which track the relation between reward, rollout length, and held-out performance. Phase I tests whether reward increase alone predicts evaluation improvement. Phase II uses the same initial-turn data slice used in the QWEN3-4B continuation, testing whether the QWEN3-4B failure was caused by the data slice or by limited base competence. Phase III continues on additional full-game samples and tests whether easier samples degrade performance, and Phase IV focuses on low-reward failure states to test whether harder examples preserve deliberation and improve transfer. We use GRPO for all our training [38]. We provide more details of our training setup in Appendix I.

Evaluation. We evaluate all trained Hanabi models on fixed seeds (1001–1010) from the Hanabi Learning Environment [14], across 2–5 player settings and all three scaffolds. We report interquartile mean (IQM) scores with 95% bootstrap confidence intervals; we report averages across player counts by default, provide player count-wise results in Appendix D. For MYCROFT state tracking, we use a 500-prompt validation set with the weighted reward in Equation 2. For FIREWORKS, we use a 61-prompt validation set with the strict exact-match reward. We provide a detailed evaluation setup discussion in Appendix J.

3 Experimental Findings

First, we examine whether RL on structured Hanabi scaffolds transfers to a held-out scaffold when the base model has low Hanabi competence (Section 3.1). Second, we repeat the transfer test with a stronger base model to test whether the failure is competence-gated (Section 3.2). Third, motivated by the large-model training dynamics, we examine what can still transfer when direct cross-scaffold Hanabi competence does not: whether hard state-tracking RL changes inference-time computation, improves performance on tasks where the model already has relevant knowledge or when the knowledge is injected in context, in our case SHERLOCK, even when the learned state-tracking skill does not transfer zero-shot to a different Hanabi scaffold (MYCROFT) (Section 3.3). Finally, we examine whether QWEN-ST is useful not as a zero-shot policy, but as an initialization for later RL, especially under reward interference (Section 3.4).

3.1 RL Training vs. Pre-RL Task Competence

We begin with a simple scaffold transfer test: train on a mixture of SHERLOCK and MYCROFT and evaluate on WATSON, as seen in Table 1. The QWEN3-4B base model is poor across scaffolds; after gameplay RL, our QWEN-HANABI reaches much stronger scores on the trained scaffolds: 2.96 $\rightarrow$ 13.29 on SHERLOCK and 0.88 $\rightarrow$ 12.83 on MYCROFT. WATSON, however, improves over the base model by less than a point (1.38 $\rightarrow$ 2.23), far below competent Hanabi play (See Appendix D for how frontier LMs score on Hanabi). The same trained model is not generally incapable of Hanabi improvement; its gains appear when Hanabi rules, state support, and deductive information are supplied by the scaffold. The failure is specifically transfer to WATSON, which requires the model to have game knowledge from pre-training.

Table 1: Comparison of Hanabi scores across scaffolds of our RL-trained models over the base model with a Qwen3-4B backbone. Δ denotes the score gain of QWEN-HANABI+ over QWEN-HANABI.

Model	WATSON	SHERLOCK	MYCROFT
QWEN3-4B Base	1.38	2.96	0.88
QWEN-HANABI	2.23	**13.29**	12.83
QWEN-HANABI+	2.46	11.00	**13.38**
Δ	+0.23	−2.29	+0.55

We note that lack of cross-scaffold transfer is not simply a short-training artifact. Across the Hanabi training, QWEN3-4B sees roughly 23K Hanabi RL samples, including structured SHERLOCK/MYCROFT training, yet the gain remains largely "scaffold-local" *i.e.*, improving only on the trained scaffold. The model improves when the scaffold exposes enough state and reasoning structure, but it does not acquire the scaffold-independent Hanabi competence needed to transfer to WATSON. This suggests that RLVR is inefficient at injecting the internal Hanabi knowledge and state abstractions required by WATSON, even though it can strongly improve performance under the trained scaffold.

There is a second sign of scaffold dependence even within the training scaffolds. After observing poor transfer to held-out WATSON, we ask whether further optimization toward one trained scaffold remains compatible with the other. To test this, we continue training QWEN-HANABI on additional MYCROFT data to obtain QWEN-HANABI+. We use only the initial 30 turns because early

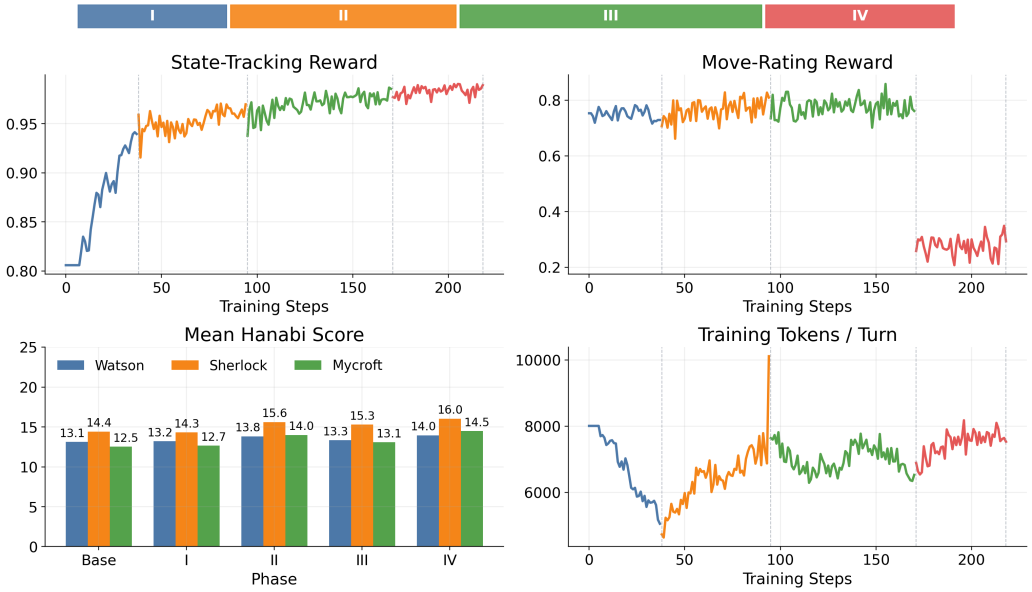

Figure 3: Training dynamics of GPT-OSS-120B on MYCROFT data across phases (Figure 2). The top panels visualize training rewards, the bottom-right panel plots training rollout completion length, and the bottom-left panel charts the mean of the IQMs across 2-5P setups for WATSON, SHERLOCK, and MYCROFT in each phase.

belief-state errors can propagate through the rest of the game, and improving models on early turns improves the Hanabi scores. We choose to train on 120 games with 30 turns to strike a balance between covering diverse game trajectories and obtaining enough per-game continuation data, balanced across 2P–5P settings (yielding 120×30=3600 training samples). As shown in Table 1, this continued training slightly improves MYCROFT from 12.83 to 13.38, but degrades SHERLOCK from 13.29 to 11.00. The full breakdown in Appendix D shows that the drop in QWEN-HANABI+ is concentrated in lower-player regimes: SHERLOCK 2P falls from 11.00 to 5.33 and 3P from 14.67 to 11.83, while 4P is roughly unchanged and 5P drops mildly. Thus, scaffold dependence appears not only in transfer to held-out WATSON, but also between the multiple scaffolds used to train a single model. This is why Hanabi is a useful testbed for our question 1: the same underlying game can be evaluated under different scaffolds and under different player counts, which change the strategic structure. The issue is not only transfer to WATSON, extra MYCROFT optimization helps MYCROFT, but does not reliably transfer back to SHERLOCK.

3.2 Cross-scaffold transfer is competence-gated

Hypothesis: Poor scaffold transfer is due to low scaffold-independent Hanabi competence in the base LLM; a base LLM with stronger competence will allow RL gains to survive scaffold changes and generalize well.

To verify our hypothesis, we train a large LM with broader pretrained knowledge, GPT-OSS-120B, on MYCROFT data. The base model already scores around 13–14 across scaffolds, far above QWEN3-4B. As shown in Figure 3 (bottom left), after RL training on only the MYCROFT scaffold, the final Phase IV GPT-OSS-120B model improves on all three scaffold families: WATSON from 13.1 to 14.0, SHERLOCK from 14.4 to 16.0, and MYCROFT from 12.5 to 14.5. Unlike QWEN3-4B, GPT-OSS-120B improves across both trained and held-out scaffolds after RL training on only a single scaffold, MYCROFT. We train GPT-OSS-120B in four phases (Figure 2) to study when RL gains transfer across scaffolds. In Phase I, we train 10 games from each player-count setting (total 40). The overall reward increases during training, but Mean Hanabi Score changes little. Thus, motivating a targeted Phase II continuation rather than simply more whole-game training to improve the model performance.

In Phase II, we continue GPT-OSS-120B on the initial 30 turns from 120 games. Similar to the QWEN3-4B continued training experiment (QWEN-HANABI -> QWEN-HANABI+) (Section 3.1), we focus on these initial turns because early belief-state errors can propagate through later turns and significantly lower the final score. This continuation also lets us verify whether the QWEN3-4B

failure to transfer back to SHERLOCK reflects limited model competence rather than a problem with the initial-turn data. Unlike QWEN3-4B, the same type of continued training improves GPT-OSS-120B across all scaffolds Figure 3. Because the same early-turn continuation degrades SHERLOCK in QWEN3-4B but improves it in GPT-OSS-120B, the issue is unlikely to be the data slice alone; it is tied to the smaller base model's limited scaffold-independent Hanabi competence.

Phases I and II show different training dynamics (Figure 3), average rollout length decreases during Phase I but increases during Phase II. To test whether this difference is caused by the data regime, we run Phase III on additional whole-game (similar to Phase I) trajectories: 80 games total, with 20 games from each 2P–5P setting. Mean Hanabi scores drop in Phase III, even though move-rating reward does not decrease. Average rollout length also falls relative to the Phase II checkpoint. This suggests a possible explanation: whole-game training may include many easier or already-solved states, reducing the pressure to deliberate and lowering mean Hanabi scores despite stable training reward.

Phase IV tests this hypothesis directly. We curate failure data by selecting samples where the model's move-rating reward is below 0.2, i.e., states on which the current model receives low action-quality reward. These are harder, low-confidence states under the training reward. Training on this failure-focused subset increases rollout length again and produces the strongest evaluation performance across all three scaffolds. This interpretation is consistent with recent RLVR work showing that prompts at extreme success rates provide a weak learning signal, motivating online difficulty filtering or dynamic sampling toward intermediate-difficulty examples [7, 55].

Figure 3 shows the full phase trajectory. Across phases, higher training reward alone is not sufficient for cross-scaffold transfer; tracking reward alone would miss whether RL is still eliciting useful computation under hard states. The phases with stronger held-out gains are the phases that train on harder or lower-confidence states and maintain longer rollouts, while phases with weaker transfer reduce rollout length. This suggests that generated training length is a useful diagnostic (though not by itself a causal explanation) for whether RL is still eliciting deliberative computation, rather than merely optimizing reward on states the model can already solve with shorter completions.

3.3 What transfers: not direct state tracking, but computation tendency

Hypothesis: RL on a hard task like state-tracking may not produce zero-shot transfer when the same state-tracking problem is evaluated in a different scaffold (FIREWORKS trained model -> QWEN-ST evaluated in MYCROFT). However, it may still transfer a tendency to allocate more inference compute and persist through difficult reasoning: the model may produce longer rollouts and improve on tasks where it already has sufficient relevant knowledge, or when knowledge is injected through prompts.

Our phased training study of GPT-OSS-120B (Section 3.2) suggested that the stronger transfer coincides when RL is applied to harder, lower-confidence states, and that these gains coincide with longer training rollouts. We now explicitly test this hypothesis on QWEN3-4B using FIREWORKS, a strict state-transition dataset we create that is separate from Hanabi gameplay: different setup (See Appendix K.4), no move selection is involved, and the model receives a reward only if every card is tracked correctly. We train QWEN3-4B on 10K samples from FIREWORKS across 2P–5P settings to obtain QWEN-ST.

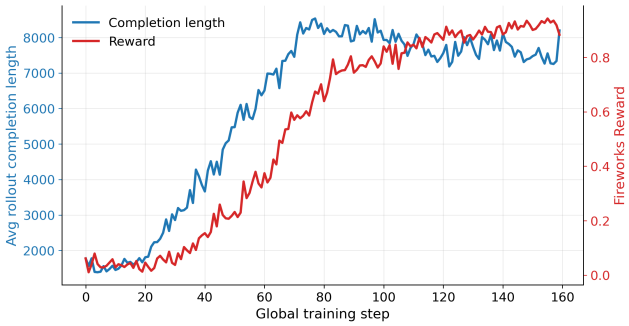

Figure 4: FIREWORKS state-transition training base QWEN3-4B → QWEN-ST. Reward & LM response length rise together.

The task is difficult for the base model: on a held-out 5P FIREWORKS validation set, QWEN3-4B solves only 10 of 61 prompts even at Pass@1024, with Pass@1 essentially zero (Appendix E). During FIREWORKS RL, generated length rises from roughly 1.5K tokens to over 8K tokens while reward rises from near zero to high accuracy (Figure 4).

We test whether this computation tendency helps outside Hanabi using hard held-out logic questions. Table 2 evaluates 100 examples from the PI-LOGIC subset of INTELLECT-3-RL [32]. These examples contain no Hanabi content. We sample 8 completions per example and score by answer match. See Appendix J for more evaluation details.

Both Hanabi-derived checkpoints improve PI-LOGIC performance and produce longer completions. QWEN-ST raises Pass@8 from 0.15 to 0.37, with average length increasing from 5962 to 6360 tokens. QWEN-HANABI+ raises Pass@8 further to 0.61, with average length increasing to 7198 tokens. These improvements come from the same 4B backbone, trained on state tracking and Hanabi gameplay respectively, yet transferring to non-Hanabi logic tasks. This supports the interpretation that RL on a difficult task (state-tracking here) can transfer computation style to tasks where the model already has relevant reasoning ability.

We use completion length as a diagnostic for changed inference-time computation, not as a causal explanation by itself. In Appendix G.1, we discuss instances where the base model struggles to get the correct answer even after spending 4x as many tokens as the RL-trained model. This shows that RL not only increases deliberation but also improves search efficiency and overcomes the attractor state that the base model suffers from.

Table 2: We evaluate QWEN-ST and QWEN-HANABI+ (Figure 2) on held-out 100 PI-Logic examples. The evaluations are done with 8 completions per example. We also compare the Hanabi scores of the 3 models, averaged over 2-5 player settings across 10 held-out seeds.

Model	PI-LOGIC				Hanabi		
	Avg@8	Pass@2	Pass@4	Pass@8	WATSON	SHERLOCK	MYCROFT
QWEN3-4B base	0.03	0.05	0.09	0.15	1.38	2.96	0.88
QWEN-ST	0.13	0.20	0.28	0.37	2.12	7.25	2.84
QWEN-HANABI+	**0.20**	**0.31**	**0.45**	**0.61**	**2.46**	**11.00**	**13.38**

The QWEN-ST checkpoint also improves average Hanabi score over the base model across all three scaffolds. The largest gain is on SHERLOCK (+4.29), where the prompt provides richer state support and mainly tests action quality. This result helps explain the pattern in Section 3.1: QWEN3-4B is not generally incapable of Hanabi improvement; it improves most when the scaffold supplies the missing game knowledge and state support, allowing RL-induced persistence to focus on action selection, supporting the hypothesis the model lack of knowledge is reason for non generalization not scale.

The gain on MYCROFT is smaller, which raises a key question: did the FIREWORKS state-tracking skill transfer into MYCROFT?

This direct transfer test fails. But the failure is specifically about the exact state-tracking skill under a different prompt schema, not about all useful transfer from FIREWORKS. On a 500-sample, 5P MYCROFT state-tracking probe using the weighted card reward from Equation 2, QWEN-ST scores slightly below the base model: 0.117 versus 0.127. This shows that the state-tracking skill learned in FIREWORKS does not directly transfer zero-shot into the MYCROFT scaffold. The strict transition task asks the model to update a previous belief state under a narrow schema, while MYCROFT asks the model to perform state tracking inside a richer gameplay prompt with theory-of-mind instructions and action selection. Appendix K.4 and Appendix K.3 show the two prompt formats, and Appendix F analyzes why QWEN-ST scores lower than QWEN3-4B on the MYCROFT state-tracking probe. Once again, shows that when trained on hard tasks, where the base model's reasoning ability is low, the RL training on an academic scale does not transfer across scaffolds. The positive transfers (SHERLOCK and PI-LOGIC gains, and the trainability benefit shown in Section 3.4) all survive this zero-shot schema failure.

As a reverse control, RL training on PI-LOGIC does not improve held-out Hanabi performance, suggesting that task difficulty alone is insufficient. Useful transfer appears tied to the structured state-maintenance pressure that FIREWORKS repeatedly applies; the more heterogeneous PI-LOGIC prompts may not provide this sustained pressure as directly. We discuss this further in Appendix G.2.

3.4 Reward interference and cascading

Hypothesis: A hard task RL checkpoint may fail as a zero-shot policy under a new scaffold, but still provide a better initialization for continued training by inducing longer rollouts and better exploration.

Trainability transfer, the idea that a checkpoint can be valuable not as a zero-shot policy but as a better initialization for subsequent RL is one of the most practically relevant outcomes of hard-task training. The previous section 3.3 showed that QWEN-ST does not directly transfer as MYCROFT state-tracking skill; it is slightly worse than the base model on the MYCROFT state-tracking probe, but the RL-trained model performs better with more context compared to the base model. We now ask whether we can utilize this checkpoint for later RL training. The benefit we test for is faster optimization and whether longer rollouts will help with exploration during continued training.

We test this on the 40-game/2.4K-sample MYCROFT dataset described in Appendix I. Figure 5 combines the first training phase and the joint continuation phase. The dashed vertical line marks the start of Phase II, where selected checkpoints are continued with the joint state-tracking (ST) and move-rating (MR) objective.

The clean cascade comparison is between the QWEN-ST-initialized run and the base-initialized state-tracking run. In Phase I, both train on the same MYCROFT ST objective and the same data; the only difference is initialization. Although QWEN-ST does not start with a higher MYCROFT state-tracking score (check Appendix F), it produces much longer rollouts and learns the state-tracking objective substantially faster; QWEN3-4B takes 2x steps to reach the same reward. This provides evidence that the FIREWORKS-trained checkpoint transfers not as a zero-shot state-tracking policy, but as a better initialization for subsequent MYCROFT RL.

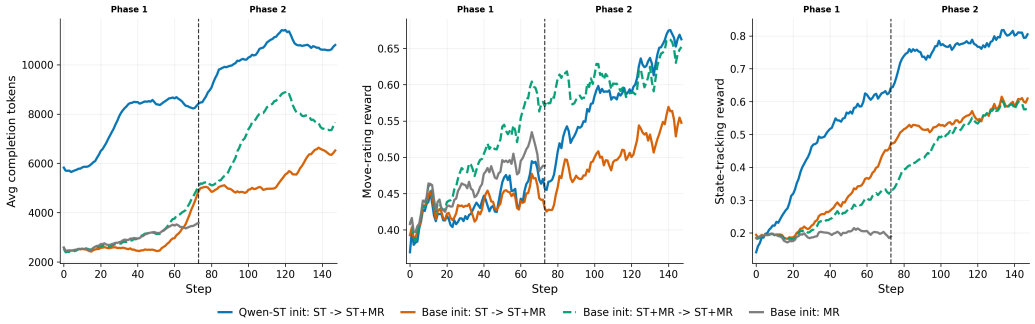

Figure 5: Here we compare the rollout length (**Left**) and rewards for move-ratings (**Middle**) and state-tracking (**Right**) for different base model initializations (QWEN-ST and Base QWEN3-4B) and reward formulations of multi-stage training of QWEN3-4B on MYCROFT data. The dashed-vertical line at the center splits training into 2 phases which use one of three rewards: a State-tracking reward (ST), a move-rating reward (MR), or a combined reward (ST + MR). Each entry in the legend is labeled by the model initialization (init: one of QWEN-ST or Base QWEN3-4B) followed by the rewards used in phase I and II separated by an arrow. For example, the dashed green line is the QWEN3-4B base model trained on both ST + MR rewards in both phases (ST+MR $\rightarrow$ ST+MR). The gray line represents QWEN3-4B base model trained only on move-rating reward for Phase I.

Figure 5 also isolates reward interference among runs that all start from the QWEN3-4B base model. In Phase I, one run trains with ST reward only, one with the joint ST+MR objective, and one with MR reward only. Adding the MR reward slows ST optimization: the ST-only run reaches higher ST reward than the joint run. At the same time, the joint run achieves higher MR reward than the MR-only baseline. This asymmetry suggests that the ST reward changes the rollout distribution: by encouraging the model to maintain and update beliefs, it induces longer completions that also help move-rating optimization. Again, evidence in favor of longer rollouts leading to better rewards.

This reward-interference result matters because modern agentic RL commonly mixes multiple environments, reward functions, or task families in a single run, where progress on one objective can interfere with another. Figure 5 shows this interference phenomenon: among base-initialized runs, the joint ST+MR objective slows state-tracking progress relative to ST-only training. We therefore use Phase II to test whether cascading mitigates this interference. Starting from selected Phase I checkpoints, we continue training with the same joint ST+MR objective. The QWEN-ST-initialized continuation preserves the longest rollouts and reaches the highest ST reward, while staying competitive on MR reward. Thus, the benefit of QWEN-ST is not only faster convergence on a single reward; it also provides a better starting point for later joint optimization, where the harder state-tracking behavior would otherwise be partially suppressed. Staged hard-task training is therefore a

concrete recipe implication of our findings; validating this recipe across broader multi-environment and multi-task settings is a natural direction for future work.

4 Conclusion

Our results separate three outcomes that are often conflated in RL post-training. First, a model can improve in the scaffold used for training without acquiring competence that transfers across scaffolds. Failure mode in the small-model Hanabi experiments shows trained-scaffold scores improve substantially, while held-out WATSON remains weak (Section 3.1). Crucially, this is not a generic small-model failure: the same QWEN3-4B model improves strongly when it is trained with hard task RL (QWEN-ST), and the scaffold supplies Hanabi rules and state support, so the WATSON failure specifically identifies missing internal, scaffold-independent task competence/knowledge. Second, RL gains transfer more reliably when the base model already has task-relevant competence. The GPT-OSS-120B experiments show this contrast: unlike the smaller model, GPT-OSS-120B MYCROFT RL improves performance across held-out scaffolds (Section 3.2). Third, even when zero-shot scaffold transfer fails, the checkpoint can still transfer trainability. The hard-task (FIREWORKS) trained checkpoint is not better than the base model on the MYCROFT state-tracking probe, but it produces longer rollouts and converges faster during later MYCROFT RL (Section 3.4).

When training involves multiple tasks, rewards, or environments, our results favor staged training over mixing all objectives from a cold start. Joint state-tracking and move-rating optimization shows interference, while a hard structured checkpoint produces longer rollouts and makes later RL converge faster (Section 3.4). We discuss connections to cascade RL, multi-teacher distillation, and directions for future recipe comparisons in Appendix B.

The QWEN-ST results also show why scaffold-local RL can still be useful in realistic agent settings. Although QWEN-ST does not directly transfer FIREWORKS state tracking into the MYCROFT probe, it improves most on SHERLOCK (Section 3.3), where the scaffold supplies more state support and the model can focus on using that information for action selection (Table 2; Appendix D). This resembles deployed agent systems where external context, documents, tool outputs, or skill files can supply missing information. In such settings, RL-trained models may be valuable even when they have not internalized all task knowledge, because RL can improve persistence, state use, and action selection once the relevant information is present in context.

Overall, our results argue against reading RL gains as a single object. An RL-trained agent may acquire scaffold-local behavior, transferable task competence, or a better initialization for later training. Distinguishing these outcomes is necessary for evaluating whether post-training has produced deployable agent competence or only improved performance under one scaffold.

References

[1] AgentMarketCap. Scaffold over model: How the same ai scores 17 issues apart with different agent frameworks. https://agentmarketcap.ai/blog/2026/04/06/scaffold-over-model-agent-framework-swe-bench-scores, 2026.

[2] A. Ahmadian, C. Cremer, M. Gallé, M. Fadaee, J. Kreutzer, O. Pietquin, A. Üstün, and S. Hooker. Back to basics: Revisiting REINFORCE-style optimization for learning from human feedback in LLMs. In L.-W. Ku, A. Martins, and V. Srikumar, editors, *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 12248–12267, Bangkok, Thailand, Aug. 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.662. URL https://aclanthology.org/2024.acl-long.662/.

[3] Anthropic. Claude code. https://docs.anthropic.com/en/release-notes/claude-code, 2025. Release notes; Claude Code research preview announced February 24, 2025 and general availability announced May 22, 2025.

[4] Anthropic. Introducing claude opus 4.5. https://www.anthropic.com/news/claude-opus-4-5, Nov. 2025. Official Anthropic release for Claude Opus 4.5.

[5] Anysphere. Cursor documentation. https://docs.cursor.com/, 2025. Documentation for Cursor, an AI-powered code editor.

[6] Augment Code. Auggie tops swe-bench pro. https://www.augmentcode.com/blog/auggie-tops-swe-bench-pro, 2026.

[7] S. Bae, J. Hong, M. Y. Lee, H. Kim, J. Nam, and D. Kwak. Online difficulty filtering for reasoning oriented reinforcement learning. *arXiv preprint arXiv:2504.03380*, 2025. URL https://arxiv.org/pdf/2504.03380.

[8] H. Bai, Y. Sun, W. Hu, S. Qiu, M. Z. Huan, P. Song, R. Nowak, and D. Song. How and why llms generalize: A fine-grained analysis of llm reasoning from cognitive behaviors to low-level patterns, 2025. URL https://arxiv.org/abs/2512.24063.

[9] N. Bard, J. N. Foerster, S. Chandar, N. Burch, M. Lanctot, H. F. Song, E. Parisotto, V. Dumoulin, S. Moitra, E. Hughes, I. Dunning, S. Mourad, H. Larochelle, M. G. Bellemare, and M. Bowling. The hanabi challenge: A new frontier for AI research. *Artificial Intelligence*, 280:103216, 2020. doi: 10.1016/j.artint.2019.103216. URL https://dl.acm.org/doi/10.1016/j.artint.2019.103216.

[10] B. Brown, J. Juravsky, R. Ehrlich, R. Clark, Q. V. Le, C. Ré, and A. Mirhoseini. Large language monkeys: Scaling inference compute with repeated sampling, 2024. URL https://arxiv.org/abs/2407.21787.

[11] T. Chu, Y. Zhai, J. Yang, S. Tong, S. Xie, D. Schuurmans, Q. V. Le, S. Levine, and Y. Ma. SFT memorizes, RL generalizes: A comparative study of foundation model post-training. In A. Singh, M. Fazel, D. Hsu, S. Lacoste-Julien, F. Berkenkamp, T. Maharaj, K. Wagstaff, and J. Zhu, editors, *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pages 10818–10838. PMLR, 13–19 Jul 2025. URL https://proceedings.mlr.press/v267/chu25c.html.

[12] G. Comanici, E. Bieber, M. Schaekermann, I. Pasupat, N. Sachdeva, I. Dhillon, M. Blistein, O. Ram, D. Zhang, E. Rosen, et al. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. *arXiv preprint arXiv:2507.06261*, 2025. URL https://arxiv.org/abs/2507.06261.

[13] X. Deng, J. Da, E. Pan, Y. Y. He, C. Ide, K. Garg, N. Lauffer, A. Park, N. Pasari, C. Rane, K. Sampath, M. Krishnan, S. Kundurthy, S. Hendryx, Z. Wang, V. Bharadwaj, J. Holm, R. Aluri, C. B. C. Zhang, N. Jacobson, B. Liu, and B. Kenstler. Swe-bench pro: Can ai agents solve long-horizon software engineering tasks?, 2025. URL https://labs.scale.com/leaderboard/swe_bench_pro_public.

[14] Google DeepMind. The Hanabi Learning Environment. https://github.com/google-deepmind/hanabi-learning-environment, 2019.

[15] Q. Huang, J. Vora, P. Liang, and J. Leskovec. Benchmarking large language models as ai research agents. In *NeurIPS 2023 Foundation Models for Decision Making Workshop*, 2023. URL https://openreview.net/pdf?id=N9wD4RFWY0.

[16] G. Ilharco, M. T. Ribeiro, M. Wortsman, S. Gururangan, L. Schmidt, H. Hajishirzi, and A. Farhadi. Editing models with task arithmetic. In *International Conference on Learning Representations*, 2023. URL https://openreview.net/pdf?id=6t0Kwf8-jrj.

[17] P. Intellect. Prime-rl, 2025. URL https://github.com/PrimeIntellect-ai/prime-rl.

[18] Jack Nichols. How we scored #1 on terminal-bench. https://www.warp.dev/blog/terminal-bench, 2025.

[19] C. E. Jimenez, J. Yang, A. Wettig, S. Yao, K. Pei, O. Press, and K. R. Narasimhan. SWE-bench: Can language models resolve real-world github issues? In *International Conference on Learning Representations*, 2024. URL https://www.swebench.com/.

[20] S. Kapoor, B. Stroebl, Z. S. Siegel, N. Nadgir, and A. Narayanan. Ai agents that matter, 2024. URL https://arxiv.org/pdf/2407.01502.

[21] Kimi Team. Kimi k1.5: Scaling reinforcement learning with llms, 2025. URL https://arxiv.org/pdf/2501.12599.

[22] N. Lambert, J. Morrison, V. Pyatkin, S. Huang, H. Ivison, F. Brahman, L. J. V. Miranda, A. Liu, N. Dziri, X. Lyu, Y. Gu, S. Malik, V. Graf, J. D. Hwang, J. Yang, R. L. Bras, O. Tafjord, C. Wilhelm, L. Soldaini, N. A. Smith, Y. Wang, P. Dasigi, and H. Hajishirzi. Tulu 3: Pushing frontiers in open language model post-training. In *Second Conference on Language Modeling*, 2025. URL https://openreview.net/pdf?id=i1uGbfHHpH.

[23] M. Liu, S. Diao, X. Lu, J. Hu, X. Dong, Y. Choi, J. Kautz, and Y. Dong. Prorl: Prolonged reinforcement learning expands reasoning boundaries in large language models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025. URL https://openreview.net/pdf?id=YPsJha5HXQ.

[24] B. Lu, H. Zhao, S. Sun, H. Peng, R. Ding, and H. Mei. Generalization of RLVR using causal reasoning as a testbed. In *International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=DZjbL9BuHs.

[25] Y. Lu and M. Jiang. Uncovering cross-objective interference in multi-objective alignment, 2026. URL https://arxiv.org/pdf/2602.06869.

[26] M. Meireles, N. Lauffer, R. Bhati, and C. Allen. The influence of scaffolds on coordination scaling laws in llm agents. In *NeurIPS 2025 Workshop on Scaling Environments for Agents*, 2025. URL https://openreview.net/forum?id=E9whrbtgUA.

[27] M. A. Merrill, A. G. Shaw, N. Carlini, B. Li, H. Raj, I. Bercovich, L. Shi, J. Y. Shin, T. Walshe, E. K. Buchanan, et al. Terminal-bench: Benchmarking agents on hard, realistic tasks in command line interfaces. *arXiv preprint arXiv:2601.11868*, 2026. URL https://arxiv.org/pdf/2601.11868.

[28] F. Ndzomga. Efficient benchmarking of ai agents, 2026. URL https://arxiv.org/abs/2603.23749.

[29] OpenAI. Learning to reason with llms. https://openai.com/index/learning-to-reason-with-llms/, 2024.

[30] OpenAI. Introducing GPT-4.1 in the API. https://openai.com/index/gpt-4-1/, Apr. 2025. Official OpenAI release.

[31] OpenAI. Introducing openai o3 and o4-mini. https://openai.com/index/introducing-o3-and-o4-mini/, 2025. Accessed 2026-05-05.

[32] Prime Intellect Team. INTELLECT-3-RL. https://huggingface.co/datasets/PrimeIntellect/INTELLECT-3-RL, 2026. Hugging Face dataset.

[33] M. Ramesh, K. Jayakumar, A. Ramkumar, P. Thodima, A. Rege, and E.-V. Vlatakis-Gkaragkounis. Sparks of cooperative reasoning: Llms as strategic hanabi agents. In *Forty-third International Conference on Machine Learning*, 2026. URL https://kaousheik-26.github.io/llms-hanabi-cooperative-reasoning/.

[34] S. S. Ramesh, X. Ji, M. Zimmer, S. Yoon, Z. Wang, H. Bou Ammar, A. Lucchi, and I. Bogunovic. Multi-task grpo: Reliable llm reasoning across tasks, 2026. URL https://arxiv.org/abs/2602.05547.

[35] J. Rosser and J. N. Foerster. Agentbreeder: Mitigating the ai safety risks of multi-agent scaffolds via self-improvement. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025. URL https://openreview.net/forum?id=mlU9KqdZUS. Spotlight.

[36] M. Sclar, Y. Choi, Y. Tsvetkov, and A. Suhr. Quantifying language models' sensitivity to spurious features in prompt design or: How i learned to start worrying about prompt formatting. In *International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=RIu5lyNXjT.

[37] Sebastian Mondragon. Agent scaffolding beats model upgrades: 42% to 78% on swe-bench. https://particula.tech/blog/agent-scaffolding-beats-model-upgrades-swe-bench, 2026.

[38] Z. Shao, P. Wang, Q. Zhu, R. Xu, J. Song, X. Bi, H. Zhang, M. Zhang, Y. K. Li, Y. Wu, and D. Guo. Deepseekmath: Pushing the limits of mathematical reasoning in open language models, 2024. URL https://arxiv.org/abs/2402.03300.

[39] Z. S. Siegel, S. Kapoor, N. Nadgir, B. Stroebl, and A. Narayanan. CORE-bench: Fostering the credibility of published research through a computational reproducibility agent benchmark. *Transactions on Machine Learning Research*, 2024. ISSN 2835-8856. URL https://openreview.net/forum?id=BsMMc4MEGS.

[40] C. Snell, J. Lee, K. Xu, and A. Kumar. Scaling llm test-time compute optimally can be more effective than scaling model parameters, 2024. URL https://arxiv.org/abs/2408.03314.

[41] H. Sun, S. Zhang, L. Niu, L. Ren, H. Xu, H. Fu, F. Zhao, C. Yuan, and X. Wang. Collab-overcooked: Benchmarking and evaluating large language models as collaborative agents. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, page 4922–4951. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.emnlp-main.249. URL http://dx.doi.org/10.18653/v1/2025.emnlp-main.249.

[42] D. team. Deepseek-r1 incentivizes reasoning in llms through reinforcement learning. *Nature*, 645(8081):633–638, Sept. 2025. doi: 10.1038/s41586-025-09422-z. URL https://www.nature.com/articles/s41586-025-09422-z.

[43] Thinking Machines Lab. Tinker documentation. https://tinker-docs.thinkingmachines.ai/, 2025. RL training and hyperparameter documentation.

[44] X. Wang, Y. Chen, L. Yuan, Y. Zhang, Y. Li, H. Peng, and H. Ji. Executable code actions elicit better LLM agents. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 50208–50232. PMLR, 2024. URL https://proceedings.mlr.press/v235/wang24h.html.

[45] X. Wang, B. Li, Y. Song, F. F. Xu, X. Tang, M. Zhuge, J. Pan, Y. Song, B. Li, J. Singh, H. H. Tran, F. Li, R. Ma, M. Zheng, B. Qian, Y. Shao, N. Muennighoff, Y. Zhang, B. Hui, J. Lin, R. Brennan, H. Peng, H. Ji, and G. Neubig. OpenHands: An open platform for AI software developers as generalist agents. In *International Conference on Learning Representations*, 2025. URL https://openreview.net/pdf?id=OJd3ayDDoF.

[46] X. Wen, Z. Liu, S. Zheng, S. Ye, Z. Wu, Y. Wang, Z. Xu, X. Liang, J. Li, Z. Miao, J. Bian, and M. Yang. Reinforcement learning with verifiable rewards implicitly incentivizes correct reasoning in base llms. In *International Conference on Learning Representations*, 2026. URL https://arxiv.org/pdf/2506.14245.

[47] R. J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8(3–4):229–256, 1992. doi: 10.1007/BF00992696. URL https://doi.org/10.1007/BF00992696.

[48] M. Wortsman, G. Ilharco, S. Y. Gadre, R. Roelofs, R. Gontijo-Lopes, A. S. Morcos, H. Namkoong, A. Farhadi, Y. Carmon, S. Kornblith, and L. Schmidt. Model soups: Averaging weights of multiple fine-tuned models improves accuracy without increasing inference time. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 23965–23998. PMLR, 2022. URL https://proceedings.mlr.press/v162/wortsman22a.html.

[49] xAI. Grok 3 beta: The age of reasoning agents. https://x.ai/news/grok-3, Feb. 2025. Official xAI release introducing Grok 3 and Grok 3 mini.

[50] xAI. Grok 4.1 fast and agent tools api. https://x.ai/news/grok-4-1-fast, Nov. 2025. Official xAI release describing Grok 4.1 Fast and the grok-4-1-fast-reasoning API variant.

[51] Xiaomi LLM-Core Team. Mimo-v2-flash technical report, 2026. URL https://arxiv.org/abs/2601.02780.

[52] P. Yadav, D. Tam, L. Choshen, C. Raffel, and M. Bansal. TIES-merging: Resolving interference when merging models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023. URL https://openreview.net/forum?id=xtaX3WyCj1.

[53] J. Yang, C. E. Jimenez, A. Wettig, K. Lieret, S. Yao, K. Narasimhan, and O. Press. SWE-agent: Agent-computer interfaces enable automated software engineering. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024. URL https://openreview.net/forum?id=mXpq6ut8J3.

[54] Z. Yang, Z. Liu, Y. Chen, W. Dai, B. Wang, S.-C. Lin, C. Lee, Y. Chen, D. Jiang, J. He, R. Pi, G. Lam, N. Lee, A. Bukharin, M. Shoeybi, B. Catanzaro, and W. Ping. Nemotron-cascade 2: Post-training llms with cascade rl and multi-domain on-policy distillation, 2026. URL https://research.nvidia.com/labs/nemotron/nemotron-cascade.

[55] Q. Yu, Z. Zhang, R. Zhu, Y. Yuan, X. Zuo, Y. Yue, W. Dai, T. Fan, G. Liu, L. Liu, et al. Dapo: An open-source llm reinforcement learning system at scale. *arXiv preprint arXiv:2503.14476*, 2025. URL https://arxiv.org/pdf/2503.14476.

[56] Y. Yue, Z. Chen, R. Lu, A. Zhao, Z. Wang, S. Song, and G. Huang. Does reinforcement learning really incentivize reasoning capacity in llms beyond the base model?, 2025. URL https://arxiv.org/abs/2504.13837.

[57] W. Zeng, Y. Huang, Q. Liu, W. Liu, K. He, Z. Ma, and J. He. Simplerl-zoo: Investigating and taming zero reinforcement learning for open base models in the wild. In *Second Conference on Language Modeling (COLM)*, 2025. URL https://openreview.net/forum?id=vSMCBUgrQj.

[58] G. Zhang, H. Geng, X. Yu, Z. Yin, Z. Zhang, Z. Tan, H. Zhou, Z.-Z. Li, X. Xue, Y. Li, Y. Zhou, Y. Chen, C. Zhang, Y. Fan, Z. Wang, S. Huang, F. Piedrahita Velez, Y. Liao, H. Wang, M. Yang, H. Ji, J. Wang, S. Yan, P. Torr, and L. Bai. The landscape of agentic reinforcement learning for LLMs: A survey. *Transactions on Machine Learning Research*, 2026. ISSN 2835-8856. URL https://openreview.net/forum?id=RY19y2RI1O. Survey Certification.

Appendix Roadmap

The appendix is organized into three groups: extended discussion, experimental details, and reference material.

Extended discussion. Appendix A situates the paper within related work on RLVR generalization, scaffold sensitivity, inference-time computation, and cascaded RL. Appendix B discusses implications and limitations, connects our findings to broader debates about whether RLVR expands or merely reshapes model competence, and outlines directions for future work including comparisons between staged RL, OPD/MOPD consolidation, and SFT-then-RL recipes.

Experimental details. Appendix D provides the full cell-level Hanabi score table with 95% bootstrap confidence intervals for all model checkpoints, scaffolds, and player counts. Appendix E characterizes the difficulty of the FIREWORKS state-transition task for the QWEN3-4B base model via pass@k estimates. Appendix F analyzes why the state-tracking skill learned in FIREWORKS fails to transfer zero-shot to MYCROFT, using a diagnostic breakdown of preservation versus stale-copy errors. Appendix G.1 demonstrates the improved search efficiency of QWEN-HANABI+ over QWEN3-4B. Appendix G.2 reports the PI-LOGIC control experiment and explains why PI-LOGIC RL does not produce the same sustained increase in generation length observed under Hanabi state-tracking RL. Appendix H describes each scaffold and dataset in detail, including the FIREWORKS data curation recipe and the offline gameplay reward labeling procedure. Appendix I provides full training recipes, including dataset sizes, batch structures, reward functions, and the software stacks used for each experimental line. Appendix J describes the evaluation setup, including the held-out seed protocol, the MYCROFT state-tracking probe, and the PI-LOGIC evaluation criteria.

Reference material. Appendix K provides the complete prompt templates for all four interfaces: WATSON K.1, SHERLOCK K.2, MYCROFT K.3, and FIREWORKS K.4, each accompanied by a representative model response. These are included to make the scaffold differences concrete and to support reproducibility.

A Related Work

RL with verifiable rewards has become a central post-training recipe for improving reasoning language models. Open post-training systems such as Tulu 3 use verifiable rewards as part of a broader SFT, preference optimization, and RL pipeline [22], while DeepSeekMath [38], DeepSeek-R1 [42], and Kimi k1.5 [21] show strong gains on math, code, and reasoning benchmarks from optimizing verifiable rewards. These works establish that RLVR can substantially improve benchmark accuracy and often changes the form or length of model reasoning. Our question is not whether RLVR can improve a trained benchmark. We ask what part of that improvement survives when the same underlying agentic task is evaluated through a different scaffold.

Recent work has begun to study why RLVR works and when it generalizes. Yue et al. [56] argue that RLVR can improve small-k performance by shifting probability mass toward successful reasoning paths already present in the base model, while Wen et al. [46] argue that RLVR can incentivize correct reasoning rather than only improve final-answer sampling using more refined CoT-Pass@K evaluations. Closest to our competence-gating claim, Lu et al. [24] study RLVR generalization in causal graphical-model reasoning and find that RLVR benefits depend on model scale, training query level, and the model's initial reasoning competence. Our setting is different: rather than asking whether RLVR transfers across causal query levels or task difficulties, we ask whether RL gains on the same underlying agentic task survive changes in scaffold, observation format, state exposure, and reasoning instructions. This distinction matters because trained-scaffold evaluation can be deceptive. A model can improve strongly under the scaffold used for RL training while failing when the same task is presented through a different observation, reasoning, or action interface.

Our work is also related to studies of prompt and scaffold sensitivity. Language-model performance can change substantially under semantically equivalent prompt formats [36]. In agentic settings, the scaffold is even more central because it determines the observations, available tools, action interface, context management, and recovery behavior. SWE-bench evaluates software agents on real GitHub issues, and SWE-agent shows that the agent-computer interface itself can strongly affect benchmark performance [19, 53]. For example, CORE-Bench [39] reports large differences when Claude Opus 4.5 [4] is evaluated through different agent scaffolds, and SWE-bench Pro results

with Claude Opus 4.5 vary across Auggie [6] , Cursor [5], Claude Code [3], and SWE-Agent-style scaffolds . These results motivate evaluating not only model performance inside a fixed harness, but whether improvements survive scaffold changes. Recent work on Collab-Overcooked [41] similarly shows that scaffold design can determine whether coordination scaling trends appear at all: explicit role scaffolds produce clearer self-play and cross-play scaling, while less prescriptive scaffolds break these trends [26]. These works study scaffold sensitivity at evaluation time. We extend this concern to RL post-training: rather than only asking whether prompts or harnesses affect evaluation, we train under one scaffold and test whether the RLVR gain transfers to held-out scaffolds of the same environment.

A separate line of work studies inference-time computation. Scaling test-time compute, repeated sampling, and verifier-guided selection can improve reasoning and coding performance [10, 40]. We study a complementary effect: hard RLVR can change the model's rollout behavior itself. In our experiments, a state-tracking-trained checkpoint does not directly transfer its state-tracking skill to the MYCROFT scaffold, but it produces longer rollouts, improves held-out hard logic performance, and converges faster under later Hanabi RL. Thus, the checkpoint can transfer trainability even when it does not transfer zero-shot scaffold competence.

Finally, our sequential-training result connects to recent interest in cascaded RL, multi-task RLVR, and multi-objective interference. Nemotron-Cascade argues for training domains sequentially rather than mixing heterogeneous RL domains from the start, partly because domains differ in length, verifier cost, and optimization dynamics [54]. Recent multi-task RLVR work similarly observes that naive multi-task optimization can produce imbalanced progress across tasks, and multi-objective alignment work identifies cross-objective interference as a general failure mode [34, 25]. Our experiments give a controlled agentic instance of this issue. When state-tracking and move-rating rewards are optimized together from the base model, one objective slows the other. Training first on a hard structured task produces longer rollouts and makes later RL converge faster. This supports staged RL as a practical recipe for multi-task or multi-environment agent training when parallel reward optimization creates interference.

B Discussion and Future Work

Our results sit between two current views of RLVR. One view is that RLVR mainly shifts probability mass toward reasoning paths already available in the base model [56]. Another view is that prolonged RL can expand the model's reasoning boundary when training is long, diverse, and stable [23]. Our results suggest that the answer depends on what kind of transfer is being measured. The paper distinguishes several transfer notions, and each behaves differently. For direct scaffold transfer, our evidence is closer to the competence-boundary view: QWEN3-4B improves strongly inside structured Hanabi scaffolds but does not acquire scaffold-independent Hanabi competence, while GPT-OSS-120B transfers more consistently once it already has stronger pre-RL Hanabi competence. However, this does not mean RL produces no transferable benefit. Computation-style and trainability can transfer even in the weaker model: the QWEN-ST checkpoint transfers as inference-time behavior and downstream trainability, even when the state-tracking skill itself does not transfer zero-shot across scaffolds.

This distinction matters for RL-first training recipes. Recent zero-RL work shows that rule-based RL can elicit long reasoning behavior directly from base models [42, 57], and other work argues that RL can generalize better than SFT in some post-training settings [11]. Our small-model Hanabi result adds a caution: after roughly 23K Hanabi RL samples, QWEN3-4B learns to perform well under the trained scaffolds but remains weak under the least-scaffolded interface. This suggests that, when the bottleneck is missing task knowledge, missing state abstractions, or missing scaffold-independent competence, narrow RLVR may be an inefficient way to create that competence from scratch. In such cases, adding the missing competence first through SFT, OPD, distillation, synthetic data, or continued pretraining, followed by RL, may be more efficient than relying on RL alone.

At the same time, our results should not be read as an anti-RL conclusion. The FIREWORKS checkpoint is slightly worse than the base model on the MYCROFT state-tracking probe, so strict state-transition skill does not directly transfer into the richer gameplay scaffold. But the same checkpoint improves hard held-out logic performance and makes later MYCROFT state tracking RL converge

faster. This separates direct skill transfer from trainability transfer: a checkpoint can fail as an immediate policy under a new scaffold while still being valuable as an optimization state.

Appendix G.1 sharpens this interpretation. The hard-logic gains are not explained by the model merely spending more tokens. On held-out PI-LOGIC examples, the RL-trained checkpoints more often convert mid-length rollouts into correct answers, while the base model frequently enters degenerate long failures, such as repeated invalid candidates, cycling final-answer markers, or runaway generations that still receive reward 0. In this sense, hard-task RL appears to improve the quality of search, not only the amount of test-time computation.

The inference-time computation results help explain this distinction, but they should be interpreted carefully. Across the GPT-OSS-120B phases and the QWEN3-4B state-tracking line, stronger transfer or faster later convergence is associated with longer rollouts. This is consistent with RL changing how the model searches during generation, as also observed in RLVR systems that induce longer reasoning traces [42, 21, 57]. We therefore treat length as a diagnostic, not as causal proof. A model can generate more tokens without reasoning better; the relevant question is whether those tokens support productive search rather than unproductive verbosity. Future work should use length-matched decoding, fixed-budget evaluations, and explicit verbosity controls to separate useful deliberation from degenerate long-form failure.

Our results also connect to recent large-scale recipes that avoid naive joint RL over heterogeneous domains. Cascade RL trains domains sequentially rather than mixing them from the start, and Multi-Teacher On-Policy Distillation consolidates expertise from domain-specialized teachers into a single student model [54, 51]. Our contribution is complementary: in a controlled agentic setting, we show one mechanism that can make staging useful. Joint optimization of state-tracking and move-rating rewards slows one objective relative to training it alone, consistent with multi-task and multi-objective interference observed in recent RLVR and alignment work [34, 25]. A hard-task checkpoint then produces longer rollouts and faster subsequent RL convergence. This suggests that staged RL can be more than an engineering convenience for heterogeneous data: it can improve downstream trainability by reducing optimization interference.

This also separates our result from online policy distillation (OPD)/ Multi-teacher online policy distillation (MOPD)-style consolidation. Those methods transfer behavior from stronger or domain-specialized teachers into a student through on-policy teacher feedback, whereas our experiments ask what RL alone changes in the learner when the scaffold changes. The two approaches are likely complementary: our results suggest that SFT, OPD, MOPD, distillation, or continued pretraining may be better suited for adding missing task competence, while RL is more effective once the model already has enough competence to exploit or when the goal is to improve persistence, inference-time computation, and downstream trainability.

Future work should compare these recipes under the same scaffold-transfer protocol: naive parallel multi-environment RL, staged/cascade RL, OPD/MOPD consolidation from domain specialists, and mixtures of SFT followed by RL. This will help us identify the trade-off between the methods. We have noticed that cascading leads to improved usage of test-time compute and better convergence in subsequent training. Comparing this with MOPD will help us identify the best methodology. A related direction is model merging: if checkpoints trained on different scaffolds specialize in different behaviors, merging scaffold-specialized models before continued RL may recover complementary gains while exposing interference between them [48, 16, 52].

C Limitations

There are a few limitations to our evidence. First, Hanabi is a controlled agentic environment, but it is still one environment; the same scaffold-transfer protocol should be tested in software, terminal, web, and tool-use agents. Second, we study two model families and a limited set of scales, so the boundary at which scaffold transfer appears remains unresolved. More importantly, our scaffolds differ along several axes simultaneously (supplied game rules, candidate-move enumeration, engine-computed state deductions, and state formatting), and the current experiments do not isolate which component supplies the missing competence that gates transfer. Future ablations should vary these axes independently. Third, our RL data is fixed offline data. Online RL may behave differently because the model can change the state distribution it visits during training. Fourth, our move-rating reward uses strong LLM judges, so although the state-tracking reward is verifiable, part of the

gameplay signal inherits the limitations of judge-based reward labels. To be precise: the scaffold-transfer conclusions (Sections 3.1–3.2) and the trainability-transfer results (Section 3.4) rest on held-out Hanabi game scores and the deterministic state-tracking reward, not on the judge-based move-rating signal. The move-rating reward is used to train and evaluate action-selection quality within gameplay; its judge-based nature does not affect the paper's main transfer claims.

D Full Hanabi score table

Table 3 reports 95% confidence interval scores of all the checkpoints discussed in this paper along with the mean scores over 10 held-out seeds across player settings and across all 3 scaffolds. QWEN3-4B base model shows poor performance on all player counts and across scaffolds, confirming low Hanabi competence. After training QWEN3-4B on 10K strict state-tracking FIREWORKS data, the resulting QWEN-ST shows notable improvement, particularly on SHERLOCK, where 3P rises from 2.33 to 10.67, consistent with the interpretation that RL-induced persistence helps when the scaffold supplies the missing Hanabi knowledge and state support. Further training QWEN-ST on SHERLOCK and MYCROFT data produces QWEN-HANABI, which shows a drastic increase in performance on the trained scaffolds but remains close to baseline on the held-out WATSON. This suggests that prolonged scaffolded RL does not, by itself, create the scaffold-independent Hanabi knowledge required by WATSON.

Since Hanabi belief states in earlier turns propagate to later ones, we further train QWEN-HANABI on the first 30 turns of 120 MYCROFT games, expecting better performance resulting in QWEN-HANABI+, which shows a further increase as expected in MYCROFT performance but a decline in SHERLOCK. This decline is concentrated in 2P (11.00 $\rightarrow$ 5.33) and 3P (14.67 $\rightarrow$ 11.83), while WATSON remains close to baseline throughout. Highlights that the transfer across scaffolds is brittle.

Compared to QWEN3-4B, the GPT-OSS-120B base model shows strong performance across all three scaffolds (WATSON 13.12, SHERLOCK 14.42, MYCROFT 12.54). In Phase I, we train on 40 MYCROFT games; evaluation scores change only marginally, indicating that reward improvement alone does not predict transfer. In Phase II, we train on the first 30 turns of 120 MYCROFT games, producing consistent improvements across all scaffolds (WATSON 13.83, SHERLOCK 15.62, MYCROFT 14.00). This is notable because the same early-turn data distribution degraded SHERLOCK in the QWEN3-4B line (13.29 $\rightarrow$ 11.00), suggesting that QWEN3-4B performance drop is due to low pre-RL competence of the model rather than the data. Phase III continues on 80 additional full games but performance regresses, suggesting that training on data containing easier samples degrades the performance. In Phase IV, we train only on failure cases (move-rating reward below 0.2), producing the strongest checkpoint across all scaffolds (Watson 13.96, Sherlock 16.04, Mycroft 14.50). This checkpoint matches or exceeds O4-MINI on SHERLOCK and MYCROFT across all player counts, with large margins on MYCROFT (e.g., 15.33 vs. 11.20 at 2P). On WATSON, Phase IV remains slightly below O4-MINI at 2P and 3P but closes the gap at 4P and 5P.

Overall, Figure 3 suggests an association between average rollout completion length at inference time and Hanabi evaluation scores across all 3 scaffolds.

Table 3: Hanabi scores across 2–5 player counts and scaffolds. Values are IQM over 10 held-out seeds with 95% bootstrap confidence intervals, reported as IQM $\pm$ CI radius.

	WATSON				SHERLOCK				MYCROFT			
Model	2P	3P	4P	5P	2P	3P	4P	5P	2P	3P	4P	5P
O4-MINI†	15.00±1.00	15.30±0.80	14.30±0.80	13.30±0.80	16.80±2.00	16.70±1.00	13.80±0.90	12.80±1.00	11.20±2.50	11.80±1.10	11.20±1.00	10.50±1.50
QWEN3-4B Base	1.67±1.00	1.67±0.83	1.50±0.50	0.67±0.50	1.33±1.17	2.33±1.33	5.00±1.00	3.17±1.66	0.67±0.83	0.50±1.00	1.67±1.16	0.67±0.50
QWEN3-4B PI-Logic RL	1.83±1.00	1.67±1.00	2.83±0.84	1.33±1.17	2.00±1.83	3.00±1.83	2.67±1.83	3.67±1.67	0.67±1.33	0.17±1.16	2.33±2.17	1.50±0.83
QWEN-ST	1.50±1.17	2.67±0.83	2.33±0.83	2.00±0.50	3.67±3.00	10.67±1.67	7.67±1.17	7.00±2.00	2.50±1.83	3.50±1.00	3.17±2.00	2.17±0.84
QWEN-HANABI	2.00±0.83	2.00±1.00	2.50±0.67	2.33±0.66	11.00±5.50	14.67±1.00	13.67±2.67	13.83±0.83	12.33±3.00	12.83±2.83	13.50±1.33	12.67±1.00
QWEN-HANABI+	2.67±0.83	2.33±1.00	2.50±0.50	2.33±0.84	5.33±3.17	11.83±4.00	14.00±3.17	12.83±3.00	12.17±3.34	13.67±2.50	14.33±2.33	13.33±1.00
GPT-OSS-120B	13.67±1.67	13.33±0.66	13.33±1.50	12.17±1.00	14.83±2.17	14.50±0.83	15.17±1.84	13.17±1.00	13.50±1.83	12.67±1.17	12.83±1.16	11.17±1.66
GPT-OSS-120B Phase I	13.83±1.67	13.33±1.34	13.33±1.33	12.33±1.66	15.33±1.50	14.67±1.17	14.50±1.17	12.83±1.17	14.00±1.50	12.83±1.00	11.83±1.66	12.00±0.67
GPT-OSS-120B Phase II	14.83±1.50	13.50±1.33	14.00±1.17	13.00±1.17	15.50±1.33	16.50±1.33	15.67±0.84	14.83±1.34	14.67±2.67	14.33±1.00	13.83±1.34	13.17±1.34
GPT-OSS-120B Phase III	13.17±1.33	13.33±1.00	14.00±0.50	12.83±1.00	16.67±3.17	15.33±1.16	15.33±1.33	13.83±0.67	13.50±1.83	13.17±1.34	13.33±0.67	12.33±0.50
GPT-OSS-120B Phase IV	14.00±2.50	14.00±1.33	14.17±1.00	13.67±1.00	16.83±2.33	16.17±1.00	16.17±1.16	15.00±1.33	15.33±1.84	14.67±1.33	14.33±0.84	13.67±0.50

†The O4-MINI row is to show reference frontier model performance in Hanabi scaffolds.

E Hard state-transition validation

We use a held-out 5-player FIREWORKS validation set with 61 hard strict state-transition prompts. Each prompt requires exact reconstruction of the updated hidden-belief state, and a completion is counted as correct only if every card slot matches the oracle state.

For the QWEN3-4B base model, we sample 1024 completions per prompt. The model finds at least one correct completion for only 10 of 61 prompts: 51 prompts are never solved, 5 prompts are solved once, 4 prompts are solved three times, and 1 prompt is solved seven times. Table 4 reports the resulting pass@k estimates. The near-zero pass@1 and low pass@1024 show that the strict FIREWORKS state-transition task is difficult for the base model.

Table 4: Base QWEN3-4B pass@k on the held-out hard 5-player strict state-transition (FIREWORKS) validation set. All values are calculated over 61 samples.

Metric	Percent
pass@1	0.0384%
pass@16	0.6026%
pass@64	2.2649%
pass@256	7.2649%
pass@1024	16.3934%

After strict state-transition RL, QWEN-ST solves 57 of the 61 prompts at pass@1, showing that the FIREWORKS training task itself was learned under the strict exact-match reward.

F State tracking transfer failure:

To probe further why the state tracking from FIREWORKS didn't transfer to MYCROFT. We analyzed the outputs of QWEN-ST and QWEN3-4B on the 500 sample 5P evaluation mycroft data. The diagnostic in Table 5 explains the failure mode. Conditional on parseable outputs, QWEN-ST is much better at preserving cards that should remain unchanged. But it also stale-copies cards that should change more than twice as often as the base model. This suggests partial transfer rather than no transfer: FIREWORKS RL teaches a strong state-preservation prior, but under the richer MYCROFT gameplay prompt this prior is not reliably coupled to detecting when intervening actions should update a card slot. As a result, QWEN-ST preserves stable beliefs better but hurts changed-card accuracy.

Table 5: State-tracking diagnostic on 500 MYCROFT 5P validation samples. QWEN-ST learned to preserve state, but the same prior causes stale-copy errors when cards should change.

Model	Preserve unchanged	Stale-copy changed	Unchanged acc.	Changed acc.
QWEN3-4B base	0.3170	0.1985	0.3170	0.1175
QWEN-ST	**0.5331**	0.4241	**0.5331**	0.1052

G PI-LOGIC

G.1 Search Efficiency on Held-Out PI-LOGIC

Section 3.3 reports PI-LOGIC gains after hard-task RL. We use completion length only as a **diagnostic** for changed inference-time computation, not as a causal explanation. The analysis below uses the same 100-example held-out set (8 samples each) and three checkpoints along the QWEN3-4B RL trajectory: base, QWEN-ST, and QWEN-HANABI+.

RL shifts probability mass toward correct completions (Pass@8: 0.15 $\rightarrow$ 0.37 $\rightarrow$ 0.61) and Gains appear on both Mathador (Pass@8: 0.62 $\rightarrow$ 0.69 $\rightarrow$ 0.77) and zebra puzzles (0.08 $\rightarrow$ 0.32 $\rightarrow$ 0.59), with the largest incremental lift on harder 5–6 house grids.

Long rollouts are not uniformly productive. Base-model failures often enter degenerate loops: repeated invalid candidates, cycling "Final Answer" markers, or runaway generation to ∼32k tokens, while still receiving reward 0. Median length on these concession trajectories is 32k tokens for the base model vs. 2.3k on its successful rollouts; QWEN-HANABI+ produces fewer such rollouts (23 vs. 52 out of 800) yet achieves longer *successful* completions when needed (median 3.9k). RL therefore sustains productive deliberation without collapsing into the base model's failure regime.

The largest QWEN-HANABI+ uplift occurs at 2k–8k tokens (e.g. 0.05 → 0.36 in the 2k–4k bin), where Pass@8 nonetheless rises 4× while median correct length grows from 2.3k → 3.9k. On Mathador, rollout recovery (from degenerated loops) rises from 0/56 (base) to 11/55 (QWEN-HANABI+). The results suggest that the tokens' length is a good diagnostic, but the effectiveness of RL is not only due to the test time deliberation. Hard-task RL improves held-out PI-LOGIC by changing search behavior: trained checkpoints reach correct answers earlier, avoid attractor collapse, convert mid-length rollouts more effectively, and repair near-solutions in context.

Examples: Mathador ($[3, 5, 6, 8, 9] \rightarrow 29$): base fails after 32,683 tokens (cycles on invalid `(5*6)-(9-8)`); QWEN-HANABI+ succeeds in 4,184 tokens with `(9+3)*(8-6)+5` (**7.8**× shorter). **Zebra** (5 houses): base fails after 32,259 tokens (87 "Final Answer" cycles, empty grid); QWEN-HANABI+ succeeds in 6,767 tokens (**4.8**× shorter).

G.2 PI-Logic Training

We trained a QWEN3-4B checkpoint on PI-LOGIC RL (10K samples) alone and evaluated it on Hanabi. The Hanabi scores are included in Table 6. PI-LOGIC RL averages 3.2 on SHERLOCK and 1.53 on MYCROFT, close to the weak QWEN3-4B base regime. We also notice the training does not reproduce the same sustained length increase observed in Hanabi-derived hard structured RL. The training length curve is shown in Figure 6. This also demonstrates that not any RL training on hard tasks elicits increased inference-time computation in the models. We believe that homogeneous environments repeatedly reinforcing state maintenance and belief updates are faster to induce useful computation tendencies than heterogeneous logic prompts whose surface form varies across samples.

Table 6: PI-LOGIC RL does not transfer to Hanabi. Values are means over 2P-5P IQMs; each cell-level IQM uses 10 games evenly across player counts

Model	**WATSON**	**SHERLOCK**	**MYCROFT**
QWEN3-4B Base	1.38	2.96	0.88
QWEN3-4B PI-LOGIC RL	1.92	3.2	1.53

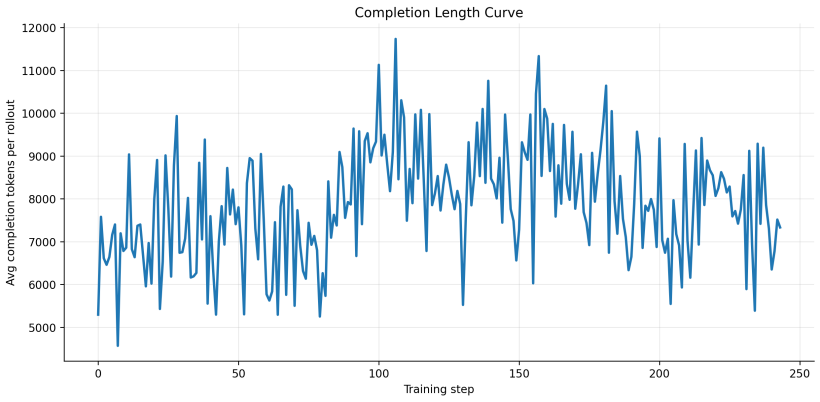

Figure 6: Training-rollout length for the PI-LOGIC-only Qwen control. Unlike Hanabi state tracking, this run does not show a comparable sustained increase in generation length, which is why we do not interpret PI-LOGIC RL as an equivalent hard-structured-RL control.

H Detailed scaffold and dataset descriptions

WATSON. WATSON serves as the least-scaffolded evaluation interface. It presents the game state with minimal support: the model receives visible hands and clue-derived knowledge but no Hanabi rules and no engine-computed deductions. Strong WATSON performance therefore indicates that the model has enough usable Hanabi knowledge to act without explicit guidance. In our experiments, WATSON is not treated as inherently harder or more important than the other scaffolds; rather, because it provides the least explicit game knowledge and deductive support, it tests whether RL-trained behavior survives when the model must supply more of the Hanabi competence itself. The full WATSON prompt is provided in Appendix K.1.

SHERLOCK. SHERLOCK supplies the richest context among the three scaffolds: the model receives the full game state, candidate legal moves, Hanabi rules, and engine-computed deductive information, and must select an action. Because the state is already provided, SHERLOCK does not require or train the model to reconstruct beliefs, and its reward signal is move rating only. This separation makes SHERLOCK useful for isolating action-selection ability from state-tracking ability. The full SHERLOCK prompt is provided in Appendix K.2.

MYCROFT. MYCROFT requires the model to reconstruct the current belief state from the previous turn's state and all intervening actions, reason about what each teammate knows (theory of mind), and then select a move. It is the primary scaffold for reward decomposition because it exposes two partially competing training signals: a verifiable state-tracking reward that scores belief-state accuracy and a judged move-rating reward that scores action selection. We modified the MYCROFT from [33] to include theory of mind in the prompt. Throughout the paper, MYCROFT refers to this theory-of-mind variant. The full MYCROFT prompt is provided in Appendix K.3.

FIREWORKS. FIREWORKS is a standalone state-transition reconstruction task, distinct from the three gameplay scaffolds. To build the dataset, we let frontier models play Hanabi across 2–5 player settings and collect 10K trajectory states. We use GPT-4.1, Grok-3-mini, and Gemini-2.5-Pro for trajectory generation [30, 49, 12], and obtain oracle belief-state deductions programmatically from the Hanabi Learning Environment [14]. Each prompt presents the previous belief state and the chronological list of intervening actions, and the model must produce the exact updated belief state with no move selection involved. The FIREWORKS prompt format is provided in Appendix K.4.

Gameplay reward data. Gameplay training uses fixed offline game samples. For MYCROFT we use 18k full-game samples; for SHERLOCK we use 4k samples. Candidate legal actions are labeled in advance by a strong reasoning-model judge [31, 50, 49]. During RL, only the chosen action's stored label becomes the move reward. This is important because the policy is not interacting with a live Hanabi environment during training; it is optimizing over a fixed prompt/reward corpus. We made sure the training data doesn't contain seeds 1001-1010 trajectories.

We will open-source all our data and code upon acceptance.

I Detailed training recipes

I.1 Training data configuration

All runs use fixed offline datasets with precomputed rewards. We use Prime-RL [17] for the small-model gameplay runs in Section 3.1 (QWEN-HANABI and QWEN-HANABI+), and Tinker [43, 47, 2] for the remaining RL experiments. For both stacks, we report the effective rollout batch as completions per update. For QWEN3-4B runs, 512 completions/update means 32 unique prompts with 16 rollouts per prompt; for GPT-OSS-120B runs, it means 64 unique prompts with 8 rollouts per prompt. Maximum generation length is 16,384 tokens for all RL training rollouts. Unless otherwise stated, all other hyperparameters use the corresponding training-stack defaults. For both QWEN3-4B and GPT-OSS-120B maximum generation length is 16,384 tokens. Length curves in the figures are generated from training rollouts.

Release plan. Upon acceptance, we will publicly release the following assets: the full FIREWORKS dataset (10K training + 61 validation samples), the 18K MYCROFT-scaffold move-rating labels, all scaffold prompt templates (WATSON, SHERLOCK, MYCROFT, and FIREWORKS), the held-out evaluation seeds (1001–1010), evaluation and training scripts, and the QWEN-ST, QWEN-HANABI,

Table 7: Training recipes used in the paper. "Completions/update" is the effective rollout batch; unique prompts per update and rollouts per prompt are shown separately. ST here means State Tracking reward in equation 2

Line	Stack	Init.	Data	Rewards	Batch structure	Purpose
QWEN-ST	Tinker	QWEN3-4B	10K FIREWORKS samples	strict exact state	512 completions/update; 32 prompts × 16 rollouts	Isolate hard state tracking and obtain QWEN-ST
QWEN-HANABI	Prime-RL	QWEN-ST	18K MYCROFT + 4K SHERLOCK	MYCROFT: ST+move; SHERLOCK: move	512 completions/update; 32 prompts × 16 rollouts	Gameplay RL on trained scaffolds; test transfer to held-out WATSON
QWEN-HANABI+	Prime-RL	QWEN-HANABI	first 30 turns from 120 MYCROFT games	ST+move	512 completions/update; 32 prompts × 16 rollouts	Continued MYCROFT training; test whether one scaffold improves while another degrades
GPT-OSS-120B phases	Tinker	GPT-OSS-120B	MYCROFT phase slices	ST+move	512 completions/update; 64 prompts × 8 rollouts	Test competence-gated cross-scaffold transfer
Interference runs	Tinker	QWEN3-4B or QWEN-ST	40 games / 2.4K MYCROFT samples	ST only, move only, or both	512 completions/update; 32 prompts × 16 rollouts	Isolate reward interference and trainability transfer
PI-LOGIC control	Tinker	QWEN3-4B	PI-LOGIC training split	answer match	512 completions/update; 32 prompts × 16 rollouts	Test whether unrelated hard-task RL transfers to Hanabi

and QWEN-HANABI+ model checkpoints. These will be hosted on Hugging Face under a permissive open-source license.

I.2 Rewards used

Table 8: Reward functions used across datasets. Some Hanabi runs optimize a single reward, while others jointly optimize state-tracking and move-quality rewards.

Dataset	Reward type	Definition	Used in
FIREWORKS	Strict state-tracking	$R = \mathbf{1}[\text{all card slots correct}]$; all-or-nothing binary reward	QWEN-ST
Hanabi state-tracking (MYCROFT)	Weighted state-tracking	Slot-weighted card accuracy (Eq. 2); changed slots receive weight 1.0, unchanged slots 0.1	QWEN-HANABI, QWEN-HANABI+, GPT-OSS-120B line, Interference runs
Hanabi move-rating (SHERLOCK/MYCROFT)	LLM-as-judge	Move ratings scored in $[-1, 1]$ by a strong reasoning model (o3 / Grok-3-mini / Grok-4.1 Fast Reasoning)	QWEN-HANABI, QWEN-HANABI+, GPT-OSS-120B line, Interference runs
PI-LOGIC control	Answer match	$R = \mathbf{1}[\hat{a} = a^*]$; binary match against gold answer	PI-LOGIC control

J Evaluation Setup

We evaluate all Hanabi scores in Appendix D on fixed seeds 1001–1010 across all three scaffolds and 2–5 player settings, using the prompts in Appendix K.

For the MYCROFT state-tracking probe in Section 3.2, we curate 500 held-out 5P MYCROFT prompts from Grok-4.1-Fast-Reasoning trajectories [50] on seeds 1001–1010. We score these with the weighted state-tracking reward in Equation 2. For the FIREWORKS state-tracking probe (Appendix E), we curate 61 held-out 5P prompts from Gemini-2.5-Pro trajectories [12] on the held-out seeds, and score them with the strict exact-match reward.

For PI-LOGIC evaluation, we use 100 hard examples from the logic subset of INTELLECT-3-RL [32]. We select examples whose dataset-reported `avg@8_qwen3_4b_instruct_2507` is zero, sample 8 completions per example from QWEN3-4B, QWEN-ST, and QWEN-HANABI+, and score by exact answer match. We report Pass@8 and average completion length in Section 3.3 and Table 2.

K Prompts

K.1 WATSON Prompt

WATSON Prompt & Response

Input Prompt:
{ **"system_prompt"**: "You are an expert AI in the cooperative card game Hanabi. Your goal is to help the team achieve the highest possible score (max 25).
Analyze the entire game state provided, including your hand knowledge, visible hands of other players, fireworks, discards, deck size, lives, and info tokens.
Consider all strategic priorities:
1. **Safe Plays:** Prioritize playing cards you KNOW are playable on the fireworks.
2. **Useful Clues:** If no safe play and info tokens > 0, consider giving clues that enable immediate plays, save critical cards, or provide significant new information without being redundant.
3. **Safe Discards:** If no safe play and no high-value clue (or info tokens == 0), discard the safest possible card.
Explain your reasoning clearly, referencing the game state, and then state your chosen move number.

OUTPUT FORMAT:
Reasoning: [Your detailed reasoning justifying your choice based on the game state and strategic priorities]
Chosen Move Number: [number]",
"user_prompt": "You are Player 1. Analyze the game state and propose the best move number. Explain your reasoning.

Game State:
P1 (2p Game). Lives: 3, Info: 1, Deck: 0.
Fireworks: R2 Y4 G2 W3 B1. Discards: 1 red card rank 1, 1 red card rank 2, 2 red cards rank 3, 1 red card rank 4, 2 yellow cards rank 1, 1 yellow card rank 2, 1 yellow card rank 4, 1 yellow card rank 5, 1 green card rank 1, 1 green card rank 2, 1 green card rank 3, 2 green cards rank 4, 1 green card rank 5, 2 white cards rank 1, 1 white card rank 2, 1 white card rank 3, 2 white cards rank 4, 2 blue cards rank 1, 1 blue card rank 2, 2 blue cards rank 4, 1 blue card rank 5.
FINAL ROUND! 1 turns left (P1 is last).
Visible Hands: P0 Hand: [Blue 3, Red 1, Red 4, White 5, Blue 3]
Your Knowledge (Hints): [UnknownColor UnknownRank, UnknownColor UnknownRank, UnknownColor 3, Blue 2, UnknownColor UnknownRank] (Indices 0–4).
Others' Knowledge: P0 Knows: [UnknownColor 3, UnknownColor 1, UnknownColor UnknownRank, UnknownColor UnknownRank, UnknownColor UnknownRank]

Legal Moves:
0. (Discard 0)
1. (Discard 1)
2. (Discard 2)
3. (Discard 3)
4. (Discard 4)
5. (Play 0)
6. (Play 1)
7. (Play 2)
8. (Play 3)
9. (Play 4)
10. (Reveal player +1 color R)
11. (Reveal player +1 color W)
12. (Reveal player +1 color B)
13. (Reveal player +1 rank 1)
14. (Reveal player +1 rank 3)

15. (Reveal player +1 rank 4)
16. (Reveal player +1 rank 5)

Current Info Tokens: 1

Output Format:
Reasoning: [Your detailed reasoning justifying your choice based on the game state and strategic priorities]
Move Ratings: [Rate each legal move from -1 (terrible) to 1 (excellent), like "Move 0: 0.5, Move 1: -0.3, Move 2: 1.0, ..."]
Chosen Move Number: [number]"
}
response: "Reasoning:

It's the last turn (no cards left in deck, P1's only action). The current fireworks are R2, Y4, G2, W3, B1 (total 12). The only way to increase score is to play a card now. You know your card at index 3 is Blue 2, which is exactly the next playable card on the blue firework (B1→B2). Playing it gains +1 point, raising the score to 13.
All other play slots are uncertain and risk a misplay for no guaranteed score; discards or clues yield zero benefit (P0 won't get another action and you can't draw). Therefore the uniquely correct move is to play card 3.

Move Ratings:
Move 0 (Discard 0): -0.5
Move 1 (Discard 1): -0.5
Move 2 (Discard 2): -0.5
Move 3 (Discard 3): -0.5
Move 4 (Discard 4): -0.5
Move 5 (Play 0): 0.0
Move 6 (Play 1): 0.0
Move 7 (Play 2): 0.2
Move 8 (Play 3): 1.0
Move 9 (Play 4): 0.0
Move 10 (Clue P0 color R): -1.0
Move 11 (Clue P0 color W): -1.0
Move 12 (Clue P0 color B): -1.0
Move 13 (Clue P0 rank 1): -1.0
Move 14 (Clue P0 rank 3): -1.0
Move 15 (Clue P0 rank 4): -1.0
Move 16 (Clue P0 rank 5): -1.0

Chosen Move Number: 8"

K.2 SHERLOCK Prompt

SHERLOCK Prompt & Response

"user_prompt": "You are a master of hanabi game. You are playing a game of Hanabi with 2 players. Hanabi is a cooperative card game where players work together to create a series of fireworks by playing cards in ascending numerical order starting from 1. Each player holds their cards facing outward so that all players can see everyone else's cards but not their own. The objective is to play cards in sequence (1 through 5) for each color without making mistakes. There are 5 different colors and each color has cards numbered 1 to 5.

Key Rules:

On your turn, you have three types of possible actions:

Give a Hint(Reveal): Provide a hint to another player about their cards, specifying either a color or a number present in their hand. Hints must be accurate and can only reveal positions of cards matching the hint.
Discard a Card: Discard one of your own cards to potentially gain an Info token.
Play a Card: Attempt to play a card from your hand. If played correctly in sequence, it adds to the fireworks; if not, it reduces one life token.

Tokens:
Life Tokens: Deducted when a wrong card is played.
Info Tokens: Used to give clues.
Illegal Moves: Playing a card that cannot be placed properly costs a life token. If life tokens reach zero, the game ends in failure.
Game End: The game ends when all fireworks are completed (perfect score of 25), or when the deck is exhausted and each player has taken one final turn, or when the players run out of life tokens.

State Representation: The game state is represented with the following details:

Life tokens: Number of remaining life tokens.
Info tokens: Number of available information tokens.
Fireworks: Current progress on each firework color (e.g., R1, Y0, G1, W0, B0).
Discards: Cards that have been discarded.

Your Role:

You are one of the players, cooperating with others to maximize the total score of the fireworks (the number of cards correctly played in sequence).
Although you cannot see your own cards, you can see the cards in the hands of your teammates. Use hints, discards, and plays strategically to guide the team towards successful sequences.

Remember, communication is limited to hints about colors or numbers only, and sharing illegal or extraneous information is not allowed. Work together, follow the rules, and aim for the highest cooperative score possible!

Below is the current detailed state information.

Game State:
There are 3 life tokens and 2 info tokens remaining.
The fireworks progress: R stack is at 5, Y stack is at 5, G stack is at 3, W stack is at 2, B stack is at 4.
Your hand contains the following cards:
Card 0:
- Known info: 'XX'. No hints about this card's color or rank have been given yet.
- Could be any of these colors: Red, Yellow, Green, White with ranks: 1, 3, 4, 5.
Card 1:
- Known info: 'XX'. No hints about this card's color or rank have been given yet.
- Could be any of these colors: Red, Yellow, Green, White with ranks: 1, 2, 3, 4, 5.
Card 2:
- Known info: 'XX'. No hints about this card's color or rank have been given yet.
- Could be any of these colors: Red, Yellow, Green, White with ranks: 1, 2, 3, 4, 5.
Card 3:
- Known info: 'BX'. Known: color is blue.
- Could be any of these colors: Blue with ranks: 1, 2, 3, 4, 5.
Card 4:
- Known info: 'XX'. No hints about this card's color or rank have been given yet.
- Could be any of these colors: Red, Yellow, Green, White, Blue with ranks: 1, 2, 3, 4, 5.
From your perspective, you can see the other players' hands clearly. Here's what you observe:
Player +1's hand:

- A card: You can see the card: 'W1', This player has no specific hints about the card's identity, This player knows it could be any of these colors: Yellow, Green, White with ranks: 1, 2, 3.
- A card: You can see the card: 'W2', This player has no specific hints about the card's identity, This player knows it could be any of these colors: Red, Yellow, Green, White with ranks: 1, 2, 3.
- A card: You can see the card: 'Y4', This player has no specific hints about the card's identity, This player knows it could be any of these colors: Red, Yellow, Green, White with ranks: 1, 2, 3, 4, 5.
- A card: You can see the card: 'R3', This player has no specific hints about the card's identity, This player knows it could be any of these colors: Red, Yellow, Green, White, Blue with ranks: 1, 2, 3, 4, 5.

There are 0 cards remaining in the deck. The discard pile contains: 2 red cards rank 1, 1 red card rank 4, 1 yellow card rank 1, 1 yellow card rank 2, 1 yellow card rank 3, 2 green cards rank 1, 1 green card rank 2, 1 green card rank 3, 2 green cards rank 4, 1 green card rank 5, 1 white card rank 1, 2 white cards rank 3, 1 white card rank 5, 2 blue cards rank 1, 1 blue card rank 2, 1 blue card rank 3, 1 blue card rank 5.

FINAL ROUND: The deck is empty. You are the final player and this is the final turn for the whole game.

Please think step by step based on the current state
 # Think step by step

Evaluate Playable Cards in Hand

Look at each card in your hand.
Cross-reference with the current game state to see if any card can be immediately played to complete or extend a firework stack.
Consider hints you have received about each card (color/rank information) to determine if it might be safe to play.
If a card can be played without risk, prioritize playing it to score a point.

Consider Teammates' Hands and Hint Opportunities

Analyze the visible cards in your teammates' hands.
Identify if any of their cards can now be played based on the current firework stacks or previous hints.
If you notice a teammate holds a card that can be played but they may not realize it, think about what hints you could give them.
Use hints to communicate critical information, such as color or rank, to help them make the right play.
Choose the hint that maximizes the chance for a correct play while considering the limited hint tokens.

Assess Discard Options to Gain Info Tokens

Look for cards in your hand that are least likely to be playable or helpful in the near future.
Consider the remaining deck composition and cards already played/discarded to predict the value of each card.
Discard a card that you believe to be least useful to gain an Info token, especially if no immediate playable or hint options are available.
Ensure that discarding this card won't permanently remove a critical card needed to complete any firework stack.

Now it's your turn. You can choose from the following legal actions:

The legal actions are provided in a mapping of action identifiers to their descriptions:

{0: '((Discard 0))', 1: '((Discard 1))', 2: '((Discard 2))', 3: '((Discard 3))', 4: '((Discard 4))', 5: '((Play 0))', 6: '((Play 1))', 7: '((Play 2))', 8: '((Play 3))', 9: '((Play 4))', 10: '((Reveal player +1 color R))', 11: '((Reveal player +1 color Y))', 12: '((Reveal player +1 color W))', 13: '((Reveal player +1 rank 1))', 14: '((Reveal player +1 rank 2))', 15: '((Reveal player +1 rank 3))', 16: '((Reveal player +1 rank 4))'}

(Reveal player +N color C): Give a hint about color C to the player who is N positions ahead of you.
(Reveal player +N rank R): Give a hint about rank R to the player who is N positions ahead.
(Play X): Play the card in position X from your hand (Card 0, Card 1, Card 2, etc.).
(Discard X): Discard the card in position X from your hand (Card 0, Card 1, Card 2, etc.).

Based on the annotated state and the list of legal actions, decide on the most appropriate move to make. Consider factors like current tokens, firework progress, and information available in hands. Then, output one of the legal action descriptions as your chosen action.

Your output should be in this format:
{r̈eason": string, äction": int} And the action should be one of the legal actions provided above. You can only use json valid characters. When you write json, all the elements (including all the keys and values) should be enclosed in double quotes!!!

CRITICAL: Also include move ratings in this exact JSON format:
{
"move_ratings": [
{
action: 0,
rating: 0.1},
{
action: 1,
rating: -0.3},
{
action: 2,
rating: 0.9},
... (one entry for each legal move)
],
reason: Your detailed reasoning for the chosen action,
action: 2
}

IMPORTANT FORMATTING RULES:
- Rate each legal move from -1 (terrible) to 1 (excellent)
- Include ALL legal moves in the move_ratings array
- The action field should be the index of your chosen move
- Use valid JSON with proper quotes around all strings

To win, you need to play the cards in the correct sequence and maximize the total score of the fireworks. Good luck!

Calculate the probability of each card in your hand and the other players' hands to make better decisions.

Card Distribution and Probability Calculation:
- Each color has a specific number of cards per rank:
* Rank 1: 3 cards per color (15 total)
* Rank 2: 2 cards per color (10 total)
* Rank 3: 2 cards per color (10 total)
* Rank 4: 2 cards per color (10 total)
* Rank 5: 1 card per color (5 total)

- Total deck: 50 cards (5 colors \times 10 cards = 50)

Try to save the critical cards like rank 5, second card of each color, rank 2,3,4.

When evaluating unknown cards (your own or others'), calculate probabilities by:
1. Take the initial distribution of cards and subtract the cards you can see in other players' hands
2. Subtract cards you can see in the fireworks stacks
3. Subtract cards that have been discarded (check the discard pile)
4. Calculate probability

Use these probability calculations to make better decisions about plays, hints, and discards. Make use of the possible cards/ranks provided actively for your decisions and probability calculations. They were gathered from historical clues. For example, if you see a card could only be green, yellow we can deduce that the card is not red, blue or white. If you see a card could only be rank 1, 2, 3 we can deduce that the card is not rank 4 or 5."
}

Output:
{
"move_ratings": [
{ **"action"**: 0, **"rating"**: 0.7 },
{ **"action"**: 1, **"rating"**: 0.7 },
{ **"action"**: 2, **"rating"**: 0.7 },
{ **"action"**: 3, **"rating"**: 0.8 },
{ **"action"**: 4, **"rating"**: 0.7 },
{ **"action"**: 5, **"rating"**: -0.5 },
{ **"action"**: 6, **"rating"**: -0.5 },
{ **"action"**: 7, **"rating"**: -0.5 },
{ **"action"**: 8, **"rating"**: -0.6 },
{ **"action"**: 9, **"rating"**: -0.5 },
{ **"action"**: 10, **"rating"**: -1.0 },
{ **"action"**: 11, **"rating"**: -1.0 },
{ **"action"**: 12, **"rating"**: -1.0 },
{ **"action"**: 13, **"rating"**: -1.0 },
{ **"action"**: 14, **"rating"**: -1.0 },
{ **"action"**: 15, **"rating"**: -1.0 },
{ **"action"**: 16, **"rating"**: -1.0 }
],
"reason": "With the deck empty and this as our final turn, there are no remaining playable cards. The cards needed to advance the fireworks (G4, G5, W3, W5, B5) are all in the discard pile. Any attempt to play a card will result in a penalty. Giving a hint is also pointless, as this is the last turn of the game. Therefore, the only logical move is to discard a card to end the game without losing a life token. The card in position 3 is known to be Blue, and given the state of the board and discards, it is unplayable. Discarding this known useless card is the safest and most optimal action.",
"action": 3
}

K.3 MYCROFT Prompt

MYCROFT prompt and response

Input

Hanabi Explanation

You are a master of hanabi game. You are playing a game of Hanabi with 5 players. Hanabi is a cooperative card game where players work together to create a series of fireworks by playing cards in ascending numerical order starting from 1. Each player holds their cards facing outward so that all players can see everyone else's cards but not their own. The objective is to play cards in sequence (1 through 5) for each color without making mistakes. There are 5 different colors and each color has cards numbered 1 to 5.
Key Rules:

On your turn, you have three types of possible actions:

Give a Hint(Reveal): Provide a hint to another player about their cards, specifying either a color or a number present in their hand. Hints must be accurate and can only reveal positions of cards matching the hint.
Discard a Card: Discard one of your own cards to potentially gain an Info token.
Play a Card: Attempt to play a card from your hand. If played correctly in sequence, it adds to the fireworks; if not, it reduces one life token.

Tokens:
Life Tokens: Deducted when a wrong card is played.
Info Tokens: Used to give clues.
Illegal Moves: Playing a card that cannot be placed properly costs a life token. If life tokens reach zero, the game ends in failure.
Game End: The game ends when all fireworks are completed (perfect score of 25), or when the deck is exhausted and each player has taken one final turn, or when the players run out of life tokens.

State Representation: The game state is represented with the following details:

Life tokens: Number of remaining life tokens.
Info tokens: Number of available information tokens.
Fireworks: Current progress on each firework color (e.g., R1, Y0, G1, W0, B0).
Discards: Cards that have been discarded.

Persona Setup

Your Role:

You are one of the players, cooperating with others to maximize the total score of the fireworks (the number of cards correctly played in sequence).
Although you cannot see your own cards, you can see the cards in the hands of your teammates.
Use hints, discards, and plays strategically to guide the team towards successful sequences.

Remember, communication is limited to hints about colors or numbers only, and sharing illegal or extraneous information is not allowed. Work together, follow the rules, and aim for the highest cooperative score possible!

Strategic Guidelines

Please think step by step based on the current state

Think step by step

Evaluate Playable Cards in Hand

Look at each card in your hand.
Cross-reference with the current game state to see if any card can be immediately played to complete or extend a firework stack.
Consider hints you have received about each card (color/rank information) to determine if it might be safe to play.
If a card can be played without risk, prioritize playing it to score a point.

Consider Teammates' Hands and Hint Opportunities

Analyze the visible cards in your teammates' hands.

Identify if any of their cards can now be played based on the current firework stacks or previous hints.
If you notice a teammate holds a card that can be played but they may not realize it, think about what hints you could give them.
Use hints to communicate critical information, such as color or rank, to help them make the right play.
Choose the hint that maximizes the chance for a correct play while considering the limited hint tokens.

Assess Discard Options to Gain Info Tokens

Look for cards in your hand that are least likely to be playable or helpful in the near future.
Consider the remaining deck composition and cards already played/discarded to predict the value of each card.
Discard a card that you believe to be least useful to gain an Info token, especially if no immediate playable or hint options are available.
Ensure that discarding this card won't permanently remove a critical card needed to complete any firework stack.

The legal actions are provided in a mapping of action identifiers to their descriptions:

Example of legal actions:
(Reveal player +N color C): Give a hint about color C to the player who is N positions ahead of you.
(Reveal player +N rank R): Give a hint about rank R to the player who is N positions ahead.
(Play X): Play the card in position X from your hand (Card 0, Card 1, Card 2, etc.).
(Discard X): Discard the card in position X from your hand (Card 0, Card 1, Card 2, etc.).

Based on the annotated state and the list of legal actions, decide on the most appropriate move to make. Consider factors like current tokens, firework progress, and information available in hands. Then, output one of the legal action descriptions as your chosen action.

Output Specification

WHAT TO RETURN
Produce one JSON object (no markdown fences) with these exact top-level keys in order:
1. "move_ratings" – every legal move once, e.g. [{"action":0,"rating":0.2}, . . .] (ratings in [-1,1]).
2. "deduction" – what you and every other player know about their current cards.
3. "reason" – brief justification.
4. "action" – integer index of the chosen move.
All keys/strings must be double-quoted JSON.

Example structure (not content):
```
{
"move_ratings": [
{"action": 0, "rating": 0.1},
{"action": 1, "rating": -0.3},
{"action": 2, "rating": 0.9}
],
"deduction": {
"player+1": {card1: color is .. or color cannot be . rank is .. or rank cannot be. card2: ....},
"player+2": {....} and so on ]
"you": {"card0": "...", "card1": "...", "card2": "...", "card3": "..."},
"player+1": {"card0": "...", "card1": "...", "card2": "...", "card3": "..."},
"player+2": {"card0": "...", "card1": "...", "card2": "...", "card3": "..."},
"player+3": { ... },
"player+4": { ... }
},
"reason": "Your detailed reasoning for the chosen action",
"action": 2
}
```

State management protocol

CRITICAL: The deduction block must reflect, for this turn's state, what you AND every other player know about their current cards. Follow the step-by-step logic below each turn:

Definition: The deduction field must track the accumulated knowledge a player has about their own cards by listing all remaining possibilities for color and rank. This is built from the complete public history of hints and actions.

Deduction Logic (Follow these steps for each player):

1. Recall Previous State: Start with the list of possibilities for each card from the previous turn. (For Turn 1, all cards start with "color could be R, Y, G, W, B; rank could be 1, 2, 3, 4, 5").

2. Analyze the Most Recent Action: Look at the last move made before your turn.

* If a Hint was GIVEN TO this Player:
* Update with Positive Information: For the card(s) identified by the hint, narrow down the possibilities. If the hint was "Blue," the deduction for that card's color becomes "color is Blue."
* Update with Negative Information (MANDATORY): For all other cards in their hand not identified by the hint, you MUST remove the hinted value from their list of possibilities. (e.g., color possibilities become "R, Y, G, W").

* If this Player ACTED (Played or Discarded):
* This is a critical state update. Follow this sequence carefully:
* The card they acted on is removed from their hand.
* Retain Knowledge: For all other cards remaining in their hand, their known information is retained, but their position shifts to fill the gap.
* The new card drawn into the last slot of their hand is a complete unknown. Its deduction is: "color could be R, Y, G, W, B; rank could be 1, 2, 3, 4, 5."

3. Synthesize and Format: Present the final list of possibilities for each card in its new position.

Example of Correct Deduction:

* Scenario: Player+1 has a hand of R2, B4, W2. It is your turn. In the previous round, another player gave Player+1 a "rank 2" hint.
* Your Deduction Output for Player+1 MUST be:

"player+1": {
"card0": "color could be R, Y, G, W, B; rank is 2",
"card1": "color could be R, Y, G, W, B; rank could be 1, 3, 4, 5",
"card2": "color could be R, Y, G, W, B; rank is 2"
}

Example of a Player Action (Play/Discard):

* Scenario: It is Turn 5. On Turn 4, Player+1 had the following knowledge about their 4-card hand:
* card0: "color could be R, Y, G, W, B; rank is 2"
* card1: "color is Blue; rank could be 3, 4"
* card2: "color could be R, Y, G, W, B; rank is 5"
* card3: "color could be Y, G, W, B; rank could be 1, 2, 3, 4, 5" (They were previously told their other cards were not Red)

* Action: On their turn, Player+1 plays card 1.

* Your Deduction Output for Player+1 on Turn 5 MUST be:

"player+1": {
"card0": "color could be R, Y, G, W, B; rank is 2",
"card1": "color could be R, Y, G, W, B; rank is 5",
"card2": "color could be Y, G, W, B; rank could be 1, 2, 3, 4, 5",
"card3": "color could be R, Y, G, W, B; rank could be 1, 2, 3, 4, 5"
}

(Notice how the knowledge for the old card 0 remains at position 0, the knowledge for the old

card 2 shifts to position 1, the knowledge for the old card 3 shifts to position 2, and the new card at position 3 is completely unknown).

Do not be lazy. You MUST perform this analysis for your own hand plus all four other players, covering every card, to keep the deduction block 100 % accurate. An incorrect deduction state will lead to poor team performance.
FORMATTING RULES
• Rate each legal move from -1 (terrible) to 1 (excellent).
• Include all moves in move_ratings.
• "action" is the index of your chosen move.
• Output must be valid JSON.

To win, you need to play the cards in the correct sequence and maximize the total score of the fireworks. Good luck!

Calculate the probability of each card in your hand and the other players' hands to make better decisions.

Card Distribution and Probability Calculation:
- Each color has a specific number of cards per rank:
* Rank 1: 3 cards per color (15 total)
* Rank 2: 2 cards per color (10 total)
* Rank 3: 2 cards per color (10 total)
* Rank 4: 2 cards per color (10 total)
* Rank 5: 1 card per color (5 total)
- Total deck: 50 cards (5 colors × 10 cards = 50)

Try to save the critical cards like rank 5, second card of each color, rank 2,3,4.

When evaluating unknown cards (your own or others'), calculate probabilities by:
1. Take the initial distribution of cards and subtract the cards you can see in other players' hands
2. Subtract cards you can see in the fireworks stacks
3. Subtract cards that have been discarded (check the discard pile)
4. Calculate probability

Use these probability calculations to make better decisions about plays, hints, and discards. Make use of the possible cards/ranks provided actively for your decisions and probability calculations. They were gathered from historical clues. For example, if you see a card could only be green, yellow we can deduce that the card is not red, blue or white. If you see a card could only be rank 1, 2, 3 we can deduce that the card is not rank 4 or 5.
Except for your first turn, you will receive the previous turn's game state and your last reasoning; use them for context, but your deduction block must describe knowledge in the current state only.

Below is the current detailed state information.

Game State Information

Game State:
You are Player P2, Turn 58
Since your last turn the following actions occurred:
- P3 (Reveal player P4 color W) | Fireworks: R3 Y4 G2 W3 B2 | Info: 0
- P4 (Play 1) | Fireworks: R3 Y4 G2 W4 B2 | Info: 0
- P0 (Discard 0) | Fireworks: R3 Y4 G2 W4 B2 | Info: 1
- P1 (Reveal player P2 color G) | Fireworks: R3 Y4 G2 W4 B2 | Info: 0

There are 3 life tokens and 0 info tokens remaining.
The fireworks progress: R stack is at 3, Y stack is at 4, G stack is at 2, W stack is at 4, B stack is at 2.
Your hand (what you know):
This is your explicit knowledge, showing only what you've been directly told through clues.
For further deductions (what each card cannot be, based on prior history and reasoning), use your deduction block.
Card 0: unknown

Card 1: unknown
Card 2: unknown
Card 3: G, unknown rank
From your perspective, you can see the other players' hands clearly. Here's what you observe:
Player +3's hand:
- Y5
- G5
- Y3
Player +4's hand:
- B1
- R1
- R4
- W1
Player +1's hand:
- G2
- R2
- B4
- R3
Player +2's hand:
- B1
- R1
- Y2
There are 0 cards remaining in the deck. The discard pile contains: 1 red card rank 4, 1 red card rank 5, 1 yellow card rank 1, 1 yellow card rank 4, 2 green cards rank 1, 1 green card rank 3, 2 green cards rank 4, 1 white card rank 2, 1 white card rank 3, 1 white card rank 4, 1 white card rank 5, 1 blue card rank 2, 2 blue cards rank 3, 1 blue card rank 4.

FINAL ROUND: The deck is empty. You are the final player and this is the final turn for the whole game.

Please note down all the deductions you make so that they will help you in future turns.

For example, if you have deduced that a card cannot be yellow based on previous clues (e.g., when a yellow clue is given to the player and the card is not chosen, you know it is not yellow—so only RGBW are possible), or if you know a card cannot be rank 2 or 5 based on previous clues, write this down.

Write down all such deductions in the scratch pad, as they will be useful in future turns.

Use the history well. Avoid giving the same clue to the same player it would be redundant. Try to write down what other players know from the history and your previous turns' reasoning, and use this to plan your actions.

Legal moves this turn:
0: ((Discard 0))
1: ((Discard 1))
2: ((Discard 2))
3: ((Discard 3))
4: ((Play 0))
5: ((Play 1))
6: ((Play 2))
7: ((Play 3))

You have been given the previous game-state and your last reasoning

Previous Game State

PREVIOUS GAME-STATE:
You are Player P2, Turn 53
Since your last turn the following actions occurred:
- P3 (Reveal player P4 color W) | Fireworks: R3 Y4 G2 W2 B2 | Info: 0
- P4 (Play 0) | Fireworks: R3 Y4 G2 W3 B2 | Info: 0
- P0 (Discard 0) | Fireworks: R3 Y4 G2 W3 B2 | Info: 1

- P1 (Reveal player P4 rank 4) | Fireworks: R3 Y4 G2 W3 B2 | Info: 0

There are 3 life tokens and 0 info tokens remaining.
The fireworks progress: R stack is at 3, Y stack is at 4, G stack is at 2, W stack is at 3, B stack is at 2.
Your hand (what you know):
This is your explicit knowledge, showing only what you've been directly told through clues.
For further deductions (what each card cannot be, based on prior history and reasoning), use your deduction block.
Card 0: unknown
Card 1: unknown
Card 2: unknown
Card 3: unknown
From your perspective, you can see the other players' hands clearly. Here's what you observe:
Player +3's hand:
- G4
- Y5
- G5
- Y3
Player +4's hand:
- B1
- R1
- R4
- W1
Player +1's hand:
- G2
- R2
- B4
- R3
Player +2's hand:
- B1
- W4
- R1
- Y2
There are 1 cards remaining in the deck. The discard pile contains: 1 red card rank 4, 1 red card rank 5, 1 yellow card rank 1, 1 yellow card rank 4, 2 green cards rank 1, 1 green card rank 3, 1 green card rank 4, 1 white card rank 2, 1 white card rank 3, 1 white card rank 4, 1 white card rank 5, 1 blue card rank 2, 1 blue card rank 3, 1 blue card rank 4.

Agent's response from previous turn

PREVIOUS TURN RESPONSE:
{
"move_ratings": [
{"action": 0, "rating": -0.3},
{"action": 1, "rating": 0.1},
{"action": 2, "rating": 0.45},
{"action": 3, "rating": 0.35},
{"action": 4, "rating": -1.0},
{"action": 5, "rating": -0.2},
{"action": 6, "rating": -0.3},
{"action": 7, "rating": -0.3}
],
"deduction": {
"you": {
"card0": "color could be W, B; rank could be 1, 5",
"card1": "color could be G, W, B; rank could be 1, 3, 5",
"card2": "color could be Y, G, W, B; rank could be 1, 3, 5",
"card3": "color could be Y, G, W, B; rank could be 1, 3, 5"
},
"player+1": {
"card0": "color could be Y, G, W, B; rank could be 1, 2, 3, 4, 5",
"card1": "color could be Y, G, W, B; rank could be 1, 2, 3, 4, 5",

"card2": "color could be Y, G, W, B; rank could be 1, 2, 3, 4, 5",
"card3": "color could be Y, G, W, B; rank could be 1, 2, 3, 4, 5"
},
"player+2": {
"card0": "color could be Y, G, B; rank could be 1, 2, 5",
"card1": "color is W; rank is 4",
"card2": "color could be Y, G, B; rank could be 1, 2, 3, 5",
"card3": "color could be Y, G, W, B; rank could be 1, 2, 3, 5"
},
"player+3": {
"card0": "color could be Y, G, W, B; rank is 4",
"card1": "color could be Y, G, W, B; rank could be 1, 2, 3, 5",
"card2": "color could be Y, G, W, B; rank could be 1, 2, 3, 5",
"card3": "color could be Y, G, W, B; rank could be 1, 2, 3, 4, 5"
},
"player+4": {
"card0": "color could be Y, G, W, B; rank could be 1, 2, 3, 4, 5",
"card1": "color could be Y, G, W, B; rank could be 1, 2, 3, 4, 5",
"card2": "color could be Y, G, W, B; rank could be 1, 2, 3, 4, 5",
"card3": "color could be Y, G, W, B; rank could be 1, 2, 3, 4, 5"
}
},
"reason": "Only five cards are unaccounted for: Y1, G3, W1, B3, B5. Four are in our hand, one in the deck. Card 0 is either W1 or the unique, critical B5 (50 % risk), so we must not touch it. Card 1 is 25 % B5 and 50 % playable. Cards 2/3 each have just a 20 % chance of being B5 and 40 % chance of being playable. Discarding card 2 safely creates the badly-needed information token while minimizing the chance of trashing the lone B5 or a likely playable card.",
"action": 2

"response_content": "{
"move_ratings": [
{"action": 0, "rating": -0.2},
{"action": 1, "rating": -0.2},
{"action": 2, "rating": -0.2},
{"action": 3, "rating": -1.0},
{"action": 4, "rating": -0.8},
{"action": 5, "rating": -0.8},
{"action": 6, "rating": -0.8},
{"action": 7, "rating": 1.0}
],
"deduction": {
"you": {
"card0": "color could be W, B; rank could be 1 or 5",
"card1": "color could be W, B; rank could be 1 or 5",
"card2": "color could be Y, W, B; rank could be 1 or 5",
"card3": "color is G; rank is 3"
},
"player+1": {
"card0": "color could be R, Y, G, W, B; rank could be 1, 2, 3, 4, 5",
"card1": "color could be R, Y, G, W, B; rank could be 1, 2, 3, 4, 5",
"card2": "color could be R, Y, G, W, B; rank could be 1, 2, 3, 4, 5",
"card3": "color could be R, Y, G, W, B; rank could be 1, 2, 3, 4, 5"
},
"player+2": {
"card0": "color could be Y, G, B; rank could be 1, 2, 5",
"card1": "color could be Y, G, B; rank could be 1, 2, 3, 5",
"card2": "color could be Y, G, B; rank could be 1, 2, 3, 5"
},
"player+3": {
"card0": "color could be R, Y, G, W, B; rank could be 1, 2, 3, 5",
"card1": "color could be R, Y, G, W, B; rank could be 1, 2, 3, 5",
"card2": "color could be R, Y, G, W, B; rank could be 1, 2, 3, 5"
},

"player+4": {
"card0": "color could be R, Y, G, W, B; rank could be 1, 2, 3, 4, 5",
"card1": "color could be R, Y, G, W, B; rank could be 1, 2, 3, 4, 5",
"card2": "color could be R, Y, G, W, B; rank could be 1, 2, 3, 4, 5",
"card3": "color could be R, Y, G, W, B; rank could be 1, 2, 3, 4, 5"
}
},
"reason": "Last hint told me all green cards; only my right-most card was marked, so cards 0-2 are not green. From public information all remaining green cards are: one G3 and one G5 (visible in P+3's hand). Both G1, both G4 and both G2 copies are gone or visible. Therefore my marked green card must be the sole G3, which is immediately playable (fireworks G is at 2). Playing it gives the team an extra point; discarding would add nothing, and any other play is almost certainly wrong.",
"action": 7
}"

K.4 FIREWORKS training data prompt

FIREWORKS Prompt

Input Prompt:
{
"prompt": "You are a deterministic Hanabi belief-state tracker.
Update the previous belief state using the action list.

Rules:
1. Chronology: The actions list is chronological; apply them in order. It includes actions made by all players, including you.
2. Card format: "KNOWN|POSS" (KNOWN=2 chars [RYGWBX][12345X], POSS=colors subset in RYGWB order then ranks subset in 12345 order).
3. NO-PROMOTION: Never change the KNOWN characters based on deduction (even if POSS becomes singleton). Only change KNOWN when a reveal directly hints/touches that card.
4. Reveal: Touched indices list which cards match the hinted value. Touched indices refer to the target's hand AT THAT MOMENT, after applying all prior actions in the list.
- For touched cards: Set the hinted KNOWN character; in POSS set only the hinted dimension to that singleton while leaving the other dimension unchanged.
- For non-touched cards: Remove that value from the POSS set.
5. Play/Discard: remove that card slot, shift remaining left.
- If drew=true: append a brand new unknown card "XX|RYGWB12345".
- If drew=false: the hand shrinks (no new card). Do not output a placeholder slot.

Mapping (relative to you):
{"player+2": "P0", "player+3": "P1", "player+4": "P2", "you": "P3", "player+1": "P4"}

Previous belief state:
{
"player+2": {
"card0": "XX|RYGWB12345",
"card1": "XX|RYGWB12345",
"card2": "XX|RYGWB12345",
"card3": "XX|RYGWB12345"
},
"player+3": {
"card0": "XX|RYWB2345",
"card1": "XX|RYWB2345",
"card2": "XX|RYWB2345",
"card3": "XX|RYGWB12345"
},

“player+4”: {
“card0”: “XX|RYGWB12345”,
“card1”: “XX|RYGWB12345”,
“card2”: “XX|RYGWB12345”,
“card3”: “XX|RYGWB12345”
},
“you”: {
“card0”: “XX|RYGWB12345”,
“card1”: “XX|RYGWB12345”,
“card2”: “XX|RYGWB12345”,
“card3”: “XX|RYGWB12345”
},
“player+1”: {
“card0”: “XX|RYGB12345”,
“card1”: “WX|W12345”,
“card2”: “XX|RYGB12345”,
“card3”: “WX|W12345”
}
}

Actions since previous belief:
[
{“actor”: “P3”, “type”: “REVEAL_RANK”, “target”: “P4”, “value”: “1”, “touched”: [1, 3]},
{“actor”: “P4”, “type”: “PLAY”, “index”: 1, “drew”: true},
{“actor”: “P0”, “type”: “REVEAL_RANK”, “target”: “P1”, “value”: “5”, “touched”: [2]},
{“actor”: “P1”, “type”: “REVEAL_RANK”, “target”: “P4”, “value”: “2”, “touched”: [3]},
{“actor”: “P2”, “type”: “REVEAL_COLOR”, “target”: “P0”, “value”: “R”, “touched”: [1, 3]}
]

Belief state (fill in the values in this JSON template):
{
“player+2”: {“card0”: “”, “card1”: “”, “card2”: “”, “card3”: “”},
“player+3”: {“card0”: “”, “card1”: “”, “card2”: “”, “card3”: “”},
“player+4”: {“card0”: “”, “card1”: “”, “card2”: “”, “card3”: “”},
“you”: {“card0”: “”, “card1”: “”, “card2”: “”, “card3”: “”},
“player+1”: {“card0”: “”, “card1”: “”, “card2”: “”, “card3”: “”}
}

Output EXACTLY two lines, no markdown:
Reasoning: ⟨brief reasoning text⟩
JSON: ⟨completed belief state JSON⟩”
}

NeurIPS Paper Checklist

1. **Claims**

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

   Answer: [Yes]

   Justification: The abstract and introduction (Section 1) state three main claims-scaffold - local gains, competence-gated transfer, and trainability/deliberation transfer - and Sections 3.1–3.4 provide corresponding experimental evidence for each.

   Guidelines:

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

2. **Limitations**

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

   Answer: [Yes]

   Justification: The paper includes a dedicated limitations Appendix C.

   Guidelines:

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

3. **Theory assumptions and proofs**

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

Answer: [N/A]

Justification: The paper does not present formal theoretical results.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

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

Answer: [Yes]

Justification: Appendix I describe base models, dataset sizes, reward definitions, batch sizes, and generation length. Appendix K provides full prompts for all scaffolds. The authors commit to open-sourcing code, data, and checkpoints upon acceptance (Appendix H).

Guidelines:

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

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

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

   Answer: [No]

   Justification: Code, data, and checkpoints are not included with the submission. The authors state they will be open-sourced upon acceptance (Section H).

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (https://neurips.cc/public/guides/CodeSubmissionPolicy) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (https://neurips.cc/public/guides/CodeSubmissionPolicy) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

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

   Answer: [Yes]

   Justification: Training recipes (batch size, rollout count, max generation length) are in Appendix I. Evaluation seeds (1001–1010), player counts (2–5P), and reporting methodology (IQM with 95% bootstrap CIs) are specified in Appendix J

   Guidelines:

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

7. **Experiment statistical significance**

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

   Answer: [Yes]

   Justification: Table 3 (Appendix D) reports IQM scores with 95% bootstrap confidence intervals for every model–scaffold–player-count combination. The main tables report means over these IQMs.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.

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

8. **Experiments compute resources**

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

   Answer: [No]

   Justification: The paper does not report GPU types, wall-clock training times, or total compute used for any of the experimental runs.

   Guidelines:

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

9. **Code of ethics**

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

   Answer: [Yes]

   Justification: The research studies RL training dynamics in a cooperative card game and does not raise ethical concerns. The work conforms with the NeurIPS Code of Ethics.

   Guidelines:

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

10. **Broader impacts**

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

    Answer: [N/A]

    Justification: This is foundational research studying transfer properties of RL post-training in a card-game testbed. There is no direct path to negative societal applications.

Guidelines:

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

11. **Safeguards**

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

Answer: [N/A]

Justification: The released assets will be small RL-finetuned checkpoints for Hanabi play and a state-tracking dataset. These pose negligible misuse risk.

Guidelines:

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

12. **Licenses for existing assets**

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

Answer: [Yes]

Justification: The paper cites the Hanabi Learning Environment, base models Qwen3-4B and GPT-OSS-120B, the INTELLECT-3-RL dataset, and prior scaffold prompts from Ramesh et al. [33]. All are open-sourced. license – CC-BY 4.0

Guidelines:

- The answer [N/A] means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.

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

13. **New assets**

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

    Answer: [Yes]

    Justification: The paper introduces the FIREWORKS dataset (Appendix I) and multiple trained checkpoints. Prompt formats are documented in Appendix K. Full release with documentation is planned upon acceptance.

    Guidelines:

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

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

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

    Answer: [N/A]

    Justification: No crowdsourcing or human-subject experiments were conducted.

    Guidelines:

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

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

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

    Answer: [N/A]

    Justification: No human-subject research was conducted.

Guidelines:

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

16. **Declaration of LLM usage**

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

Answer: [Yes]

Justification: LLMs are core to the methodology. The base models (Qwen3-4B-Instruct-2507, GPT-OSS-120B) are the training subjects. OpenAI o3, Grok-3-mini, and Grok-4.1 Fast Reasoning are used to generate move-quality reward labels (Section 2). All uses are described in the paper.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.