

Why Do LLM Agents Fail in Exploring New Environments? A World-Modeling Perspective

Shiqi Chen^{*†1}, Tongyao Zhu^{†7}, Zian Wang^{†8}, Jinghan Zhang^{†3}, Kangrui Wang², Ruochen Zhou¹, Siyang Gao¹, Teng Xiao^{†5,6}, Yee Whye Teh⁴, Junxian He^{*3}, Manling Li^{*2}

¹City University of Hong Kong, ²Northwestern University

³The Hong Kong University of Science and Technology, ⁴Oxford University

⁵Allen Institute for AI (AI2), ⁶University of Washington, ⁷National University of Singapore

⁸The Hong Kong Polytechnic University

Abstract

Large Language Models (LLMs) as agents often fail to improve in *new* environments. We identify and characterize a failure mode we call *exploration collapse*: under reinforcement learning (RL) in environments whose states are unfamiliar to the policy, $Pass@k$, the probability that at least one of k sampled trajectories succeeds, drops markedly over training even as $Pass@1$ edges up, revealing increasingly brittle exploration; environments closer to the pretraining distribution show no such decline. We trace this collapse to weak grounding in environment states and dynamics, and study a simple remedy: explicitly teaching the agent to estimate the current state and predict its transitions before optimizing for reward. We instantiate it as SPA, an explore-then-exploit recipe that cold-starts the policy with a Self-Experience supervised finetuning (SFT) stage, collecting the model’s own interaction trajectories and supervising state and next-state prediction, and then runs standard RL. The resulting world model serves as a grounded *initialization* for RL rather than an inference-time planner. Across unseen environments, SPA consistently and substantially improves over vanilla RL: for example, it raises the Sokoban success rate from **25.6%** to **59.8%** on Qwen2.5-1.5B-Instruct, letting sub-3B models surpass a 20B baseline on these tasks. Controlled studies indicate that the gains track four factors: grounded state representations, explicit transition modeling, self-experience trajectories from a sufficiently strong exploration policy, and adequate coverage of transition data.

1 Introduction

Agentic reinforcement learning (RL) has become a primary framework for finetuning Large Language Model (LLM) agents (Wang et al., 2025b;

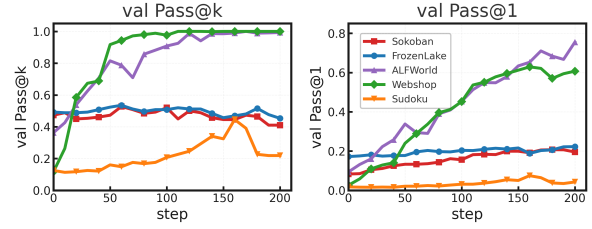


Figure 1: Validation score on five environments. Sokoban, FrozenLake and Sudoku are OOD for the models, where $Pass@k$ ($k=8$) drops with training. However, in ALFWorld and WebShop, $Pass@k$ increases greatly with training.

Jin et al., 2025). Using ReAct-style scaffolds (Yao et al., 2023) and optimizing for task rewards, this approach has been applied to computer-use planning (Cao et al., 2025), tool use (Schick et al., 2023), and web search (Wu et al., 2025; Jin et al., 2025). However, the performance degrades sharply in unseen environments that are beyond the LLM’s pre-training distribution. In such cases, the encountered state is unfamiliar to the policy model. We quantify this mismatch by combining the perplexity (PPL) of the state description with that of the state space. As shown in Table 1, Sokoban, FrozenLake, and Sudoku exhibit state PPLs far above random guess under these environments, yet ALFWorld and WebShop yield much lower state PPLs compared with random guess, indicating a stronger distributional shift in the former environments.

To diagnose when LLM agents struggle to learn in unseen environments, we empirically investigate agent performance during RL fine-tuning and uncover a systematic divergence between two key metrics: $Pass@1$, the success rate of the highest-probability trajectory, and $Pass@k$, whether at least one of the k sampled trajectories is successful. $Pass@k$ further captures whether an agent leverages a learned world model to diversify at-

^{*}Corresponding Author

[†]Core Contributor

tempts within a small rollout budget, without reward overfitting or brittle credit assignment. As shown in Figure 1, in unseen environments like Sokoban, FrozenLake, and Sudoku, standard RL causes Pass@k performance to consistently degrade, while Pass@1 increases marginally. This is different from familiar settings (e.g., ALFWorld and Webshop), where both metrics improve. This divergence reveals a fundamental limitation: current agentic RL develops a narrow, exploitative solution path (reflected in Pass@1), but fail to explore across multiple paths (in Pass@k), a phenomenon we call **exploration collapse**. We note that related large- k narrowing has been observed for single-turn RL with verifiable rewards (Yue et al., 2025; Wen et al., 2025; Zhai et al., 2026); our contribution is to study its manifestation in unfamiliar *multi-turn* environments, diagnose it as a grounding failure, and mitigate it with a grounded initialization (§7).

To this end, we ask an important research question: **How can we enable effective exploration for scalable agentic reinforcement learning in unseen environments?** We argue that a key lever for restoring exploration in unseen environments is to *ground* the agent in the environment’s states and dynamics—internalizing what we refer to as a **world model** (Xing et al., 2025) of its underlying rules. Unlike classical non-LLM RL that queries an explicit world model during planning (Janner et al., 2019; Yu et al., 2020), we internalize this knowledge *into the policy itself* as a cold-start initialization: by learning to describe the current state and predict its next state, the agent acquires a grounded prior over dynamics that it subsequently refines with RL. How to acquire such a prior effectively and scalably for LLM agents remains open. We answer this with SPA (Self-exPerience Agent)¹, a simple **explore-before-exploit** framework: the agent first gains self-experience to ground itself in the environment, then exploits this grounding to act effectively.

SPA focuses on exploration on both state space and action space: **1) State Estimation.** To explore the state space, we transform unfamiliar states to familiar ones. **2) Transition Modeling.** Exploration of the action space means making the environment’s transitions predictable with an accurate transition kernel $p_\theta(s_{t+1} | s_t, a_t)$. SPA scales improvements in RL: both Pass@1 and Pass@k

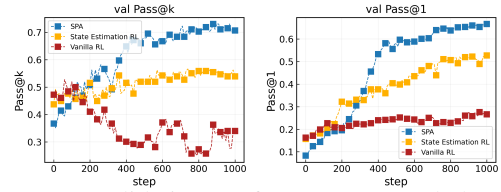


Figure 2: Validation performance on Sokoban and Frozen Lake. Red: Vanilla RL (StarPO baseline); Yellow: state-estimation RL; Blue: SPA (world-modeling SFT as a cold start, then state-estimation RL). Left: Pass@k; Right: Pass@1.

grow over training. We evaluate SPA on two well-studied grid worlds, *Sokoban* and *FrozenLake*, and a math-reasoning environment, *Sudoku*. These tasks demand multistep spatial reasoning and precise action control, which are unfamiliar for the pretrained model. Under a fixed budget of 1000 training steps, our method lifts average success rates on *Sokoban* from 25.6% to 59.8% and on *FrozenLake* from 22.1% to 70.9% (Figure 2). Qualitative case studies further show that the internal world model induces more deliberate, globally consistent plans that benefit subsequent RL. Compared against a reward-shaping world-modeling method under matched settings, SPA injects this grounding more effectively than adding a world-modeling reward during online RL (Wang et al., 2025a); we hypothesize that such an auxiliary reward interferes with the task objective rather than yielding reusable dynamics.

Finally, controlled analyses isolate which factors drive the gains. The improvements are not explained by additional finetuning alone, but track four ingredients: explicit transition modeling, grounded state representations, self-experience trajectories from a sufficiently strong exploration policy, and adequate coverage of transition data. Together they let the agent better interpret unfamiliar observations, model environment dynamics, and transfer to downstream policy optimization.

2 Unseen Environments Induce Exploration Collapse

We find that agentic RL behaves very differently across environments. In some environments, reinforcement learning improves both single-trajectory success and broader success across multiple attempts. In contrast, in unseen environments such as Sokoban, FrozenLake, and Sudoku, performance degrades much more sharply. These environments present states that are much less familiar to the policy model, making them harder to parse and reason

¹Code, configuration files, and checkpoints: <https://github.com/shiqichen17/SPA>.

Task	PPL	#States
Sokoban	163.9	7
+ State Estimation	19.6	
Frozen Lake	187.1	6
+ State Estimation	15.4	
Sudoku	15.5	5
+ State Estimation	7.3	
ALFWorld	6.0	$ V $
WebShop	11.7	$ V $

Table 1: Average PPL of state representations on Qwen2.5-Instruct-1.5B. #States denotes the number of possible values each grid cell can take (=random guess PPL).

about. As shown in Table 1, Sokoban, FrozenLake, and Sudoku exhibit much higher state perplexity relative to the size of their underlying state space, whereas environments such as ALFWorld and WebShop are much easier for the model to process. This suggests that the difficulty is due to a mismatch between the environment states and the model’s prior knowledge. We compute this perplexity over the serialized state-representation tokens, so we interpret it as a proxy for the *representational familiarity* of state strings to the LM, rather than as a calibrated measure of environment entropy or state-space size.

To better understand how RL fails in such environments, we examine two metrics throughout training: Pass@1, which measures the success rate of the highest-probability trajectory, and Pass@k, which measures the success rate across multiple sampled trajectories and thus reflects broader coverage over plausible solution paths. We observe a systematic divergence between these two metrics in unseen environments. As shown in Figure 1, in Sokoban, FrozenLake, and Sudoku, standard RL causes Pass@k to consistently deteriorate over training, while Pass@1 improves only marginally. This pattern is markedly different from training-aligned environments like ALFWorld and WebShop, where both metrics improve together, and pass@k fastly goes to 100 percent.

This divergence suggests that, in unseen environments, RL improves single-path exploitation without improving broader trajectory coverage. As training proceeds, the policy becomes narrower: it looks better at converging on one likely path, but fails to acquire the broader world knowledge needed for robust generalization. We therefore interpret the decline of Pass@k as evidence that exploration is collapsing. This observation motivates our paper: if unseen environments narrow exploration because the agent lacks grounded state and

transition understanding, improving such grounding should help mitigate this collapse, which we probe in the next section.

3 SPA: Internal World Modeling Restores Exploration

In this section, we introduce SPA, a simple and effective framework for improving LLM agent training. We show the framework in Figure 3. The central idea is to strengthen grounding through explicit world modeling prior to policy learning. Specifically, we decompose world modeling into two components: *state representation* and *transition dynamics*. By jointly modeling states and transition dynamics, the agent is regularized to follow structured state descriptions while interacting with the environment, thereby improving grounding. We train the world model via supervised fine-tuning on transition dynamics and use it to initialize policy learning. Our method does not rely on external knowledge or larger teacher models; instead, it learns entirely from the base model’s own experience. In the following, we present the state-representation schema and the transition-modeling procedure.

3.1 Optimizing State Estimation

We hypothesize that a key challenge in agent RL training is the distribution shift: observed states are unfamiliar relative to the model’s pretraining data, making them difficult to parse and reason about. This is especially pronounced when environment states are rendered as symbolic text strings. For example, a 6×6 Sokoban board is serialized as #####\n#####\n#_###\n#_####\n\n#0_XP#\n##### in Wang et al. (2025b). As shown in Table 1, such raw representations have high perplexity, whereas environments with natural-language state descriptions, such as ALFWorld and WebShop, exhibit much lower perplexity. To reduce this mismatch, we augment the raw state s'_t with an abstract representation b_t , giving $s_t = \text{Concat}(s'_t, b_t)$. In text games, b_t gives coordinates of key entities (player, boxes, goals). This lowers perplexity (Table 1) and improves both Pass@k and Pass@1 over the *Vanilla RL* baseline, as shown by *State Estimation RL* in Table 2. We next introduce transition-dynamics modeling on this representation.

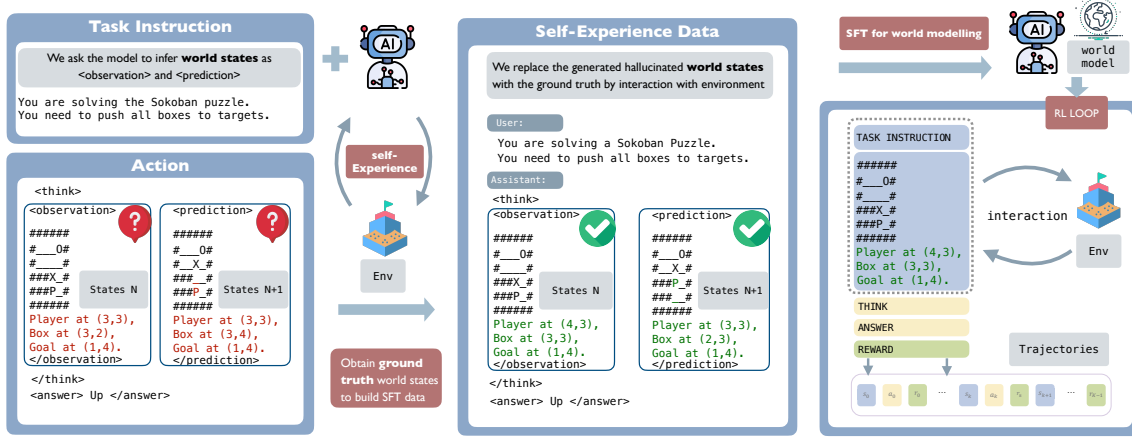


Figure 3: SPA equips an LLM agent with an inner world model. The world model is decomposed into two parts: (i) State Estimation, which converts raw observations into structured, informative natural-language descriptions of the world state; and (ii) Transition Modeling, trained in a self-experience Supervised Fine-Tuning setting to predict next-state representations from current states.

Model	Sokoban		FrozenLake		Sudoku	
	Pass@1	Pass@8	Pass@1	Pass@8	Pass@1	Pass@8
Qwen2.5-0.5B-Instruct	5.5	25.8	8.1	32.8	0.0	0.0
+Vanilla RL	16.9	35.9	22.8	30.1	0.0	0.0
+State Estimation RL	20.4	38.7	24.2	26.6	4.5	14.1
+World Model RL (VAGEN adaptation)	33.3	44.9	25.8	40.6	0.1	0.8
+ SPA	36.7	45.3	46.9	65.6	18.2	43.8
Qwen2.5-1.5B-Instruct	16.3	47.3	17.2	49.2	1.6	11.3
+Vanilla RL	25.6	34.0	22.1	30.7	0.0	0.0
+State Estimation RL	52.7	53.9	27.6	34.8	39.1	72.3
+World Model RL (VAGEN adaptation)	44.5	50.0	37.7	43.0	0.0	0.0
+ SPA	59.8	69.5	70.9	75.0	59.6	94.9
Qwen2.5-3B	12.5	35.9	6.9	26.9	0.0	0.0
+Vanilla RL	31.4	35.5	7.6	16.0	0.0	0.0
+State Estimation RL	26.2	27.7	24.7	35.5	24.1	26.2
+World Model RL (VAGEN adaptation)	31.9	43.0	30.7	34.0	0.0	0.0
+ SPA	49.7	58.2	41.3	46.1	69.9	89.8
LLaMA3.2-1B-Instruct	8.3	33.2	10.2	41.0	0.0	0.0
+Vanilla RL	21.2	39.8	10.8	24.6	0.1	1.2
+State Estimation RL	31.6	44.1	19.3	29.7	45.0	71.1
+World Model RL (VAGEN adaptation)	47.4	50.8	24.5	29.3	0.1	0.8
+ SPA	53.0	68.0	64.8	71.1	81.3	100
GPT-OSS-20B	45.8	84.4	68.8	100	61.8	100
+State Estimation	55.1	89.1	73.3	100	68.1	100

Table 2: Results on Sokoban/FrozenLake/Sudoku for different models. (Metrics in $\times 10^{-2}$).

3.2 Injecting Transition Modeling via Self-Experience Fine-tuning

Based on the interpretable environment states, we inject transition dynamics into the LLM agent via a self-experience finetuning process. It includes two stages: *trajectory collection via self-experience* and *dynamics transition training*. During the trajectory collection stage, we use the base model to freely explore and interact with the environment under a self-experience scenario to update the model’s belief of the world.

Self-Experience Data Generation We collect data by prompting the model to interact with the environment. This interaction repeats over T turns to

collect a trajectory $\tau = (s_0, o_0, a_0, r_0, \dots, s_T, o_T)$. At each step, the action a_t is parsed and executed. The environment then returns a reward r_t and transitions to a new state s_{t+1} , which gives the agent a new observation o_{t+1} . The structure of a_t is designed to incorporate explicit visual state reasoning, with a particular focus on world modeling. To reason about world states, we require the model to think before taking actions: model generation $a_t = (z_t, \hat{a}_t)$ is a sequence of text including both reasoning tokens z_t and executable action $\hat{a}_t$. To incorporate world modeling into the trajectories, we prompt the LLM agent to describe the current state and predict the next state during reasoning (Wang et al., 2025a). Specifically, the reasoning variable

z_t between `<think>` and `</think>` is required to explicitly represent both the current state and the predicted next state:

```
<think><observation> $\hat{s}_t$ </observation>[plan]
<prediction> $\hat{s}_{t+1}$ </prediction></think><answer> $\hat{a}_t$ 
</answer>.
```

For the next step, we then replace the model’s self beliefs about current states $\hat{s}_t$ and future states $\hat{s}_{t+1}$ with the ground-truth states s_t and s_{t+1} from the environment, resulting in the training trajectory:

```
<think><observation> $s_t$ </observation>[plan]
<prediction> $s_{t+1}$ </prediction></think><answer> $\hat{a}_t$ 
</answer>.
```

Here `[plan]` denotes optional free-form planning text, and $\hat{a}_t$ may contain several primitive actions separated by “|”. During SFT, we rewrite only the `<observation>` and `<prediction>` spans with grounded states and keep plans and actions as generated; we list the exact per-environment prompts in Appendix A. Next, we perform supervised fine-tuning (SFT) on these trajectories that reflect the environmental dynamics, enabling the model to learn environment dynamics. The resulting SFT model serves as a strong initialization for downstream RL training.

World Modeling. With the constructed trajectory $\tau_{1:T}$ above, and let $\tau_{<i}$ denote the prefix up to token $i - 1$, we optimize a simple masked token-level cross-entropy over tokens enclosed by the special tags to capture environmental dynamics and to internalize world models in agents.

$$\mathcal{L}_W(\theta) = -\frac{1}{\sum_{i=1}^T M_i} \sum_{i=1}^T M_i \log p_\theta(\tau_i \mid \tau_{<i}), \quad (1)$$

where $M_i = \mathbb{1}[\tau_i \in \text{span}(\text{<think>, </think>}) \cup \text{span}(\text{<answer>, </answer>})]$.

We explicitly represent both the prediction of the current state and the next state in the reasoning. This simple world-modeling objective improves the agent’s grounding by helping it generate better next states and answers. During training, tokens corresponding to environment observations are masked in the loss. The resulting model then provides an effective initialization for downstream policy RL.

We emphasize that SPA is a *policy-centric* world-modeling objective, not a standalone action-conditioned simulator $p(s_{t+1} \mid s_t, a_t)$: the next-state prediction is emitted inside the same reasoning trace that produces the action, conditioned on

a prefix (grounded current state, often an implicit plan) generated by the collecting policy, so the learned predictive distribution is tied to that policy. We therefore use the internalized world model as a grounded *initialization* for RL rather than an inference-time simulator, and validate this in §5.1 and Appendix D.

3.3 RL Training

After obtaining a good world model by SFT, we do RL to boost the policy model. We use the Proximal Policy Optimization (PPO) (Schulman et al., 2017) algorithm for RL. Denote τ as trajectory, $u_i = \frac{\pi_\theta(\tau_i \mid \tau_{<i})}{\pi_{\text{old}}(\tau_i \mid \tau_{<i})}$ as the probability ratio between the current and old policies, and let $\tau_{<i}$ denote the prefix of token i . The PPO loss is defined as:

$$J^{\text{PPO}}(\theta) = \frac{1}{\sum_i M_i} \sum_i M_i \cdot \min(u_i A_i, \text{clip}(u_i, 1 - \varepsilon, 1 + \varepsilon) A_i),$$

where M_i is a mask that is 1 for answer tokens and 0 for contextual tokens, A_i is the per-token advantage and ε is a clipping hyperparameter. This objective is to optimize the policy to gain more rewards. Here we use our optimized s_t (described in Section 3.1) to represent the world states to better encode the world knowledge during RL training.

4 Explore-then-exploit outperforms reward shaping for world modeling

Experiment setting and baselines We conduct experiments on a diverse set of models of varying sizes, including Qwen2.5-0.5B/1.5B-Instruct, Qwen2.5-3B (Team, 2024) and LLaMA-3.2-1B-Instruct (Dubey et al., 2024). Our evaluation spans multiple environments like Sokoban, Frozen-Lake, and Sudoku, all of which are highly controllable. Our codebase is developed based on RAGEN (Wang et al., 2025b). We train with a batch size of 32 and collect eight rollouts per environment at each update step. For evaluation, we use both Pass@8 and Pass@1 metrics. We use VAGEN (Wang et al., 2025a) as our baseline. This method enables online world-model learning during RL by providing observation and prediction-based reward signals. We also evaluate a SOTA model: open-source GPT-OSS-20B (OpenAI, 2025).

Small models could outperform strong models on text games with SPA. Our main results are

shown in Table 2. For every model we evaluated, SPA consistently outperforms State Estimation RL, which in turn outperforms vanilla RL. E.g., for Qwen2.5-1.5B-Instruct, SPA could obtain 59.8 Pass@1 score after 1000 training steps for Sokoban environment, and 70.9 Pass@1 for FrozenLake environment, which even outperforms GPT-OSS-20B model. Across other models, Qwen2.5-0.5B-Instruct and Qwen2.5-3B, SPA delivers substantial improvements over baselines on all our evaluated environments. This is consistent with SPA restoring exploration—reflected in the recovered Pass@k—and consolidating environment knowledge in these settings. To probe whether SPA extends to more realistic settings, we also evaluate it on ALFWorld using the Qwen2.5-0.5B-Instruct model. We choose a smaller 0.5B model as we observe that the performance quickly saturates for the 1.5B model (in Figure 1). The results are in Table 10. We observe that SPA delivers consistent gains on both the Pass@1 and Pass@k metrics. This offers initial evidence that SPA extends beyond text games, though here we verify it on a single model and leave broader real-world validation to future work.

SPA outperforms the reward-based method VAGEN. VAGEN (Wang et al., 2025a) performs world modeling by eliciting current-state observations and predicting next states, and then applies a reward to encourage accurate grounding. For a fair comparison, we reimplemented VAGEN in the same state-estimation setting as SPA, following the original formulation closely (prompt format and observation/prediction reward supervision). In our reimplementation under these matched settings, VAGEN yields little improvement on our testbed, whereas SPA delivers substantial gains (e.g., 59.8/69.5 vs. 44.5/50.0 on Sokoban, Table 2). We attribute this, tentatively, to SPA confining world-model learning to the SFT stage and leaving the RL objective unperturbed, thereby avoiding the multi-objective interference of adding a world-modeling reward during RL.

5 What makes the Explore-then-Exploit World Modeling work?

We further disentangle why SPA succeeds, highlighting four critical factors: (i) *transition modeling*, (ii) *grounded state representations*, (iii) *a strong policy for world-model interaction*, and (iv) *sufficient world-modeling data*.

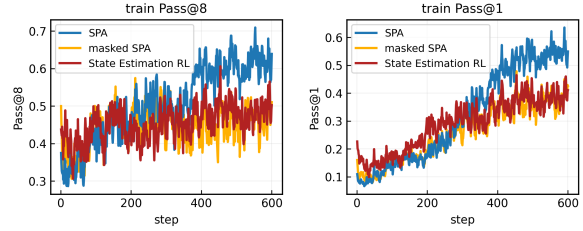


Figure 4: RL performance after SFT under three settings: State Estimation RL, No masking vs. Masking current states with next states for SPA.

Method	Pass@1	Pass@8
Vanilla RL	25.6	34.0
State Estimation RL	52.7	53.9
SPA	59.8	69.5
SPA (w/ self-belief SFT)	39.2	44.2

Table 3: We replace the ground-truth states in the self-experience SFT data with the model’s own self-belief states (i.e., without ground-truth supervision) and compare it with SPA. Relying on self-belief states degrades downstream RL performance.

5.1 Effect of Transition Modeling

Transition-model learning is key to RL scaling. Masking the SFT loss on the current and next states results in no improvement. To isolate the effect of world modeling supervision, we remove explicit transition (world modeling) annotations (*observation* and *prediction* spans) from the SFT corpus and replace them with a masking token *[MASKED]* and set the masking tokens’ loss to 0 during SFT. Starting from the base pretrained model, we run SFT on this masked data and use the resulting checkpoint to initialize the PPO stage. We show the results in Figure 4. In the masked setting, SFT yields no improvement for PPO. This suggests that PPO’s scaling benefits derive from transition-model learning, highlighting our world-model encoding stage as pivotal to the performance.

What happens if SFT relies solely on self-belief states rather than ground-truth states? To test whether ground-truth states are necessary, we train SFT using only the model’s self-belief states (i.e., without ground-truth supervision). This variant fails to improve, and in fact, degrades RL performance (Table 3), indicating that accurate world modeling during SFT, not merely adding a state-prediction signal, is necessary for downstream gains.

Without State Estimation, the world modeling SFT could also boost the performance. To ver-

Method	Pass@1	Pass@8
Vanilla RL	25.6	34.0
SPA w/o S Estimation	36.3	40.6

Table 4: SPA without the State Estimation and compare with the baselines.

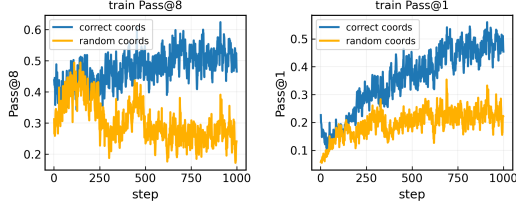


Figure 5: Pass@8/Pass@1 comparison by using correct and incorrect abstraction on Sokoban during RL training. Left: Abstraction PPO with correct coordinates. Right: randomized coordinates.

ify whether the world-modeling SFT also boosts RL performance without State Estimation, we use a similar observation-then-prediction prompt to generate self-experience data from raw symbolic states. We observe a performance gain, as shown in Table 4. Thus, SPA improves RL even without explicit state estimation, further validating its effect.

We present two further confound-controlled analyses in Appendix D. First, dropping the next-state prediction while keeping the free-form plan lowers Pass@1/Pass@8 from 36.3/40.6 to 25.6/34.0, so the prediction slot is not redundant with planning. Second, on-policy prediction accuracy rises from 31.9% to 46.6% during RL, so the policy refines rather than ignores its predictions.

5.2 Effect of Ground Truth state abstraction

Ground-truth coordinates are critical for RL training. To test whether the abstract state format alone is sufficient, we hold the text template and replace all ground-truth (x, y) coordinates (player/box/goal) with i.i.d. random coordinates drawn from the same grid at each step. This preserves the correct format but modifies the content. In Figure 5, with random coordinates, the training collapses: Pass@1 and Pass@8 stagnate at low levels with fluctuation, whereas with correct coordinates, both metrics improve steadily. It shows that coordinates provide essential supervision for spatial grounding and credit assignment; without them, the policy fails to align with actionable spatial relations, resulting in brittle exploration and poor generalization.

5.3 Effect of Policy Prior for Exploration

Method	Pass@1	Pass@8
Vanilla RL	25.6	34.0
State Estimation RL	52.7	53.9
SPA (1 epoch SFT)	29.2 $\downarrow 23.5$	52.7 $\downarrow 1.2$
SPA (5 epoch SFT)	59.8 $\uparrow 7.1$	69.5 $\uparrow 16.6$
SPA (1 epoch RandSFT)	0.1 $\downarrow 52.6$	0.4 $\downarrow 53.5$
SPA (5 epoch RandSFT)	20.2 $\downarrow 32.5$	50.0 $\downarrow 3.9$

Table 5: We replace the self-experience SFT data generator with a random-action generator and compare its performance with SPA.

Can the RL policy be replaced with random actions for generating world-modeling trajectories? We realize that world-modeling trajectories need not be collected exclusively with the learned policy; they can also be obtained via generic action generators. To ablate SPA, we replace its SFT data generator with a random-action generator: at each step, we sample uniformly from the action space Up, Down, Left, Right and substitute a fixed reasoning token sequence (“*I will push the box to the target.*”) As shown in Table 5, trajectories produced by this random policy yield substantially worse downstream RL performance, suggesting that a prior-aware policy for self-experience data collection is crucial for effective world-model training to empower the subsequent RL.

How to enable effective exploration when the policy model is weak? We find that many failures are due to instruction noncompliance during data generation: some models fail to follow the specified output format, preventing reliable synthesis of world-modeling samples. To quantify this effect, we conduct an ablation comparing performance before and after filtering out SFT samples that deviate from the required format. Building on this, we evaluate three settings under the same RL budget on the LLaMA-3.2-1B-Instruct model in Sokoban: **Original SFT**, distills world-state transitions via SFT; **Filtered SFT**, where the data samples are cleaned by enforcing minimal structural validity (`<observation>`-`<prediction>` reasoning followed by an `<answer>`); and **Abs PPO**, where PPO is trained on raw grids enriched by abstraction, without transition-model SFT. As in Figure 8, removing misaligned data improves training stability and performance (e.g., higher Pass@k and @1), indicating that format adherence is critical for learning accurate state and transition structure.

5.4 Effect of Extended SFT

To investigate the impact of spending more compute on world-modeling SFT, we vary the length

of SFT from 1 to 5 epochs, and perform RL on all 5 checkpoints under the same setup. Figure 6 shows the final performance after 1000 steps of RL training. As the world model is trained longer, downstream RL improves consistently: Pass@1 rises from 0.29 to 0.60 and Pass@k from 0.53 to 0.70. Efficiency also improves: the average number of actions per episode decreases from 8.47 to 6.44, while action effectiveness rebounds from an early dip at Epoch 2’s 0.60 to 0.77 by Epoch 5. Response length climbs from 65 tokens at Epoch 1 and then stabilizes in the 160 to 175 range for Epochs 2–5, indicating that gains are not caused by verbosity but by better spatial grounding and transition prediction.

6 Findings: Exploration, Exploitation, and Generalization

We now examine training dynamics and generalization. Multi-turn RL reveals a distinctive exploration-exploitation trajectory, and easy-to-hard SPA generalizes to higher task complexities within the same environment family, though it fails to transfer across games with fundamentally different dynamics.

Environment interaction expands reasoning frontiers. By enabling direct environment interaction, multi-turn RL broadens the reasoning frontier. Prior work (Yue et al., 2025) shows that RL with Verifiable Rewards (RLVR) improves sampling efficiency toward correct solutions but does not yield fundamentally new reasoning strategies. Moreover, RLVR exhibits a crossover in performance: RLVR-trained models outperform their base counterparts at small k (e.g., $k = 1$), while base models achieve higher Pass@k at large k . In contrast, our multi-turn setting displays no such reversal. As shown in Figure 7, Pass@k continues to increase in the early phase of training (steps 0–400), remaining high even at $k = 1024$, and only decreases during the later phase (steps 400–1000). This trajectory reflects an exploration-then-exploitation dynamic of the policy, highlighting the role of environment exploration and offering a plausible explanation for the discrepancy from math-domain RLVR results.

Beyond Pass@k. We complement Pass@k with direct trajectory metrics in Appendix D: we find that grounded SPA runs visit *fewer* total states than vanilla runs, yet cover far more success-relevant states with higher action diversity. Our claim is

Evaluation setting	Pass@1 (%)	Pass@k (%)
Simple → Complex (Sokoban)	0.9	3.1
Complex (Sokoban, baseline)	0.1	0.8
Sokoban → FrozenLake	15.9	49.2
FrozenLake (baseline)	17.2	49.2

Table 6: Results of cross-complexity and cross-game generalization experiments. The upper two rows are cross-complexity, and the lower two rows are cross-game generalization. We see that trained agents can generalize to harder environments but not to other environments.

therefore not “more exploration” in a naive coverage sense, but *more useful exploration*.

World-modeling enables easy-to-hard transfer gains. We investigate whether world modeling learned in a low-complexity setting can generalize to a more complex setting and thereby accelerate RL. Concretely, we train the world model with SFT on the simple FrozenLake environment (grid size 4×4) and then perform RL on the more challenging variant (6×6). In Figure 9, the proposed easy-to-hard transfer substantially outperforms all non-world-modeling baselines, yielding faster learning and higher asymptotic returns. Thus, learning simple world dynamics serves as a reusable prior that generalizes to more complex settings. We note that this is a *transfer analysis* rather than a same-budget comparison: easy-to-hard SPA receives extra pre-RL signal (SFT data from the easy variant) that the baselines do not use.

Generalization across games and complexity levels. We further examine whether the trained game agents could generalize beyond its training setting. Table 6 reports two evaluation protocols. In the *cross-complexity* setting, SPA is trained on a simple Sokoban variant (6×6 grid, one box) and tested on a more complex variant (10×10 grid, two boxes). SPA outperforms a baseline trained directly on the complex task, indicating that world-model SFT facilitates scaling to higher task complexity within the same environment family. In the *cross-game* setting, SPA trained on Sokoban is evaluated on FrozenLake environment, indicating that cross-game generalization is difficult.

7 Related Work

World modeling for LLM agents. LLM agents trained via prompting (Yao et al., 2023; Shinn et al., 2023) or RL pipelines (Song et al., 2024; Wang et al., 2025b) follow a short-horizon “think-

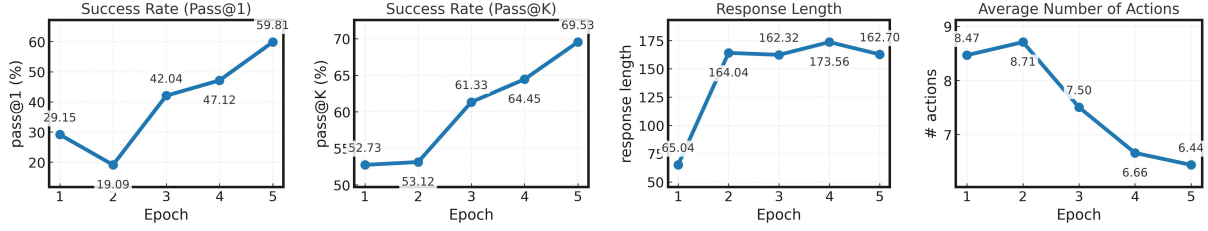


Figure 6: We evaluate RL training under initializations with varying SFT epochs and observe that increasing world-modeling SFT generally improves subsequent RL performance.

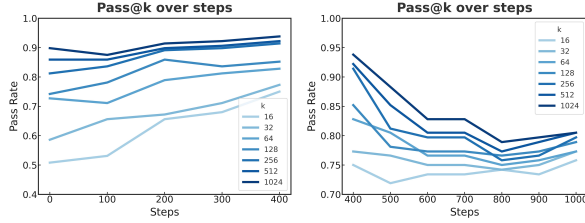


Figure 7: Pass@k performance on Sokoban during RL training. Left: steps 0–400, Pass@k consistently rises (exploration). Right: steps 400–1000, Pass@k declines as k increases (exploitation). Even at $k = 1024$, early exploration improves success rates.

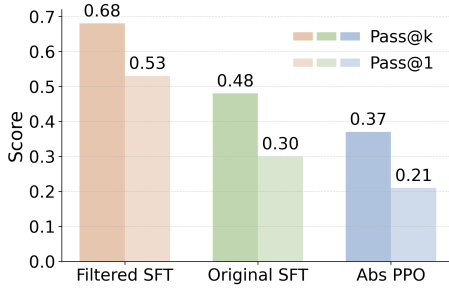


Figure 8: Results on Sokoban with LLaMA under three settings: Filtered SPA SFT, raw data SPA SFT and State Estimation PPO.

then-act” paradigm with no internal mechanism to persist or simulate environment dynamics, and degrade outside the pre-training distribution (Shao et al., 2019; Szot et al., 2023; Ning et al., 2025; Xing et al., 2025). Classical RL likewise shows the benefits of explicit world models (Janner et al., 2019; Yu et al., 2020); we internalize state and transition modeling into the policy as a grounded initialization. Closest to us, VAGEN (Wang et al., 2025a) injects world-modeling signals into online RL via reward shaping, whereas SPA confines world modeling to an SFT stage, leaving RL unperturbed.

Exploration and Pass@k dynamics in RL-trained LLMs. The Pass@1/Pass@k divergence has been discussed for single-turn RLVR: Yue et al. (2025) show that RLVR improves small- k success while narrowing the large- k boundary, Wen et al. (2025) argue Pass@k can be incomplete, and Zhai et al. (2026) analyze tool-use agents with a

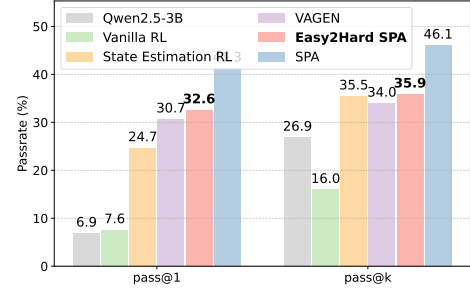


Figure 9: Easy-to-hard SPA on FrozenLake surpasses all baselines, trailing SPA trained on the target complexity.

Pass@(k, T) metric. We do not claim the divergence is new; rather, we characterize its *unfamiliar multi-turn* manifestation, give a grounding diagnosis with direct exploration measurements (§6), and mitigate it with a grounded initialization.

8 Conclusion

We identify *exploration collapse*—a decline in Pass@k under RL in environments whose states are unfamiliar to the policy—and trace it to weak grounding in environment states and dynamics. SPA addresses this with a simple explore-before-exploit recipe: a self-experience SFT stage that grounds the agent in state estimation and transition prediction, used to initialize standard RL. It consistently outperforms vanilla RL and a reward-shaping baseline across four model families, indicating that grounding the policy before reward optimization—not additional finetuning per se—restores exploration. We hope this phenomenon and simple remedy offer a reproducible starting point for agents in unfamiliar environments.

Limitations

Scope. Our main results are on grid-style interactive benchmarks (Sokoban, FrozenLake, Sudoku) with small- to mid-scale open-weight models; ALFWorld provides only a single-model check. Whether the same recipe transfers to larger models and richer, real-world agentic tasks (e.g., web navigation, tool use, embodied control) remains to be

verified.

Reliance on ground-truth states. SPA supervises the self-experience SFT stage with ground-truth states and coordinates exposed by the simulator, and our ablations indicate this supervision is necessary for the gains. Environments without accessible privileged state would first require estimating it (e.g., via a learned perception module), which we do not study here. We view this access as a matter of experimental scope rather than a property of the method: environment-provided or deterministically extracted structured states give a controlled setting for isolating whether stronger state/transition grounding improves RL under unfamiliar states, not a claim that privileged state is available in general deployments.

Evidence for the mechanism. We evidence restored exploration through the recovery of Pass@k, complemented by post-hoc trajectory metrics (successful-path coverage and action diversity; §6, Appendix D); a fully formalized account of the exploration–exploitation tradeoff (e.g., in terms of policy entropy or regret) and a finer-grained causal account of *why* transition-prediction SFT restores exploration are left to future work. Relatedly, the internalized world model is used as a grounded *initialization* for RL rather than an inference-time planner, so our claims concern initialization, not explicit planning.

Stability and comparisons. Training remains unstable in stochastic settings, and errors in the instruction/data-generation stage degrade the supervised transition data. Our VAGEN comparison is a reimplement under matched settings and may not reflect its best configuration. Future work includes uncertainty-aware transition modeling and scaling to richer modalities. Due to compute constraints, we report numbers based on single runs, but we repeat the experiments across different models, environments, and ablations. Future work could investigate the stability of SPA training as well. We do not foresee that our work carries risks that are worth mentioning here.

Acknowledgments

We thank Zihan Wang, the lead author of RAGEN, for numerous insightful suggestions on this project and for sharing technical details of the RAGEN codebase. We also thank Zichen Liu for helpful

suggestions on cross-game generalization evaluations, and Chang Ma, Wei Liu, Junteng Liu and Zhengyu Chen for valuable discussions. We also thank Shuaicheng Niu for pointing out some typos of the draft. YWT is supported by the Ministry of Digital Development and Information (MDDI) under the Singapore Global AI Visiting Professorship Program (Award No. AIVP-2024-002).

References

- Pengfei Cao, Tianyi Men, Wencan Liu, Jingwen Zhang, Xuzhao Li, Xixun Lin, Dianbo Sui, Yanan Cao, Kang Liu, and Jun Zhao. 2025. Large language models for planning: A comprehensive and systematic survey. *arXiv preprint arXiv:2505.19683*.
- Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, and 1 others. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.
- Michael Janner, Justin Fu, Marvin Zhang, and Sergey Levine. 2019. When to trust your model: Model-based policy optimization. *Advances in neural information processing systems*, 32.
- Bowen Jin, Hansi Zeng, Zhenrui Yue, Jinsung Yoon, Sercan Arik, Dong Wang, Hamed Zamani, and Jiawei Han. 2025. Search-r1: Training llms to reason and leverage search engines with reinforcement learning. *arXiv preprint arXiv:2503.09516*.
- Liangbo Ning, Ziran Liang, Zhuohang Jiang, Hao hao Qu, Yujuan Ding, Wenqi Fan, Xiao-yong Wei, Shanru Lin, Hui Liu, Philip S Yu, and 1 others. 2025. A survey of webagents: Towards next-generation ai agents for web automation with large foundation models. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V. 2*, pages 6140–6150.
- OpenAI. 2025. [gpt-oss-120b & gpt-oss-20b model card](#). Preprint, arXiv:2508.10925.
- Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. 2023. Toolformer: Language models can teach themselves to use tools. *Advances in Neural Information Processing Systems*, pages 68539–68551.
- John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.
- Kun Shao, Zhentao Tang, Yuanheng Zhu, Nannan Li, and Dongbin Zhao. 2019. A survey of deep reinforcement learning in video games. *arXiv preprint arXiv:1912.10944*.

Noah Shinn, Federico Cassano, Edward Berman, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. 2023. [Reflexion: Language agents with verbal reinforcement learning](#). *Preprint*, arXiv:2303.11366.

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

Andrew Szot, Max Schwarzer, Harsh Agrawal, Bogdan Mazouze, Walter Talbott, Katherine Metcalfe, Natalie Mackraz, Devon Hjelm, and Alexander Toshev. 2023. Large language models as generalizable policies for embodied tasks. *arXiv preprint arXiv:2310.17722*.

Qwen Team. 2024. [Qwen2.5: A party of foundation models](#).

Kangrui Wang, Pingyue Zhang, Zihan Wang, Yaning Gao, Linjie Li, Qineng Wang, Chi Wan, Hanyang Chen, Yiping Lu, Zhengyuan Yang, Lijuan Wang, Ranjay Krishna, Jiajun Wu, Li Fei-Fei, Yejin Choi, and Manling Li. 2025a. VAGEN: Reinforcing world model reasoning for multi-turn vlm agents. In *Advances in Neural Information Processing Systems*. NeurIPS.

Zihan Wang, Kangrui Wang, Qineng Wang, Pingyue Zhang, Linjie Li, Zhengyuan Yang, Xing Jin, Kefan Yu, Minh Nhat Nguyen, Licheng Liu, and 1 others. 2025b. Ragen: Understanding self-evolution in llm agents via multi-turn reinforcement learning. *arXiv preprint arXiv:2504.20073*.

Xumeng Wen, Zihan Liu, Shun Zheng, Shengyu Ye, Zhirong Wu, Yang Wang, Zhijian Xu, Xiao Liang, Junjie Li, Ziming Miao, Jiang Bian, and Mao Yang. 2025. Reinforcement learning with verifiable rewards implicitly incentivizes correct reasoning in base LLMs. *arXiv preprint arXiv:2506.14245*.

Jialong Wu, Baixuan Li, Runnan Fang, Wenbiao Yin, Liwen Zhang, Zhengwei Tao, Dingchu Zhang, Zekun Xi, Gang Fu, Yong Jiang, and 1 others. 2025. Webdancer: Towards autonomous information seeking agency. *arXiv preprint arXiv:2505.22648*.

Eric Xing, Minghai Deng, Jinyu Hou, and Zhiting Hu. 2025. [Critiques of world models](#). *Preprint*, arXiv:2507.05169.

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

Tianhe Yu, Garrett Thomas, Lantao Yu, Stefano Ermon, James Y Zou, Sergey Levine, Chelsea Finn, and Tengyu Ma. 2020. Mopo: Model-based offline policy optimization. *Advances in Neural Information Processing Systems*, 33:14129–14142.

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

Zhiyuan Zhai, Wenjing Yan, Xiaodan Shao, and Xin Wang. 2026. Does RL expand the capability boundary of LLM agents? A PASS@ (k,T) analysis. *arXiv preprint arXiv:2604.14877*.

A Prompts

Sokoban Prompt: Symbol with Coordinates

You are solving the Sokoban puzzle.
You are the player and you need to push all boxes to targets. You are provided with a symbol grid and the zero-indexed coordinates of the player, each box, and each target. Coordinates range from the top-left corner (0, 0) to the bottom-right corner (5, 5).
...
The meaning of each symbol in the state is:
#: wall, _: empty, 0: target, √: box on target, X: box, P: player, S: player on target.
Your available actions are: Up, Down, Left, Right
...
Turn 1:
State:

#_ _ _ 0#
#_ _ _ _#
###X_#
###P_#

Player (P) is at (4,3); box (X) is at (3,3); target (0) is at (1,4).
...

Sokoban Prompt: Base mode

You are solving the Sokoban puzzle.
You are the player and you need to push all boxes to targets.
You are provided with a symbol grid and the zero-indexed coordinates of the player, each box, and each target.
Coordinates range from the top-left corner (0, 0) to the bottom-right corner (5, 5).
When you are exactly next to a box, you can push it by moving in the same direction.
You cannot push a box through a wall, and you cannot pull a box.
The answer should be a sequence of actions, like <answer>Right || Right || Up</answer>.

Sokoban Prompt: Observation then prediction

You are solving the Sokoban puzzle.
You are the player and you need to push all boxes to targets.
You are provided with a symbol grid and the zero-indexed coordinates of the player, each box, and each target.

Coordinates range from the top-left corner (0, 0) to the bottom-right corner (5, 5). When you are exactly next to a box, you can push it by moving in the same direction. You cannot push a box through a wall, and you cannot pull a box. The answer should be a sequence of actions, like `<answer>Right || Right || Up</answer>`.

A sample full output is as follows:

```
<think>
<observation>
#####
#_####
#_P###
#_X#_#
#_0_#
#####
Player (P) is at (2,2); box (X) is at
(3,2); target (0) is at (4,3).
</observation>
1 Down - I push box to (4,2).
2 Left - I step to (3,1).
3 Down - I stand left of box, ready to push
it Right onto target.
<prediction>
#####
#_####
#_0_#
#_X#_#
#PX0_#
#####
</prediction>
</think>
<answer> Down || Left || Down </answer>
```

FrozenLake Prompt: Base mode

You are solving the FrozenLake puzzle. Forbid the whole and go to the target. You may move to the unintended direction due to the slippery ice. Example answer format:
`<think>To forbid the hole and go to the target, I should go left then go up.`
`</think>`
`<answer>Left || Up</answer>`

FrozenLake Prompt: Observation then prediction

You are solving the FrozenLake puzzle. Forbid the whole and go to the target. You may move to the unintended direction due to the slippery ice. Example answer format:
`<think>To forbid the hole and go to the target, I should go left then go up.`
`</think>`
`<answer>Left || Up</answer>`

A sample full output is as follows:

```
<think>
<observation>
_0__
0___
G___
__P_
```

```
</observation>
Player at (3,2); holes at (0,1) and (1,0);
goal at (2,0). 1 Up - move to safe ice
(2,2). 2 Left - slide to (2,1), adjacent to
goal. 3 Left - reach goal (2,0); player now
on G.
<prediction>
_0__
0___
√___
----
</prediction>
</think>
<answer> Up || Left || Left </answer>
```

Sudoku Prompt: Base mode

You are solving 4x4 Sudoku. Fill empty cells with digits 1-4. Use a 1-indexed grid (rows/cols start at 1). A move is exactly: row,col,value (three integers). In one turn you may output multiple moves, separated by ||. Only propose moves that keep the row, column, and 2x2 subgrid valid. Always output EXACTLY as:
`<think>[brief reasoning]</think>`
`<answer>[r,c,v || r,c,v ...]</answer>`
No extra text outside the two tags. Keep the response under 50 words. Example:
`<think>Row 1 has one empty cell -> place 1. Column 2 then needs 2.`
`</think>`
`<answer>1,3,1 || 3,2,2</answer>`

Sudoku Prompt: Observation then prediction

You are solving 4x4 Sudoku. Fill empty cells with digits 1-4. Use a 1-indexed grid (rows/cols start at 1). A move is exactly: row,col,value (three integers). In one turn you may output multiple moves, separated by ||. Only propose moves that keep the row, column, and 2x2 subgrid valid. Always output EXACTLY as:
`<think>[brief reasoning]</think>`
`<answer>[r,c,v || r,c,v ...]</answer>`
No extra text outside the two tags. Keep the response under 50 words.

An example output:

```
<think>
<observation>
| . . 1 4 | 1 4 . 3 | 4 2 . . | . 1 4 2
Empty positions to be filled are at (1,1),
(1,2), (2,3), (3,3), (3,4), (4,1)
</observation>
<prediction>
| 2 3 1 4 | 1 4 2 3 | 4 2 3 1 | . 1 4 2
Empty positions to be filled are at (4,1)
</prediction>
</think>
<answer> 1,1,2 || 1,2,3 || 2,3,2 || 3,3,3
```

```
|| 3,4,1 </answer>.
```

B Experimental Setup

These text-based game environments we evaluate are highly controllable; we adjust task difficulty by varying grid size to match model capacity. For Sokoban, we fix a 6×6 grid for all models. For FrozenLake, we use 4×4 for smaller models (Qwen2.5-0.5B-Instruct, Qwen2.5-1.5B-Instruct and LLaMA-3.2-1B-Instruct) and 6×6 for Qwen2.5-3B to avoid ceiling effects observed with the 4×4 setting. For Sudoku, we use a 4×4 grid with six empty cells to be filled.

Experiments run on up to $8 \times 80\text{G}$ NVIDIA H100. For training, we instruction-tune on 1280 samples using AdamW with learning rate 1×10^{-4} . We apply PPO initialized from the SFT checkpoint, using a learning rate of 1×10^{-6} for actor and 1×10^{-5} for critic. It takes about 12 hours on single GPU to train Qwen2.5-1.5B-Instruct for 1,000 steps. For evaluation, we report decoding-only results with temperature 1.0, top- p 1.0, and max new tokens 400. Our code, configuration files, and trained checkpoints are publicly available at <https://github.com/shiqichen17/SPA>.

C Full Results

Table 7 reports the complete results. We also include the performance before RL training (step = 0) for both settings: (i) State-Estimation PPO, evaluated with the State-Estimation prompt, and (ii) SPA immediately after the world-modeling SFT stage. At step 0, the State-Estimation prompt yields better performance than baseline, whereas the additional world-modeling SFT slightly lowers accuracy. However, after subsequent RL training, SPA quickly recovers and ultimately surpasses State-Estimation PPO across tasks, indicating that the structure learned during SFT provides a transferable inductive bias that RL can exploit for better generalization.

D Confound-Controlled Ablations and Direct Exploration Metrics

D.1 The prediction slot is not redundant with the plan

Free-form planning text inside the reasoning trace could, in principle, account for the gains attributed to next-state prediction. To isolate the two without confounds, we compare three

aligned variants on Sokoban (Qwen2.5-1.5B-Instruct, Pass@1/Pass@8, no state estimation): (i) *Vanilla RL*, whose reasoning contains a plan but no prediction; (ii) a *prediction-only* SPA variant with a grounded next-state span and action but no separate plan; and (iii) SPA w/o state estimation, which keeps both plan and prediction. As shown in Table 8, keeping the plan but dropping prediction (25.6/34.0) falls well below plan+prediction (36.3/40.6), so next-state prediction adds a substantial, separable gain even when a plan is present. Conversely, dropping the plan (28.9/37.3) also falls below plan+prediction yet still exceeds the plan-only baseline. The two slots are therefore complementary, and the takeaway does not reduce to “grounded observation plus chain-of-thought is enough”.

D.2 The policy refines, rather than ignores, its predictions during RL

If the model simply learned to ignore the `<prediction>` slot, its prediction accuracy would stagnate or decay during RL. We track the next-state matching accuracy of on-policy rollouts over the full SPA Sokoban run and find that it rises from 31.9% right after SFT to 44.4% by step 100 and 46.6% averaged over steps 301–500, then largely plateaus while task success keeps climbing (train Pass@1 rises from 12.9 to 30.8 over the same span, reaching 59.8 at final validation). The overall gain therefore reflects SFT initialization, early transition refinement, and later RL task optimization together—consistent with SPA acting as a grounded initialization recipe rather than an accurate online simulator.

D.3 Beyond Pass@k: measuring useful exploration directly

Pass@k is an indirect proxy for exploration: a change in Pass@k can reflect rollout correlation, sampling temperature, or solution diversity rather than coverage of meaningful states. We therefore complement it with post-hoc trajectory metrics. Let $\mathcal{U}(\pi)$ denote the set of unique states visited by rollouts of policy π , and $\mathcal{U}^+(\pi) \subseteq \mathcal{U}(\pi)$ those states lying on at least one successful trajectory; we define the *successful-path coverage ratio* (SPCR) $\rho(\pi) = |\mathcal{U}^+(\pi)|/|\mathcal{U}(\pi)|$. This makes our view of the exploration–exploitation tradeoff explicit: the terminal objective remains the expected return $J(\pi)$, exploration quantities such as coverage $|\mathcal{U}(\pi)|$ and action diversity are instrumental rather

Table 7: Results on Sokoban, FrozenLake, and Sudoku for different models. (Metrics in $\times 10^{-2}$).

Model	Sokoban		FrozenLake		Sudoku	
	Pass@1	Pass@8	Pass@1	Pass@8	Pass@1	Pass@8
Qwen2.5-0.5B-Instruct	5.5	25.8	8.1	32.8	0.0	0.0
PPO	16.9	35.9	22.8	30.1	0.0	0.0
+State Estimation	5.2	25.4	6.7	28.5	0.1	0.8
+State Estimation RL	20.4	38.7	24.2	26.6	4.5	14.1
+ SPA SFT	1.8	11.7	3.1	15.6	0.1	0.8
+ SPA	36.7	45.3	46.9	65.6	18.2	43.8
+VAGEN	33.3	44.9	25.8	40.6	0.1	0.8
Qwen2.5-1.5B-Instruct	16.3	47.3	17.2	49.2	1.6	11.3
PPO	25.6	34.0	22.1	30.7	0.0	0.0
+State Estimation	15.7	43.8	18.6	51.2	0.3	2.3
+State Estimation RL	52.7	53.9	27.6	34.8	39.1	72.3
+ SPA SFT	8.3	36.7	9.8	43.8	3.5	24.2
+ SPA	59.8	69.5	70.9	75.0	59.6	94.9
+VAGEN	44.5	50.0	37.7	43.0	0.0	0.0
Qwen2.5-3B	12.5	35.9	6.9	26.9	0.0	0.0
PPO	31.4	35.5	7.6	16.0	0.0	0.0
+State Estimation	9.9	39.3	6.9	26.6	0.1	0.8
+State Estimation RL	26.2	27.7	24.7	35.5	24.1	26.2
+ SPA SFT	6.0	29.7	1.6	10.5	0.3	2.7
+ SPA	49.7	58.2	41.3	46.1	69.9	89.8
+VAGEN	31.9	43.0	30.7	34.0	0.0	0.0
LLaMA3.2-1B-Instruct	8.3	33.2	10.2	41.0	0.0	0.0
PPO	21.2	39.8	10.8	24.6	0.1	1.2
+State Estimation	9.6	33.6	10.9	43.0	0.1	0.8
+State Estimation RL	31.6	44.1	19.3	29.7	45.0	71.1
+ SPA SFT	6.3	27.7	11.4	38.3	1.3	9.8
+ SPA	53.0	68.0	64.8	71.1	81.3	100
+VAGEN	47.4	50.8	24.5	29.3	0.1	0.8
GPT-OSS-20B	45.8	84.4	68.8	100	61.8	100
+State Estimation	55.1	89.1	73.3	100	68.1	100

Method (Sokoban)	Pass@1	Pass@8
Vanilla RL (plan, no prediction)	25.6	34.0
SPA prediction-only (no plan)	28.9	37.3
SPA w/o State Est. (plan + prediction)	36.3	40.6

Table 8: Confound-free decomposition of the reasoning slots on Sokoban (Qwen2.5-1.5B-Instruct, no state estimation). The prediction span is not redundant with free-form planning, and neither component alone recovers the full combination.

than terminal, and coverage helps only insofar as it concentrates on success-relevant states—high $\rho(\pi)$ at high $J(\pi)$, not maximal $|\mathcal{U}(\pi)|$.

Table 9 compares weaker (vanilla) runs against stronger grounded (SPA) runs: the grounded runs visit *fewer* total states (40 $\rightarrow$ 25 on Sokoban,

Environment	Success	SPCR ρ	Action div.
Sokoban	0.30 $\rightarrow$ 0.65	0.125 $\rightarrow$ 0.480	0.561 $\rightarrow$ 0.681
FrozenLake	0.30 $\rightarrow$ 0.75	0.281 $\rightarrow$ 0.783	0.586 $\rightarrow$ 0.699

Table 9: Direct trajectory metrics for weaker (vanilla) $\rightarrow$ stronger grounded (SPA) runs: success rate, successful-path coverage ratio ρ , and action diversity (mean pairwise edit distance between sampled action sequences).

32 $\rightarrow$ 23 on FrozenLake) yet cover far more success-relevant states and exhibit higher action diversity (mean pairwise edit distance between sampled action sequences). The trend is robust: among the 12 grounded FrozenLake final-step runs we located, 10/12 outperform the raw baseline in final success and 10/12 in SPCR. The precise claim is therefore not “more exploration” in a naive cover-

Table 10: Performance of Qwen2.5-0.5B-Instruct on Alfworld.

Method	Pass@1	Pass@8
Baseline	30.4	76.5
SPA	52.1	93.3

Table 11: Performance of Qwen2.5-1.5B-Instruct on Sokoban and FrozenLake (GRPO).

Method (GRPO)	Sokoban		FrozenLake	
	Pass@1	Pass@8	Pass@1	Pass@8
Baseline RL	0.0	0.0	0.0	0.0
State Estimation RL	40.5	45.9	31.3	34.8
+ SPA	42.2	52.5	74.1	80.6
Δ (Improvement)	+1.7 $\uparrow$	+6.6 $\uparrow$	+42.8 $\uparrow$	+45.8 $\uparrow$

age sense, but *more useful exploration*.

E Use of Large Language Models (LLMs)

During the preparation of this paper, we used large language models (LLMs) such as ChatGPT for assistance in polishing the writing and improving the clarity of figures. LLMs were not used for research ideation, experimental design, or the production of experimental results. All conceptual contributions, analyses, and conclusions are solely those of the authors. The authors take full responsibility for the content of this paper.

F SPA on Alfworld

To evaluate whether SPA can generalize to more realistic settings, we also evaluate SPA on Alfworld using the Qwen2.5-0.5B-Instruct model. We choose a smaller 0.5B model as we observe that the performance quickly saturates for the 1.5B model (in Figure 1). The results are in Table 10. We observe that SPA delivers consistent gains on both the Pass@1 and Pass@k metrics. This demonstrates that our method can be effective beyond text games.

G SPA on GRPO

To evaluate whether SPA can also work on other RL algorithms, we also evaluate SPA on GRPO using the Qwen2.5-1.5B-Instruct model. The results are in Table 11. We observe that SPA delivers consistent gains on both the Pass@1 and Pass@k metrics. This demonstrates that our method can be effective across the RL algorithms.