When in Doubt, Plan It Out: Committed Small Language Model Deliberation for Reactive Reinforcement Learning

Nathan Gavenski [*1] **Juarez Monteiro** [*2] **Francisco Galuppo** [2] **Adriano Veloso** [2] **Odinaldo Rodrigues** [1]

Abstract

Reinforcement Learning (RL) policies often degrade in unfamiliar environments because they lack explicit deliberation. We propose Plan, Align, Commit, Think (PACT), a hybrid architecture that combines a fast, reactive RL policy with a slow, deliberative Small Language Model (SLM) planner. PACT invokes the SLM asynchronously to generate and validate candidate action plans. Once a plan is verified through simulation as safe, feasible, and complete, it is executed directly, bypassing the RL policy without retraining or modifying it. Evaluated on three FrozenLake configurations of increasing difficulty, PACT outperforms all baselines while relying on a 2B-parameter SLM backbone, suggesting that deliberative planning and reactive execution are more powerful in concert than either is alone in these settings.

1. Introduction

Agentic systems are increasingly deployed in real-world applications where they must not only react to their immediate observations, but also plan sequences of actions to achieve long-horizon goals. Although reinforcement learning (RL) has made significant progress in training capable agents, standard model-free RL policies remain fundamentally *reactive*: they map states to actions without deliberation or lookahead, degrading systematically when deployed beyond training conditions.

This limitation has long been recognised: Kahneman's dual-process theory (2011) distinguishes *fast* thinking, automatic and reactive, from *slow* thinking, deliberate and goal-directed. Trained RL agents are natural fast thinkers: reliable within familiar territory, but incapable of the forward reasoning that novel situations demand. A similar limitation occurs within symbolic AI paradigms. Belief-Desire-Intention (BDI) systems (Bratman, 1999) introduced *committed plans* to enable faster, reactive behaviour, although they still require manually engineered planning models and struggle to scale to real-world complexity.

Language models (LMs) offer a compelling path forward: pre-trained on vast corpora, they encode transferable knowledge and commonsense reasoning capabilities that suggest a natural role as high-level planning assistants for reactive agents. However, LMs are known to struggle with multi-step planning on their own (Valmeekam et al., 2025; Armony et al., 2025), and large models are computationally expensive, making their direct use as planners impractical for low-latency deployment. Small LMs (SLMs) (Wang et al., 2025) provide a more practical trade-off: computationally efficient yet capable enough to propose *candidate action sequences* that, once verified, provide the structured guidance reactive agents cannot generate on their own.

We therefore propose Plan, Align, Commit, Think (PACT), a hybrid architecture that combines a fast component, a reactive RL policy, with a slow component, a deliberative SLM planner, drawing on dual-process theory and BDI-style commitment. PACT relies on the RL policy for efficient decision-making in familiar conditions, and invokes the SLM asynchronously when the agent's epistemic uncertainty exceeds a threshold, signalling that the current observation is outside its training distribution. Once a plan is generated and verified as safe, the RL policy is bypassed: the agent commits to the plan until completion, revision, or failure. This commitment mechanism, and the clean separation of reactive and deliberative control it affords, distinguishes PACT from approaches that treat LMs as step-level advisors (Monteiro et al., 2026; Ahn et al., 2022). Experiments on FrozenLake across three increasingly difficult configurations demonstrate that, without retraining, this hybrid architecture enables agents to handle situations that neither component can resolve on its own.

2. Background

Reinforcement learning is a problem in which an agent learns to maximise a reward signal by interacting with an

[*]Equal contribution [1]Department of Informatics, King's College London, London, United Kingdom [2]Kunumi Institute, Belo Horizonte, Brazil. Correspondence to: Nathan Gavenski <nathan.schneider_gavenski@kcl.ac.uk>, Juarez Monteiro <juarez@kunumi.com>.

Preprint. June 15, 2026.

environment. More formally, the agent aims to learn a policy $\pi(a \mid s) : S \times A \to [0, 1]$ that maps states $s \in S$ to a distribution over actions $a \in A$, by maximising the expected cumulative reward over time. We formulate the problem in this work as a contextual Markov Decision Process (CMDP), which is a tuple $M = \langle S, A, O, r, T, C, \phi \rangle$, where O is the observation space, r is the immediate reward function $r : O \times A \to \mathbb{R}$, T is the transition function $T : O \times A \to O$, C is the context space, and ϕ is the projection function $\phi : S \times C \to O$ that maps states and contexts to observations (Kirk et al., 2023). Note that in CMDPs, the agent observes the observation produced by ϕ, and the current context $c \in C$ remains the same in an episode. In other words, the policy perceives only one context c at a time in which all states $s \in S$ are projected into it $\{\phi(s, c) \mid s \in S\}$. We do not assume that the agent has access to the context either via input or that it can be inferred from the observations, since two different contexts $c_1, c_2 \in C$ may map two different states $s_1, s_2 \in S$ to the same observation $\phi(s_1, c_1) = \phi(s_2, c_2)$. This framing makes the generalisation challenge precise: because the training and evaluation contexts are disjoint, a purely reactive policy $\pi(a \mid o)$ operating on observations alone cannot anticipate the environment's behaviour in novel contexts, which is the gap PACT is designed to close.

Finally, we use uncertainty estimation to determine when the RL policy is unsure about the next action. Commonly, uncertainty in neural networks can be split into two categories: aleatoric, which stems from the inherent randomness in the data, and epistemic, which arises from the model's lack of knowledge about the data (Kendall & Gal, 2017). Since aleatoric uncertainty is irreducible, we focus on epistemic. To compute the epistemic uncertainty of an observation, we use the Monte Carlo Dropout method (Kendall & Gal, 2017), which approximates Bayesian inference by performing multiple stochastic forward passes through the network with dropout enabled. Thus, the epistemic uncertainty of an observation o, $\mathcal{U}(o)$, is defined as $\mathcal{U}(o) = \frac{1}{N} \sum_{i=1}^{N} \sum_a p_i(a \mid o) \log \frac{p_i(a|o)}{\bar{p}(a|o)}$, where $p_i(a \mid o)$ is the action distribution from the i-th forward pass with dropout, and $\bar{p}(a \mid o) = \frac{1}{N} \sum_{i=1}^{N} p_i(a \mid o)$ is the mean distribution (we omit π_θ from $p_i^{\pi_\theta}$ for simplicity); this requires the policy to produce explicit per-action likelihoods.

3. Plan, Align, Commit, Think

We propose Plan, Align, Commit, Think (PACT), a hybrid architecture that combines a fast component, a reactive pretrained RL policy, with a slow component, a deliberative SLM planner, drawing on dual-process theory and the recent advances in planning with LMs (Valmeekam et al., 2025). PACT uses the RL policy for efficient local decision-making, while invoking the SLM asynchronously to generate candidate long-horizon plans. Because SLMs remain unreliable as standalone multi-step planners (Monteiro et al., 2026), PACT decomposes the planning process into three modules: plan generation, plan verification, and agent alignment; unlike step-level approaches such as ASK (Monteiro et al., 2026), which query the SLM independently at each uncertain step with no forward plan or commitment. PACT further requires an approximate transition function for plan simulation and a task description sufficient for SLM prompting (see Appendix A).

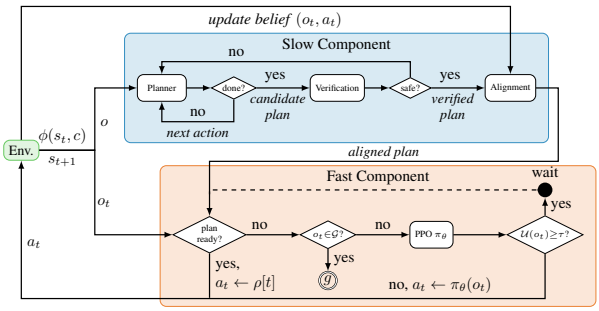

Figure 1. Plan, Align, Commit, Think architecture.

Process Overview Given an observation o, PACT routes it simultaneously to both components: the slow component initiates plan generation, verification, and alignment, while the fast component acts via the RL policy. PACT computes the uncertainty of the policy for every new observation o, $\mathcal{U}(o)$. If it is smaller than the threshold τ, the agent continues to apply the fast component until one of three conditions are met: (i) the uncertainty of the current observation o_t goes above the threshold τ; (ii) the slow component successfully generates a plan that is both verified and aligned with the agent's current trajectory; or (iii) the goal g is reached, thus, requiring no further action. If the uncertainty of the RL policy exceeds the uncertainty threshold for any observation, PACT waits for t time units for the plan generation, verification, and alignment completion. If the timeout expires, the RL component is reactivated. However, if the plan becomes ready before the agent reaches the goal, PACT bypasses the fast component and executes it; otherwise the process restarts. We note that PACT updates the alignment module's beliefs at each agent action. Thus, once an aligned plan is generated, no further alignment is necessary unless the agent encounters unexpected transitions (i.e., stochastic transition functions); plans generated from earlier observations are reconciled with the current position by the alignment module before committing.

Plan Generation Module This module generates candidate plans ρ with up to n actions. Initially, the SLM is given a description of the task and the episode's first observation o_1, and the slow component asks it for the action that takes it closest to the goal. The remaining actions $a_2, a_3, \ldots$ are generated by successively prompting the SLM until an observation indicates that the goal has been reached or the last

action a_n is generated. Due to the SLM's limited capabilities, it is important to include as much task information as possible in the prompts used for each action generation. For simplicity, we simulate action execution using a handcrafted transition function and update each prompt with the simulated observations. More sophisticated action simulation methods could be used instead, e.g., by learning a dynamics model from data (Gavenski et al., 2026). Crucially, the transition function does not need to be perfect: a heuristic approximation that captures the dominant dynamics suffices in this setting, since any residual mismatch between the simulated and actual environment is handled by the alignment module's replanning mechanism.

Plan Verification Module Candidate plans are simulated to assess their feasibility, safety and completeness. Firstly, the slow component checks that the plan is deployable given its action space and capabilities. Secondly, the slow component checks for any violation of safety constraints. If any constraint is violated, the slow component prompts the SLM to generate a new candidate plan that avoids the violation, and the process is repeated until both modules agree on a safe plan. Finally, although PACT works iteratively and can handle incomplete plans, candidate plans are checked for completeness to avoid unnecessary iterations, since the alignment process may be costly if the agent and the candidate plan differ significantly.

Agent Alignment Module PACT uses the SLM as an asynchronous planner, and the RL policy as a reactive executor. The alignment module bridges the gap between the idealised environment assumed during plan generation and verification and the agent's actual current state. If a discrepancy is found, PACT iteratively prompts the SLM, one action at a time, to guide the agent toward the closest reachable waypoint in the verified plan. Crucially, this process is *goal-directed*: the alignment module commits to a specific re-entry point in the original plan, and terminates as soon as the agent reaches it, after which committed execution of the verified plan resumes. This distinguishes the alignment module from step-level advisory approaches, which have no target waypoint and no termination criterion tied to plan re-entry. After alignment, PACT executes the remaining verified plan by bypassing the fast component and directly issuing actions in the environment.

4. Experimental Analysis

We evaluated PACT on FrozenLake (Towers et al., 2024) on three different configurations: i) a 6×6 map without slipperiness, for control over the episode termination; ii) a 6×6 map with slipperiness, to test the agent's ability to handle stochasticity; and iii) a 8×8 map, without slipperiness, to test the agent's ability to handle large state spaces with sparse rewards and long horizons. For the 'slow' component (SLM), we used a 2B model from Qwen Team (2026) as a zero-shot SLM backbone. The 'fast' component used Proximal Policy Optimisation (PPO) (Schulman et al., 2017) trained on 100 contexts not present in the evaluation and test splits. We compare PACT, PPO, and the standalone SLM with SAYCAN (Ahn et al., 2022) and ASK (Monteiro et al., 2026), which are state-of-the-art LM-RL approaches. All baselines were fine-tuned in 100 separate contexts. Full implementation details are in Appendix A–C; τ is selected on a held-out validation set (Appendix B).

Table 1 reports reward, episode length, and LM usage for all agents across the three configurations. PACT achieves the highest reward in every setting, from 0.98 ± 0.14 on the 6×6 map, to 0.93 ± 0.26 under stochasticity, to a perfect 1.00 ± 0.00 on the 8×8 map with zero variance, consistently outperforming the agents based on its individual components as well as the other LM–RL approaches.

The 6×6 slippery configuration isolates the effect of stochasticity on each method, with all baselines degrading substantially: PPO drops from 0.93 to 0.64, SAYCAN from 0.88 to 0.53, and ASK from 0.89 to 0.63. PACT's performance, by contrast, drops only by 5% (to a still respectable 0.93), remaining the top-performing method by a wide margin. Stochastic transitions frequently deviate the agent from its committed plan, leading to 2.85 replans per episode on average and a corresponding increase in LM usage to 58.4%. The most telling comparison is with SAYCAN, which consults the LM at a nearly identical rate (59.7%) yet achieves only 0.53 reward. This gap highlights the effect of plan commitment: both methods query the LM with similar frequency, but PACT uses those queries to generate, verify, and commit to structured trajectories, while SAYCAN issues independent step-level proposals with no higher-level structure to recover from transition deviation.

The 8×8 configuration evaluates a qualitatively different challenge: longer planning horizons, where PACT is the only method to achieve perfect reward, and does so with zero variance, a result that no baseline approaches. Despite performing well on the 6×6 map, PPO's performance drops to 0.79 as the horizon lengthens beyond what its reactive policy can reliably navigate. SAYCAN's falls to 0.62, performing worse than PPO alone, suggesting that step-level LM guidance without plan commitment becomes a liability rather than an asset at longer horizons. ASK's performance reaches 0.74, confirming that selective querying over PPO is beneficial, but the absence of plan-level commitment prevents it from closing the gap. The SLM's performance is only 0.42, despite 100% of its actions being issued through the LM, and with extreme length variance (35.4 ± 37.6), a clear sign of directionless wandering resulting from step-level generation without verification or commitment. PACT's LM usage increases to 81.2% while

Table 1. Results across environments for PACT and baselines.

	6×6			6×6 (With Slipperiness)			8×8		
Methods	Reward	Length	LM Usage	Reward	Length	LM Usage	Reward	Length	LM Usage
PACT	$\mathbf{0.98 \pm 0.14}$	10.0 ± 0.9	27.9%	$\mathbf{0.93 \pm 0.26}$	9.5 ± 1.9	58.4%	$\mathbf{1.00 \pm 0.00}$	15.5 ± 2.2	81.2%
PPO	0.93 ± 0.26	9.5 ± 1.9	–	0.64 ± 0.48	9.6 ± 4.2	–	0.79 ± 0.41	12.4 ± 3.4	–
SLM	0.47 ± 0.50	81.2 ± 66.7	100%	0.47 ± 0.50	15.5 ± 17.7	100%	0.42 ± 0.49	35.4 ± 37.6	100%
SAYCAN	0.88 ± 0.33	9.3 ± 1.9	60.1%	0.53 ± 0.50	9.1 ± 4.1	59.7%	0.62 ± 0.49	11.6 ± 3.6	73.1%
ASK	0.89 ± 0.31	14.4 ± 21.8	27.1%	0.63 ± 0.48	10.7 ± 5.3	26.7%	0.74 ± 0.44	13.4 ± 3.8	28.7%

replanning only 1.06 times per episode on average, suggesting that a single well-verified plan suffices to navigate the longer horizon reliably.

Taken together, PACT's LM usage (27.9% deterministic, 58.4% stochastic, 81.2% long-horizon) tracks the deliberative demands of each setting rather than being a fixed design choice. Crucially, across all settings the deciding factor is not how often the LM is consulted, but whether its outputs are structured, verified, and committed to execution.

5. Related Work

Integrating LMs into reinforcement learning has been explored as a means of improving generalisation beyond training conditions. Ahn et al. (2022) propose a system in which LM-proposed actions are scored by a learned affordance function and executed one step at a time, with no commitment to future actions and relying on large models unsuitable for low-latency deployment. Monteiro et al. (2026) (ASK) address the computational complexity by selectively querying the LM when the agent's epistemic uncertainty exceeds a certain threshold, improving OOD generalisation without the need for retraining. However, ASK still treats the LM as a step-level advisor, and thus cannot leverage the structured guidance that committed plans provide, leading to performance degradation as task difficulty increases.

The reliability of LMs as planners, however, remains limited. Huang et al. (2022) show that while large LMs can propose plausible action sequences from natural language descriptions, their outputs are not grounded in environment dynamics, thus failing in execution. Recent studies (Valmeekam et al., 2025; Armony et al., 2025) further demonstrate that even frontier models struggle to maintain consistency in long-horizon tasks requiring precise state tracking. PACT accounts for these limitations: rather than treating the SLM as an autonomous planner, it wraps plan generation in a simulation-grounded verification loop, ensuring that feasible, complete, and safe plans are committed to execution.

6. Conclusion and Limitations

In this work, we proposed Plan, Align, Commit, Think (PACT), a hybrid architecture that combines a fast, reactive RL policy with a slow, deliberative SLM planner through structured plan generation, simulation-grounded verification, and committed execution. Evaluated on FrozenLake across configurations of increasing difficulty, PACT outperforms all baselines while operating with a 2B-parameter LM, demonstrating that deliberative planning and reactive execution are more powerful in concert than either is alone. In particular, PACT's commitment mechanism provides robustness under stochasticity that step-level integration approaches cannot achieve, and its verification loop relaxes the model-scale requirement for effective integration.

Despite these encouraging results, PACT has limitations that motivate future work. First, the plan generation module relies on a hand-crafted transition function to simulate candidate plans; while this can be replaced by a learned dynamics model (Gavenski et al., 2026), the requirement for some form of transition function approximation limits applicability to environments where such a model is available or learnable. Second, PACT has been evaluated exclusively on direct-goal tasks rather than sub-goal decomposition, such as the "Door Key" environment from MiniGrid (Towers et al., 2024). However, we hypothesise that PACT's architecture is well-suited to such tasks, as the SLM can generate and verify plans that involve sub-goal decomposition, but empirical validation in this setting remains future work.

We believe PACT paves the way for a broader research agenda at the intersection of planning and SLMs: how to integrate already trained agents with smaller, more practical LMs without sacrificing performance or costly retraining. As the community continues to explore what LMs can contribute to planning, we hope PACT offers a concrete instantiation of the principle that structured commitment, not scale, is a key ingredient for reliable LM-guided behaviour in goal-directed sequential decision-making.

Acknowledgment

This work was partially supported by UK Research and Innovation [grant number EP/S023356/1], in the UKRI Centre for Doctoral Training in Safe and Trusted Artificial Intelligence, and by the Kunumi Institute, through individual grants awarded to the authors.

References

Ahn, M., Brohan, A., Brown, N., Chebotar, Y., Cortes, O., David, B., Finn, C., Fu, C., Gopalakrishnan, K., Hausman, K., Herzog, A., Ho, D., Hsu, J., Ibarz, J., Ichter, B., Irpan, A., Jang, E., Ruano, R. J., Jeffrey, K., Jesmonth, S., Joshi, N., Julian, R., Kalashnikov, D., Kuang, Y., Lee, K.-H., Levine, S., Lu, Y., Luu, L., Parada, C., Pastor, P., Quiambao, J., Rao, K., Rettinghouse, J., Reyes, D., Sermanet, P., Sievers, N., Tan, C., Toshev, A., Vanhoucke, V., Xia, F., Xiao, T., Xu, P., Xu, S., Yan, M., and Zeng, A. Do as i can and not as i say: Grounding language in robotic affordances. In *arXiv preprint arXiv:2204.01691*, 2022.

Armony, M., Meroño-Peñuela, A., and Canal, G. How far are LLMs from symbolic planners? An NLP-based perspective, 2025. arXiv:2508.01300.

Bratman, M. *Intention, plans, and practical reason*. Center for the Study of Language and Information, Stanford, Calif, 1999. ISBN 9781575861920.

Gavenski, N., Leonetti, M., and Rodrigues, O. *Towards Generalisable Imitation Learning Through Conditioned Transition Estimation and Online Behaviour Alignment*. January 2026. URL https://cyprusconferences.org/aamas2026/. 25th International Conference on Autonomous Agents and Multiagent Systems, AAMAS ; Conference date: 25-05-2026 Through 29-05-2026.

Huang, W., Abbeel, P., Pathak, D., and Mordatch, I. Language models as zero-shot planners: Extracting actionable knowledge for embodied agents. In *International conference on machine learning*, pp. 9118–9147. PMLR, 2022.

Kahneman, D. *Thinking, fast and slow*. Penguin, London, 2011. ISBN 9780141033570.

Kendall, A. and Gal, Y. What uncertainties do we need in Bayesian deep learning for computer vision? In *Advances in Neural Information Processing Systems 30 (NeurIPS)*, pp. 5574–5584, 2017.

Kirk, R., Zhang, A., Grefenstette, E., and Rocktäschel, T. A survey of zero-shot generalisation in deep reinforcement learning. *Journal of Artificial Intelligence Research*, 76: 201–264, 2023.

Monteiro, J., Gavenski, N., Zuin, G., and Veloso, A. When to ask: Uncertainty-gated language assistance for reinforcement learning. April 2026. URL https://attend.ieee.org/wcci-2026/. 2026 International Joint Conference on Neural Networks (IJCNN), IJCNN ; Conference date: 21-06-2026 Through 26-06-2026.

Qwen Team. Qwen3.5: Towards native multimodal agents, February 2026. URL https://qwen.ai/blog?id=qwen3.5.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Towers, M., Kwiatkowski, A., Terry, J., Balis, J. U., De Cola, G., Deleu, T., Goulão, M., Kallinteris, A., Krimmel, M., KG, A., et al. Gymnasium: A standard interface for reinforcement learning environments. *arXiv preprint arXiv:2407.17032*, 2024.

Valmeekam, K., Stechly, K., Gundawar, A., and Kambhampati, S. A systematic evaluation of the planning and scheduling abilities of the reasoning model o1. *Transactions on Machine Learning Research*, 2025. ISSN 2835-8856. URL https://openreview.net/forum?id=FkKBxp0FhR.

Wang, F., Zhang, Z., Zhang, X., Wu, Z., Mo, T., Lu, Q., Wang, W., Li, R., Xu, J., Tang, X., He, Q., Ma, Y., Huang, M., and Wang, S. A comprehensive survey of small language models in the era of large language models: Techniques, enhancements, applications, collaboration with llms, and trustworthiness. *ACM Trans. Intell. Syst. Technol.*, 16(6), November 2025. ISSN 2157-6904. doi: 10.1145/3768165. URL https://doi.org/10.1145/3768165.

A. Plan, Align, Commit, Think Architecture and Prompts

"The three Plan, Align, Commit, Think modules share the same SLM backbone, but differ in their prompts and the information they receive from the environment. Results in Table 1 show the performance of PACT with a Qwen3.5–2B. For the RL policy, we use a PPO to train an Actor-Critic architecture presented in Table 2 with a dropout rate of 0.2.

Table 2. PPO Actor-Critic Network Architecture (FrozenLake 6×6).

Branch	Layer (Type)	Input	Output
Input	Flatten Extractor	72	72
Policy Net (π)	Linear	72	64
	Dropout ($p = 0.2$)	64	64
	Tanh	64	64
	Linear	64	64
	Dropout ($p = 0.2$)	64	64
	Tanh	64	64
Value Net (vf)	Linear	72	64
	Dropout ($p = 0.2$)	64	64
	Tanh	64	64
	Linear	64	64
	Dropout ($p = 0.2$)	64	64
	Tanh	64	64
Action Head	Linear	64	4
Value Head	Linear	64	1

A.1. Planner Prompt

As displayed in Figure 1, the planner module receives a task description (Prompt 1 Lines 2–7) and the observation (Prompt 1 Lines 12–20). If prompted by the environment, the prompt has the first observation o_1, and if by the verification or alignment module, the last observation before breaking any constraints or alignment failure, respectively. Moreover, the planner module receives information indicating which actions are safe and which are unsafe. However, it has the freedom to choose any possible actions from the environment (even those that are invalid, such as moving to an edge).

Listing 1 Prompt for planner module for PACT

```
<|im_start|>system
    You are a robot navigation agent on a grid.
    You receive the full grid map, your position, the goal position, and your move history.
    You must choose exactly one action: UP, DOWN, LEFT, or RIGHT.
    You must NEVER choose an action listed under UNSAFE actions.
    Reply with one word only.
    No explanation.
<|im_end|>
<|im_start|>user
    You are a navigation agent on an 8x8 grid.

    Grid (A=Agent, G=Goal, H=Hole, F=Free, S=Start, #=visited multiple times):
    A H F F F F F H
    F F F F F F F F
    F F H F F F H F
    F F F F F F F H
    H H H F F F F F
    F F F H F F H H
    F F F F F F F F
    F F F F F F H G

    Agent position : row=0, col=0
    Goal position  : row=7, col=7

    SAFE actions   : DOWN
    INVALID actions : UP(edge), LEFT(edge), RIGHT(hole) - do NOT choose these
    This is your first move.
    If there are other moves which are safe but not in your last moves list, prefer these.
    Choose one of the SAFE actions that moves you toward the goal while avoiding holes.
    If no safe action moves closer, choose any safe action to go around obstacles.

    Reply with exactly ONE of the following words: UP, DOWN, LEFT, RIGHT.
<|im_end|>
```

We note that during the slipperiness configuration, the hand-crafted transition function simulates a deterministic environment, and thus the planner module receives the same observation for the same action, even if the environment is stochastic. PACT delegates the responsibility of handling stochasticity to the alignment module, which can replan when the environment deviates from the verified plan.

A.2. Verification Prompt

The verification module has two options: (i) try to fix the candidate plan by generating a new one based on the feedback from the verification process, or (ii) ask the planner module to generate a new candidate plan providing some feedback. Prompt 2 shows an example of the prompt for the verification module. Lines 2–6 describe the task and the expected output, while Lines 9–27 provide the grid, the candidate plan, and the violations that need to be fixed.

Listing 2 Prompt for verification module for PACT

```
1  <|im_start|>system
2      You are a safety verification agent.
3      You will be given a navigation plan that has been found to be unsafe, along with a description of the violations.
4      Your task is to produce a corrected plan that avoids all Holes and stays within the grid bounds.
5      Respond only with the corrected action list, e.g. [RIGHT, DOWN, RIGHT, DOWN].
6      Do not explain.
7  <|im_end|>
8  <|im_start|>user
9      The following plan is UNSAFE or INCOMPLETE.
10
11     Grid:
12     S F F F H F F F
13     F H F F H F H F
14     F H F H F F F F
15     F F F F F H H F
16     F F H F F H F H
17     F F F F F F F F
18     F F F F F F F F
19     H F F F F H H G
20
21     Plan: ['DOWN', 'DOWN', 'DOWN', 'RIGHT', 'RIGHT', 'DOWN', 'RIGHT', 'RIGHT', 'RIGHT', 'RIGHT', 'RIGHT', 'DOWN']
22
23     Violations:
24       - Step 5: action DOWN leads to (4, 2) which is a hazard(H).
25     Additionally, the plan does not reach the goal.
26
27     Provide a corrected plan from S to G avoiding all hazard cells (H). Output ONLY a bracket list: [ACTION, ACTION, ...]
28 <|im_end|>
```

If it fails to provide a safe plan, it will prompt the planner module with the last observation before the violation and the feedback from the verification process, and ask it to generate a new candidate plan using Prompt 1. This process is repeated until both modules agree on a plan that is safe, complete, and feasible.

A.3. Alignment Prompt

The alignment module tries to reconcile the agent's current position with the candidate plan since the planner module operates under the belief the agent never moved, and the RL policy operates reactively without awareness of the candidate plan. Prompt 3 shows an example of the prompt for the alignment module. Lines 2–7 describe the task and the expected output, while Lines 10–33 provide the grid, the agent's actual position, the expected position, the goal, the safe and unsafe actions, and the intended plan.

B. Uncertainty Threshold

The epistemic uncertainty threshold τ controls when the fast component pauses and waits for a plan from the slow component. We select τ using a held-out validation set of contexts that is disjoint from both the 100 training contexts and the test contexts used to report results, avoiding any leakage between threshold selection and final evaluation. To set τ, we compute the mean epistemic uncertainty $\mathcal{U}(o)$ of the fast component across all observations in this validation set, keeping τ fixed across all three map configurations. The average uncertainty was ≈ 0.09, corresponding to observations where the RL policy is operating within its trained distribution. We set $\tau = 0.1$, slightly above this average, so that the slow component is only invoked when the fast component's uncertainty is above its typical operating level, indicating a genuinely unfamiliar observation rather than normal decision-making noise.

C. Baseline Adaptations

ASK (Monteiro et al., 2026) was originally evaluated on FrozenLake and is used here without domain adaptation, replacing only the language model backbone with the same Qwen3.5–2B used by PACT. SAYCAN (Ahn et al., 2022) was adapted to FrozenLake by replacing its robotic affordance scoring with the PPO value head, which scores each LM-proposed action by its estimated future return before execution.

Listing 3 Prompt for alignment module for PACT

```
<|im_start|>system
  You are an alignment agent.
  A navigation agent has diverged from its intended trajectory.
  You will be given the original plan, the agent's current position, and where the agent was expected to be.
  Your task is to recommend the single best next action to safely guide the agent toward the goal, taking both the intended plan and the agent's actual position into account.
  Output format: respond with exactly ONE word. The word must be one of: UP, DOWN, LEFT, RIGHT.
  No other text. No explanation. No punctuation. Just the single action word.
<|im_end|>
<|im_start|>user
    Navigation agent needs the single best next action.

    Grid (A=Agent, E=Expected position, G=Goal, H=Hole, S=Start):
    S H A F F F F H
    F F F E F F F F
    F F H F F F H F
    F F F F F F F H
    H H H F F F F F
    F F F H F F H H
    F F F F F F F F
    F F F F F F H G

    Agent actual position  : row=0, col=2
    Expected position      : row=1, col=3
    Goal                   : (7,7)

    SAFE actions   : DOWN, RIGHT
    UNSAFE actions : UP(out of bounds), LEFT(hole) | do NOT choose these

    Intended plan  : ['DOWN', 'RIGHT', 'RIGHT', 'RIGHT', 'RIGHT', 'RIGHT', 'DOWN', 'DOWN', 'DOWN', 'DOWN', 'DOWN', 'RIGHT', 'RIGHT', 'DOWN']
    Remaining steps: ['RIGHT', 'RIGHT', 'DOWN', 'DOWN', 'DOWN', 'DOWN', 'DOWN', 'RIGHT', 'RIGHT', 'DOWN']

    Choose one of the SAFE actions that moves closer to the goal while staying aligned with the intended plan.
    Reply with exactly ONE word: the action name.
<|im_end|>
```