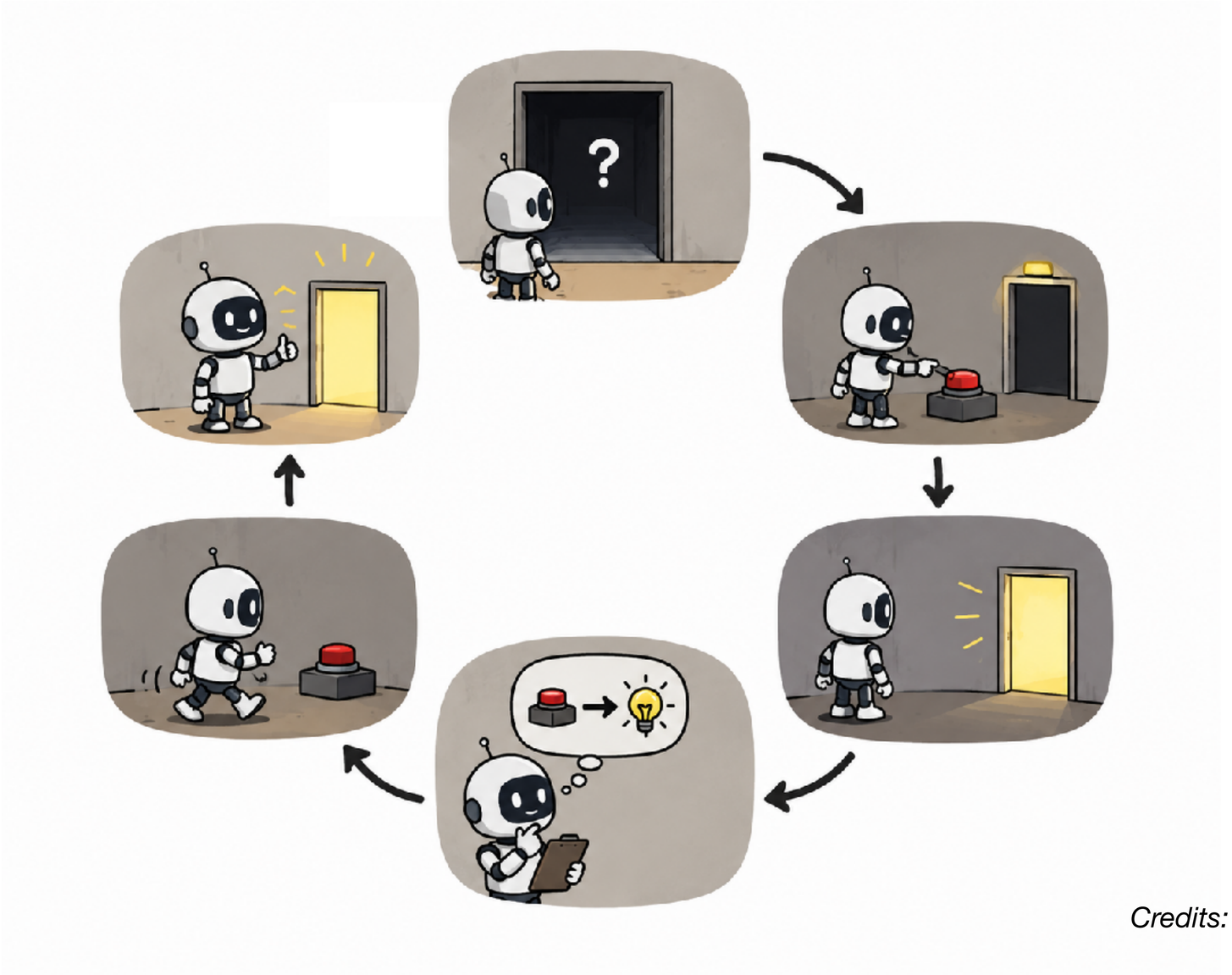
Learning POMDP World Models from Observations with Language-Model Priors

V. Six, F. Panse, L. Da Costa, M. Fajeau, M. Sharma, A. Amayuelas, T. Xiao, D. Hyland, P. Hennig, B. Schölkopf

Learning in the unknown



Credits: GPT-5.5

What are POMDPs ?

- Sequential decision-making when we can’t see true state of the world.
- It models transitions, observations, rewards, starting state.

What are POMDPs ?

- Sequential decision-making when we can’t see true state of the world.
- It models transitions, observations, rewards, starting state.

- Non-Markovian observations, so we keep a belief over states:

$$Q'(s') \propto O(s', a, o) \sum_{s \in \mathcal{S}} T(s, a, s') Q(s)$$

New guess = how well this observation fits each possible state, times your old guess

Language Model as Prior

- Classical method for learning POMPDs : tabular methods, theory-based RL…

  - Need lots of environment interaction

  - Hand-built hypothesis space per benchmark

Language Model as Prior

- Classical method for learning POMPDs : tabular methods, theory-based RL…

  - Need lots of environment interaction

  - Hand-built hypothesis space per benchmark

- LLMs have priors about the world (doors, keys, lava…)
- Using LLMs to generate executable code (WorldCoder, POMDP-Coder, CWQ)

Taking a peek at the hidden state

	WorldCoder	POMDP-Coder
POMDP Learning	Access to hidden states	Access to hidden states
Planning	Access to hidden states	No

Taking a peek at the hidden state

	WorldCoder	POMDP-Coder
POMDP Learning	Access to hidden states	Access to hidden states
Planning	Access to hidden states	No

“Post-hoc full observability assumption”

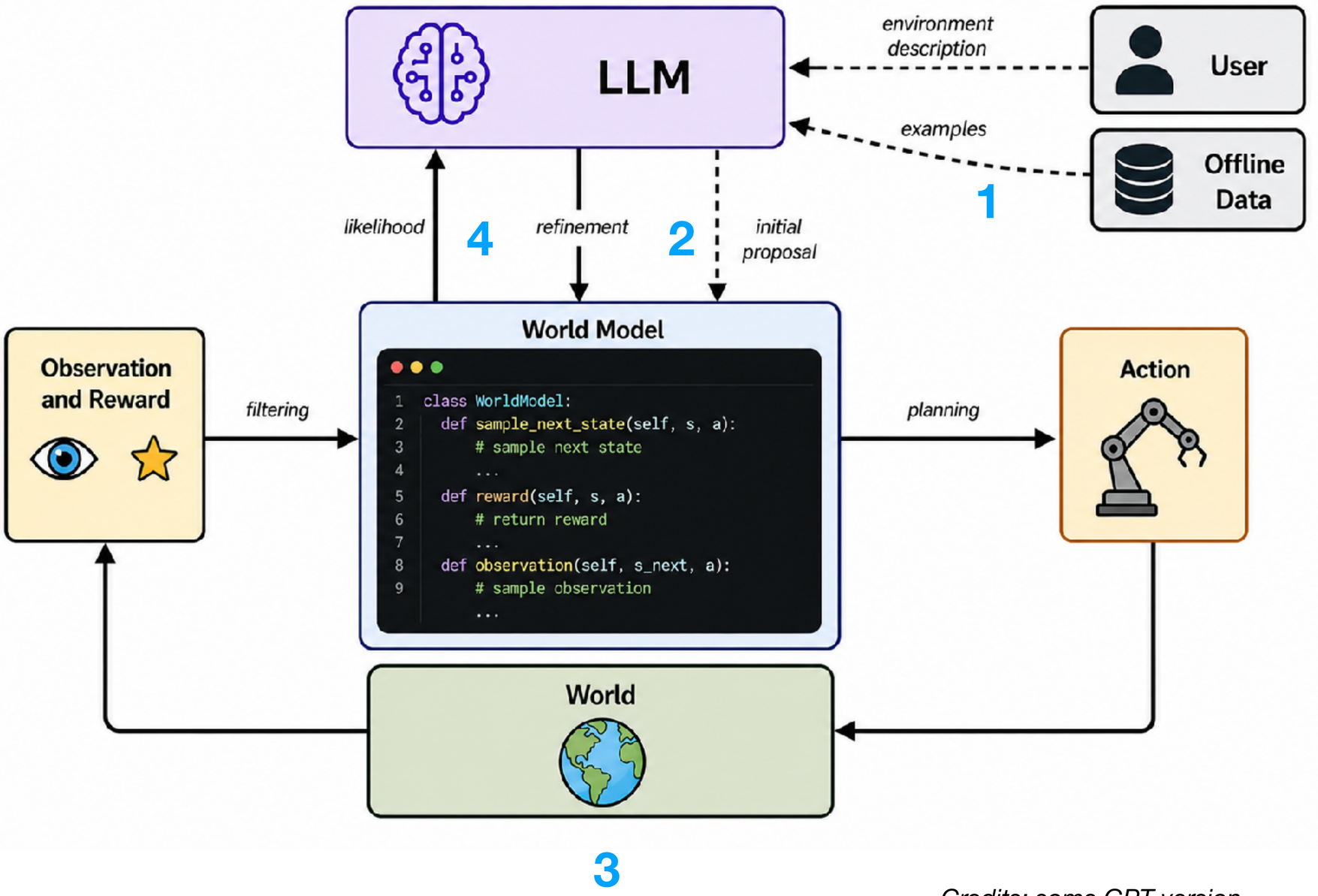
= cheating ?

What if we can’t peek ?

- Their assumption is not realistic for real world usage
- Much harder to learn what’s going on when we don’t cheat
- This is the case that we study: **can we learn efficiently under partial observability (PO) ?** How does performance drop ?

Trick 1 **Trick 2**

Method



Credits: some GPT version

Evaluating the models under PO

- We need a way to score how good the proposed models are
- Previous work: compare **predicted state** to **true hidden state**

Evaluating the models under PO

- We need a way to score how good the proposed models are
- Previous work: compare **predicted state** to **true hidden state**

- Instead we work with observations
- We compare **predicted observation** to **true observation**

How to update our belief ?

- We use Particle Filtering on observations to:

  - update our belief (planning)

  - score the models (learning)

Trick 1

$$\mathcal{L}(P^m; \mathcal{D}) = \sum_{\tau \in \mathcal{D}} \sum_{t=0}^{H-1} \mathbb{E}_{s_{t+1} \sim Q_{t+1}^{m,\tau}} \left[\log O^m(o_{t+1}^\tau \mid s_{t+1}, a_t^\tau) \right]$$

Score = “how well does the model's *own* belief predict what comes next?”

How to (smoothly) update our belief ?

- Problem: deterministic observation function = no useful likelihood
- We use distance kernel between observations ("*pseudo-likelihood*")

Trick 2

$$\mathcal{L}(P^m; \mathcal{D}) = \sum_{\tau \in \mathcal{D}} \sum_{t=0}^{H-1} \mathbb{E}_{s_{t+1} \sim Q_{t+1}^{m,\tau}} \left[\log O^m(o_{t+1}^\tau \mid s_{t+1}, a_t^\tau) \right]$$

How to (smoothly) update our belief ?

- Problem: deterministic observation function = no useful likelihood
- We use distance kernel between observations ("*pseudo-likelihood*")

Trick 2

$$\mathcal{L}(P^m; \mathcal{D}) = \sum_{\tau \in \mathcal{D}} \sum_{t=0}^{H-1} \mathbb{E}_{s_{t+1} \sim Q_{t+1}^{m,\tau}} \left[\log O^m(o_{t+1}^\tau \mid s_{t+1}, a_t^\tau) \right]$$

$$\mathcal{L}(P^m; \mathcal{D}) = -\frac{1}{\kappa} \sum_{\tau \in \mathcal{D}} \sum_{t=0}^{H-1} \mathbb{E}_{s_{t+1} \sim Q_{t+1}^{m,\tau}} \left[d_{\text{obs}}(O_{\text{LLM}}^m(s_{t+1}, a_t^\tau), o_{t+1}^\tau) \right]$$

Guiding the Refinement

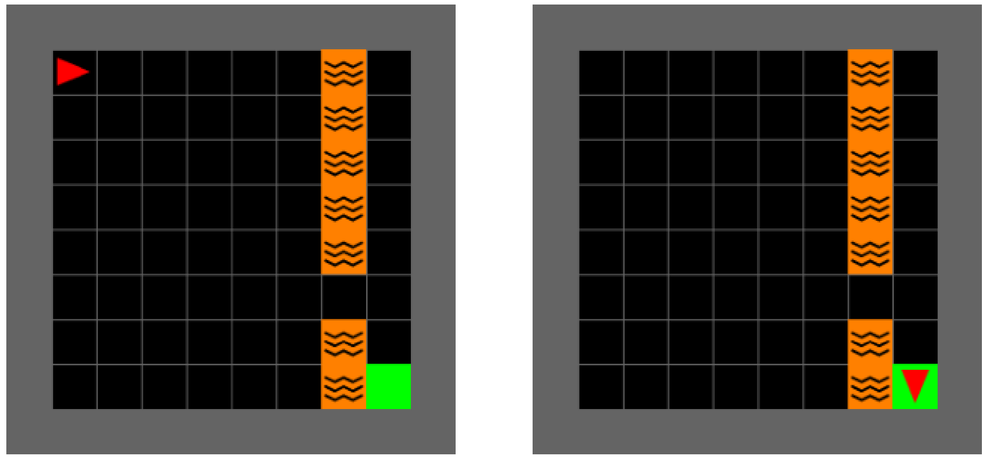
- A **single score** is not enough to guide refinement
- Additional feedback:

  - **Concrete failure cases** (reward mismatches, large prediction errors…)

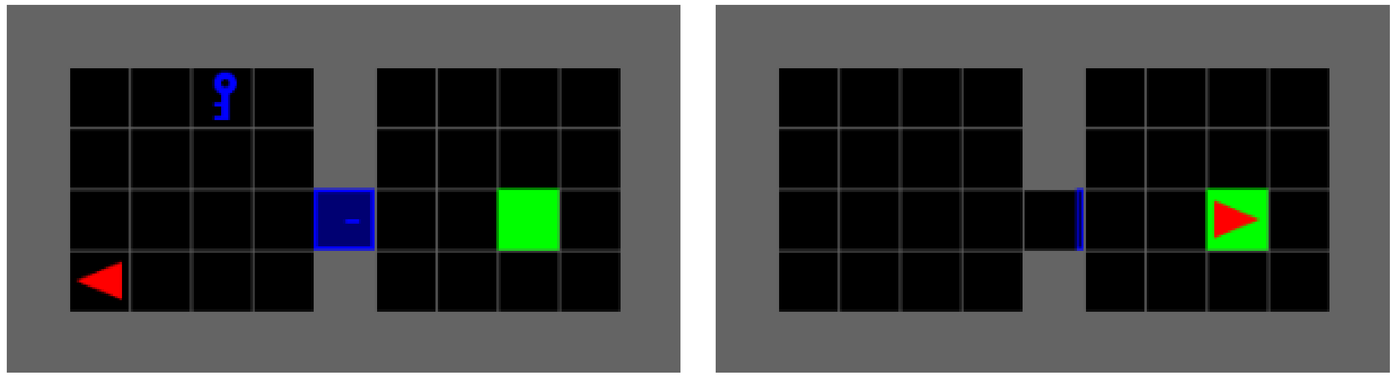
  - **Committee disagreement flags** (where different candidates differ)

- Exploiting best solution vs. Keeping enough diversity (AlphaEvolve inspired)

MiniGrid Benchmarking

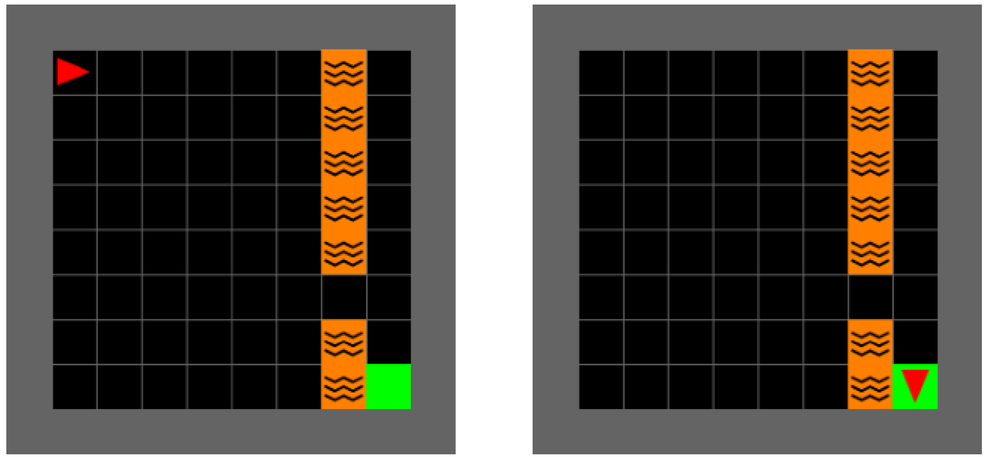


Lava

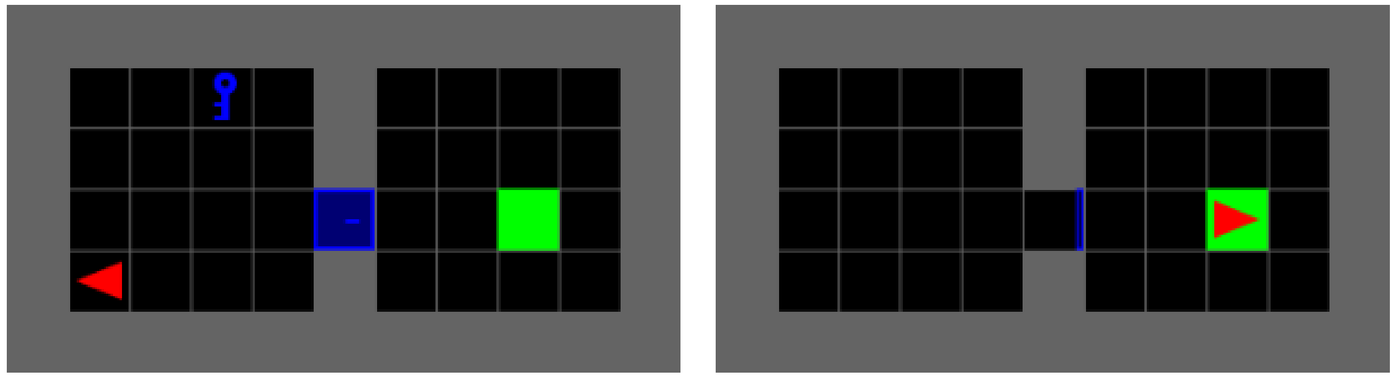


Unlock

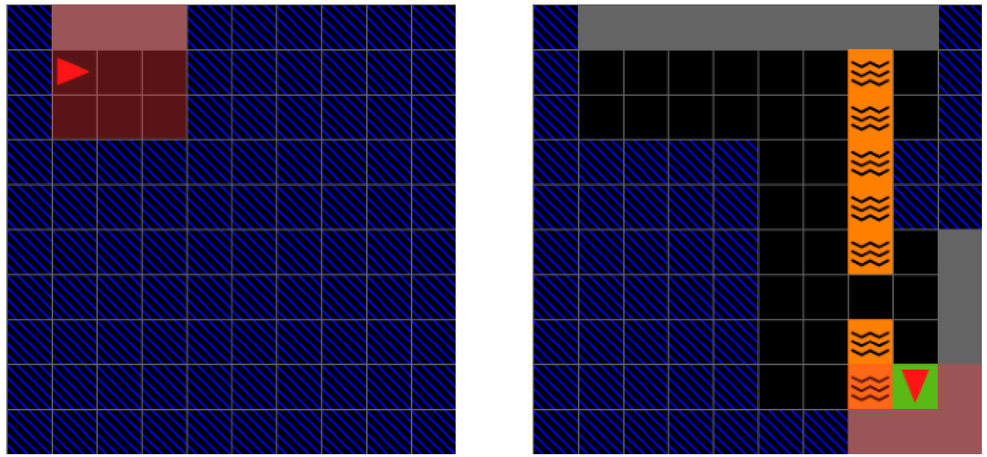
MiniGrid Benchmarking

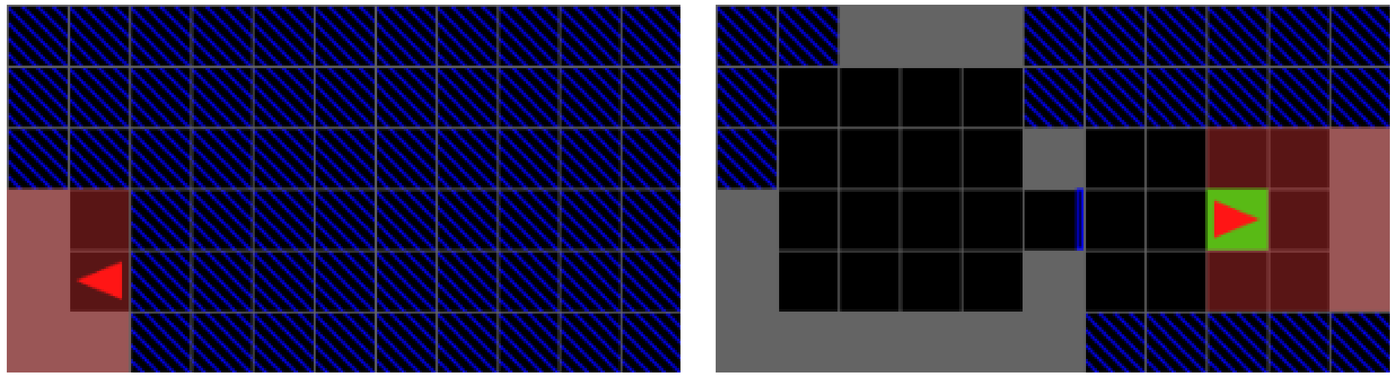


Lava

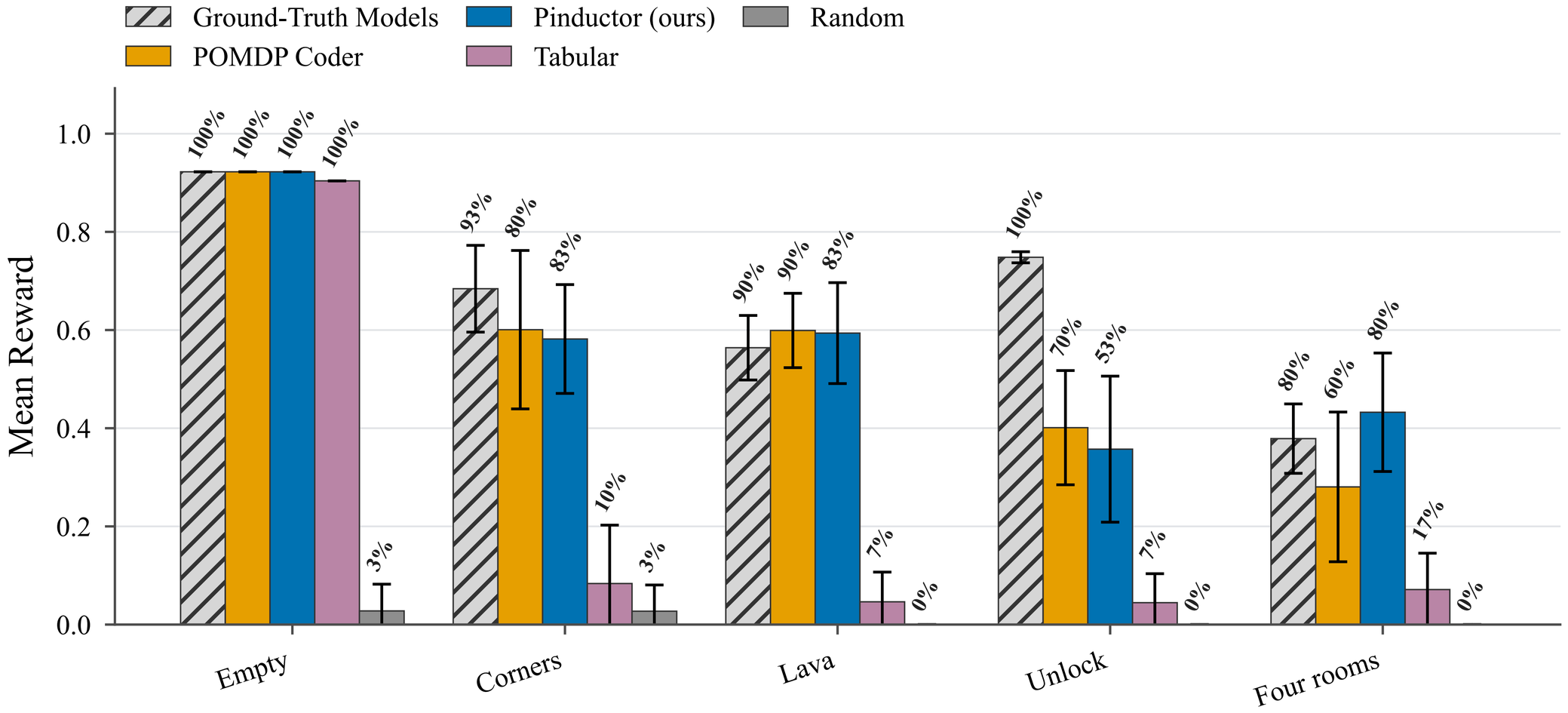


Unlock



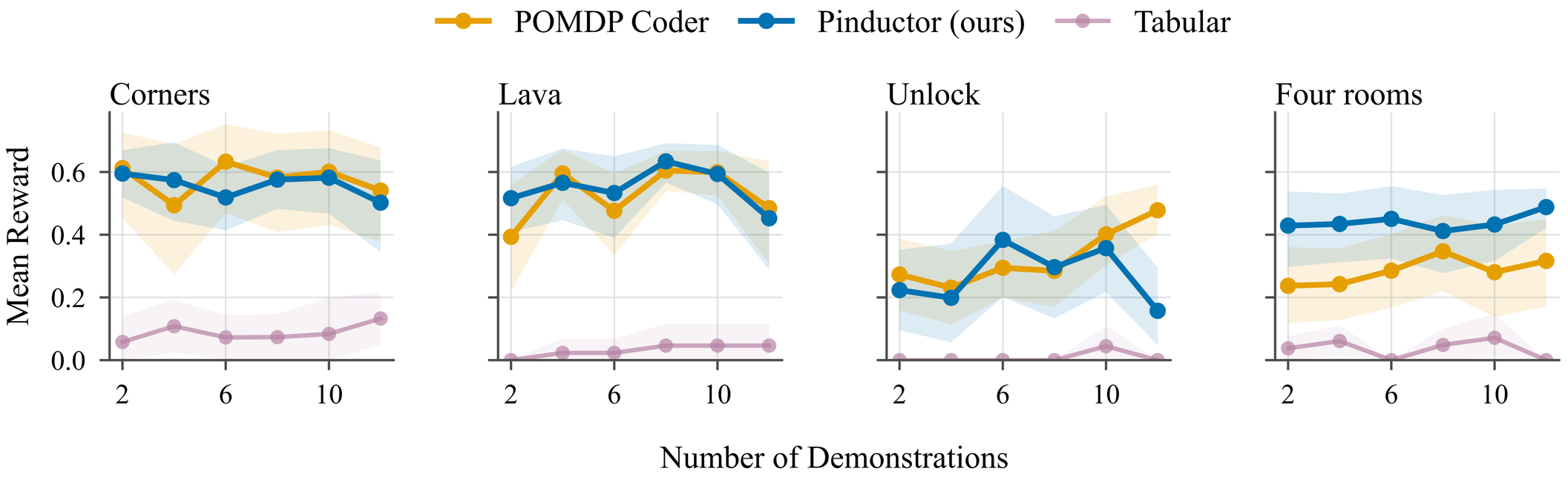


PO vs. Post-hoc full observability



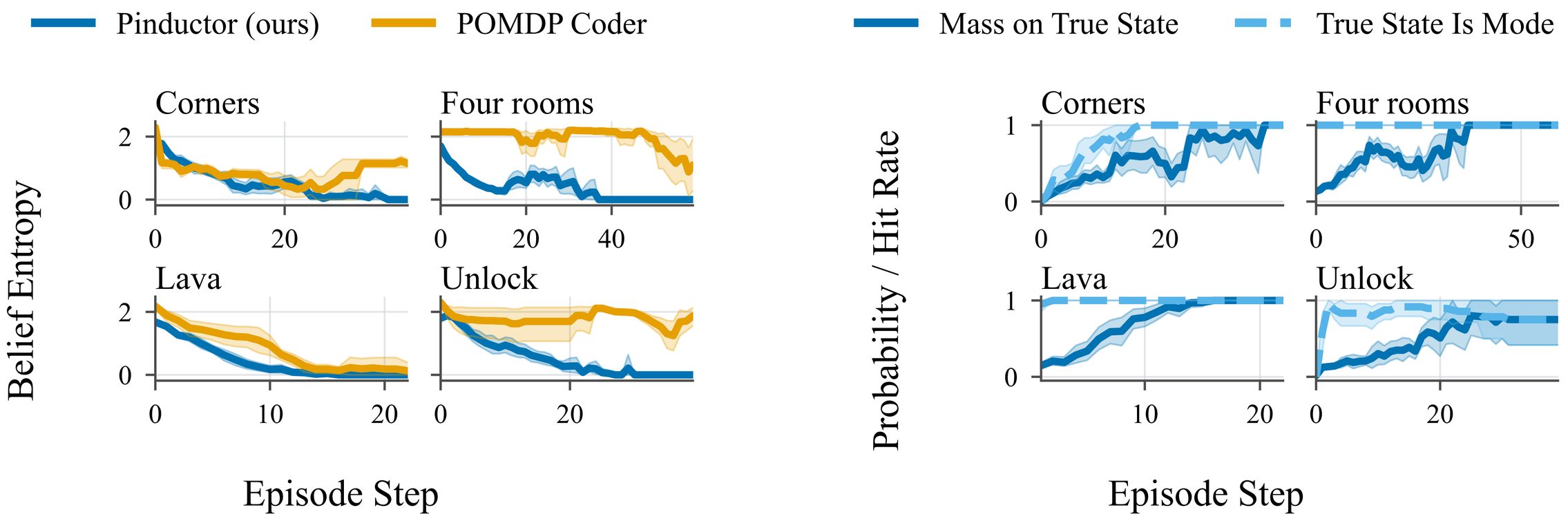
Despite seeing strictly less, we match POMDP-Coder and beat the tabular baseline

Sample efficiency of LM-guided discovery



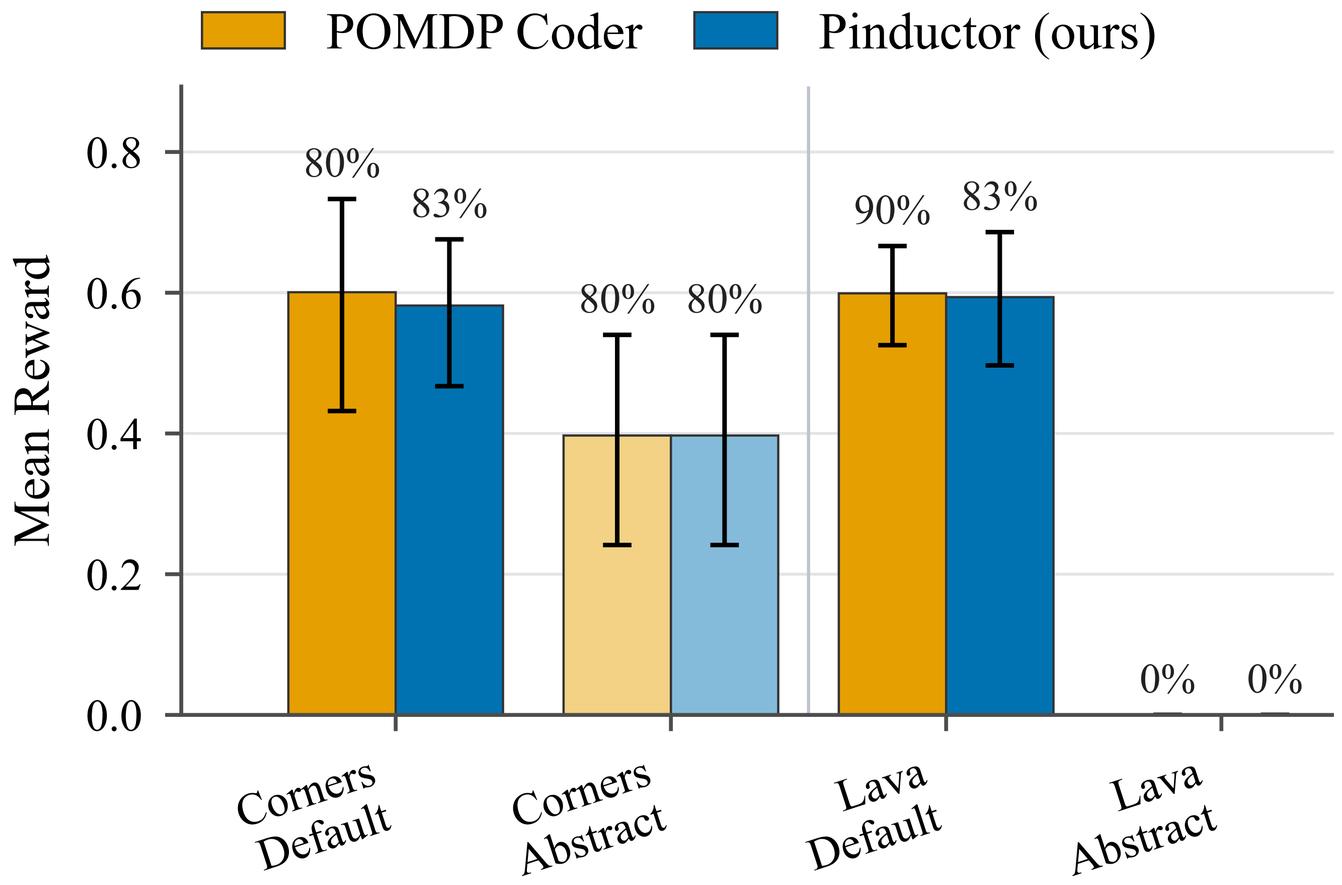
Sample efficiency is the biggest advantage of the method

Belief converges (nicely)



Pseudo-likelihood helps particle filtering

Fun semantic ablation



LLM struggles to abstract from semantics

Limitations

- Test navigation tasks in MiniGrid environments
- We need more RL baselines to benchmark against
- Generalisation / transfer-learning is hard
- High variance between runs, hard to control LLM consistency
- Online refinement is hard to guide / not very efficient ; early convergence
- Still providing instructions / info about the environment

What are we currently doing?

- More benchmarking in varied (non-grid) environments ; more competitive baselines (RL)
- Adding heuristics for more efficient planning
- Can the LLM infer the size of the state space (“we have a 20x20 grid”) ?
- Can this method do transfer-learning / generalisation ?
- Can we handle continuous observations ?

Feedback is welcome !