CRUX: Counterfactual Multi-Hypothesis Interpretation for Robust Embodied Agents

Md Muntaqim Meherab
Daffodil International University
Dhaka, Bangladesh
meherab2305101354@diu.edu.bd

Abstract

Embodied environments are a standard way to probe the reasoning abilities of large language models (LLMs) and other foundation models. Most systems follow the same pattern: a language front-end interprets the instruction, a subgoal module decomposes it, a planner proposes an action sequence, and a world model predicts the consequences. Yet performance is often reported as a single task-level success rate, which hides where the agent fails and how brittle it is under noise.

We take a modular view. We write down a formal model of a four-stage embodied pipeline (goal interpretation, subgoal decomposition, planning, and world modeling), and show that, under a mild monotonicity assumption, the overall task failure probability is bounded by the sum of the error rates of the individual modules. We refine this picture for tree-structured subgoal decompositions and show how decomposition errors scale with the number of leaves.

On top of this framework we build CRUX, a variant that treats interpretation and decomposition as *multi-hypothesis* problems. Rather than committing to a single interpretation of the language instruction, CRUX maintains a small set of candidate formal tasks and decompositions, plans for each, and uses a world model to run counterfactual rollouts under execution noise. The candidate with the highest estimated robust success is then selected and executed. We prove a finite-sample guarantee for this selection rule: with a reasonable number of rollouts, the robust performance of the chosen candidate is close to that of the best candidate in the set.

To ground the theory, we implement a compact CRUX prototype in a stochastic gridworld. In this setting, both CRUX and a single-interpretation baseline achieve perfect success in a clean environment, but under action noise and random start perturbations CRUX consistently attains higher robust success. In a moderate noise regime, the 95% confidence intervals for robust success do not overlap between CRUX and the baseline.

1 Introduction

Embodied simulation has become a standard way to study long-horizon reasoning and control. Systems built on simulators such as VirtualHome [Puig et al., 2018] and LTL-based motion planners [Fainekos et al., 2009] typically expose a multi-stage pipeline:

1. a natural-language goal is interpreted into a formal specification;
2. this specification is decomposed into subgoals or skills;
3. a planner chooses an action sequence or policy; and
4. a world model predicts how these actions play out in the environment.

39th Conference on Neural Information Processing Systems (NeurIPS 2025) Workshop: Foundation Models for Embodied Agents.

The NeurIPS "Foundation Models Meet Embodied Agents" challenge makes this structure explicit, with separate modules and metrics for goal interpretation, subgoal decomposition, action sequencing, and transition modeling.

Despite this modular design, performance is usually summarized by a single overall success rate. That makes it hard to answer basic questions such as: *Is the agent misunderstanding the instruction? Are the subgoals misaligned with the environment? Is the planner brittle? Is the world model hallucinating dynamics?* It also leaves us without a clear way to talk about robustness, even though embodied execution is noisy by nature: sensors drift, dynamics are stochastic, and controllers are not perfectly deterministic.

We aim for a cleaner picture. We formalize a four-module embodied pipeline in a way that is simple but explicit. Within this model, we show that if making any module "better" (in a precise sense) cannot hurt overall performance, then the task failure probability is bounded by the sum of the error rates of the individual modules. This is essentially a union bound, but stated in a way that matches how people already talk about these systems. We then look at tree-structured decompositions, which show up in hierarchical reinforcement learning and options [Dietterich, 2000, Sutton et al., 1999, Andreas et al., 2017], and show that in this case the contribution from decomposition errors scales with the number of leaves in the subgoal tree.

On top of this formalism, we propose CRUX. The core idea is simple: treat goal interpretation and subgoal decomposition as *choices among multiple hypotheses* rather than as one-shot decisions. Given an instruction, CRUX constructs a small set of candidate formal tasks and corresponding decompositions. For each one it plans a policy, runs a batch of counterfactual rollouts in a world model with execution noise, and estimates its robust success probability. It then selects the candidate with the highest estimated robustness and executes that policy in the real environment. Under standard concentration assumptions, the robust performance of the chosen candidate is within a small, explicitly controlled gap of the best candidate in the set.

To see how this behaves in practice, we implement CRUX in a simple stochastic gridworld. The environment is intentionally small and the front-end is a heuristic rather than an LLM, which lets us isolate the effect of multi-hypothesis selection under noise. We compare CRUX to a baseline that shares the same planner and world model but uses a single deterministic interpretation of the instruction. In a clean regime, both agents solve all tasks. Under action slip and start perturbations, CRUX consistently achieves higher robust success. In a moderate-noise configuration, the estimated robust success of CRUX and the baseline have non-overlapping 95% confidence intervals.

2 A simple model of modular embodied agents

We now write down an abstract model of the modular pipeline described above.

2.1 Environment and task specification

We work with an episodic Markov decision process (MDP)

$$\mathcal{M} = (S, A, P, \mu_0, \gamma), \tag{1}$$

where S is a finite state space, A is a finite action space, $P(s' \mid s, a)$ is a transition kernel, μ_0 is an initial-state distribution, and $\gamma \in (0, 1]$ is a discount factor. A finite trajectory of length T is

$$\tau = (s_0, a_0, s_1, a_1, \ldots, s_T). \tag{2}$$

High-level tasks are described in natural language by strings ℓ drawn from a language space $\mathcal{L}$. Internally, the agent works with linear temporal logic (LTL) specifications [Fainekos et al., 2009] over a set of atomic propositions Π. We write Φ for the space of such LTL formulas. Each state $s \in S$ is labeled with a subset of propositions via $L : S \to 2^{\Pi}$. Given a trajectory τ, satisfaction of $\varphi \in \Phi$ is written $\tau \models \varphi$ in the usual way.

For a given instruction ℓ there is an (unobserved) "true" LTL task $\varphi^\star \in \Phi$ that corresponds to what the human actually wants. We say a trajectory is successful if it satisfies this true task and define

$$R(\tau, \varphi^\star) \triangleq \mathbb{I}\{\tau \models \varphi^\star\}. \tag{3}$$

Our focus is on the probability of success, not on cumulative discounted reward.

2.2 Four modules

We model the agent as a composition of four abstract modules. The same structure applies whether each module is implemented with an LLM, a neural network, or a hand-written program.

Goal interpretation. The goal interpretation module maps the natural-language instruction to a formal task:

$$M_{\text{goal}} : \mathcal{L} \to \Phi, \qquad \hat{\varphi} = M_{\text{goal}}(\ell). \tag{4}$$

Subgoal decomposition. Given a formal task, the subgoal module produces a sequence (or tree) of subgoals:

$$M_{\text{sub}} : \Phi \to \Psi^K, \qquad (\hat{\psi}_1, \ldots, \hat{\psi}_K) = M_{\text{sub}}(\hat{\varphi}), \tag{5}$$

where Ψ is a space of subgoal specifications and K may depend on $\hat{\varphi}$.

World model. The world model uses interaction data to construct a planning model:

$$M_{\text{world}} : \mathcal{D} \to \mathcal{M}_{\text{model}}, \qquad \hat{\mathcal{M}} = M_{\text{world}}(\mathcal{D}), \tag{6}$$

where $\hat{\mathcal{M}}$ is typically a learned dynamics model or a simplified MDP built from data [e.g. Hafner et al., 2019].

Planner. Finally, the planner produces a policy given the subgoals and the world model:

$$M_{\text{plan}} : \Psi^K \times \mathcal{M}_{\text{model}} \to \Pi_{\mathcal{M}}, \qquad \pi = M_{\text{plan}}((\hat{\psi}_k)_k, \hat{\mathcal{M}}), \tag{7}$$

where $\Pi_{\mathcal{M}}$ is a class of policies for the true environment.

Putting everything together, given ℓ and data $\mathcal{D}$ the agent computes

$$\hat{\varphi} = M_{\text{goal}}(\ell), \tag{8}$$

$$(\hat{\psi}_1, \ldots, \hat{\psi}_K) = M_{\text{sub}}(\hat{\varphi}), \tag{9}$$

$$\hat{\mathcal{M}} = M_{\text{world}}(\mathcal{D}), \tag{10}$$

$$\pi = M_{\text{plan}}((\hat{\psi}_k)_k, \hat{\mathcal{M}}), \tag{11}$$

and then executes π in $\mathcal{M}$, producing a trajectory τ and success indicator $R(\tau, \varphi^\star)$.

2.3 Ideal modules and error events

To reason about where things go wrong, it is useful to introduce idealized versions of the four modules:

$$M^\star_{\text{goal}}, \quad M^\star_{\text{sub}}, \quad M^\star_{\text{world}}, \quad M^\star_{\text{plan}}. \tag{12}$$

These are conceptual; they need not correspond to any implementable algorithm. Intuitively, $M^\star_{\text{goal}}$ always recovers $\varphi^\star$, $M^\star_{\text{sub}}$ produces a decomposition compatible with optimal control, $M^\star_{\text{world}}$ returns the true environment $\mathcal{M}$, and $M^\star_{\text{plan}}$ is optimal for that decomposition and environment.

For a single episode, we define error events for each module:

$$E_{\text{goal}} \triangleq \{\hat{\varphi} \not\equiv \varphi^\star\}, \tag{13}$$

$$E_{\text{sub}} \triangleq \{(\hat{\psi}_k)_k \text{ is not equivalent to } (\psi^\star_k)_k\}, \tag{14}$$

$$E_{\text{world}} \triangleq \{\hat{\mathcal{M}} \text{ differs from } \mathcal{M} \text{ in a way that changes the optimal value of } \varphi^\star\}, \tag{15}$$

$$E_{\text{plan}} \triangleq \{\pi \text{ is suboptimal for } (\hat{\psi}_k)_k \text{ on } \hat{\mathcal{M}}\}, \tag{16}$$

where $(\psi^\star_k)_k = M^\star_{\text{sub}}(\varphi^\star)$. We do not fix a particular equivalence relation; one natural choice is equivalence with respect to the value of $\varphi^\star$.

We write

$$\epsilon_{\text{goal}} \triangleq \mathbb{P}(E_{\text{goal}}), \quad \epsilon_{\text{sub}} \triangleq \mathbb{P}(E_{\text{sub}}), \quad \epsilon_{\text{world}} \triangleq \mathbb{P}(E_{\text{world}}), \quad \epsilon_{\text{plan}} \triangleq \mathbb{P}(E_{\text{plan}}) \tag{17}$$

for the module-wise error probabilities, with the probability taken over all randomness in the environment and the agent. We also define the task-level failure event

$$E_{\text{fail}} \triangleq \{R(\tau, \varphi^\star) = 0\}, \tag{18}$$

with failure probability

$$\epsilon_{\text{task}} \triangleq \mathbb{P}(E_{\text{fail}}). \tag{19}$$

3 Theoretical analysis

We now relate task-level failure to module-wise errors and then turn to the selection step at the heart of CRUX.

3.1 A modular error bound

We consider a family of "hybrid" agents in which some modules are ideal and others are not. For a subset $S \subseteq \{\text{goal}, \text{sub}, \text{world}, \text{plan}\}$, let A^S be the agent that uses $M_i^\star$ for all $i \in S$ and M_i for all $i \notin S$. Let $\epsilon_{\text{task}}(S)$ be its task failure probability.

We assume a natural monotonicity property.

Assumption 1 (Monotonicity)**.** *For all $S \subseteq T \subseteq \{goal, sub, world, plan\}$,*

$$\epsilon_{task}(T) \leq \epsilon_{task}(S). \tag{20}$$

In words: replacing any subset of modules by their ideal versions cannot increase the task failure probability.

This holds whenever the ideal modules are defined to be optimizers (for planning and world modeling) or perfect labelers (for interpretation and decomposition).

Under this assumption we can bound ϵ_{task} in terms of the module-wise errors.

Theorem 1 (Modular union bound)**.** *Under Assumption 1,*

$$\epsilon_{task} \;\leq\; \epsilon_{goal} + \epsilon_{sub} + \epsilon_{world} + \epsilon_{plan}. \tag{21}$$

Proof. Consider the event $E^c_{\text{goal}} \cap E^c_{\text{sub}} \cap E^c_{\text{world}} \cap E^c_{\text{plan}}$, on which none of the modules makes an error. On this event, the behavior of the agent matches that of the fully ideal agent $A^{\{\text{goal,sub,world,plan}\}}$, so

$$\mathbb{P}\big(E_{\text{fail}} \mid E^c_{\text{goal}} \cap E^c_{\text{sub}} \cap E^c_{\text{world}} \cap E^c_{\text{plan}}\big) = \epsilon_{\text{task}}\big(\{\text{goal}, \text{sub}, \text{world}, \text{plan}\}\big). \tag{22}$$

By monotonicity, this is at most ϵ_{task}.

We decompose

$$\epsilon_{\text{task}} = \mathbb{P}(E_{\text{fail}}) \tag{23}$$

$$= \mathbb{P}\big(E_{\text{fail}} \cap (E_{\text{goal}} \cup E_{\text{sub}} \cup E_{\text{world}} \cup E_{\text{plan}})\big) + \mathbb{P}\big(E_{\text{fail}} \cap E^c_{\text{goal}} \cap E^c_{\text{sub}} \cap E^c_{\text{world}} \cap E^c_{\text{plan}}\big). \tag{24}$$

The second term is at most $\epsilon_{\text{task}} \cdot \mathbb{P}(E^c_{\text{goal}} \cap E^c_{\text{sub}} \cap E^c_{\text{world}} \cap E^c_{\text{plan}})$. For the first term we use a union bound:

$$\mathbb{P}\big(E_{\text{goal}} \cup E_{\text{sub}} \cup E_{\text{world}} \cup E_{\text{plan}}\big) \leq \epsilon_{\text{goal}} + \epsilon_{\text{sub}} + \epsilon_{\text{world}} + \epsilon_{\text{plan}}. \tag{25}$$

Combining these and using $\mathbb{P}(E^c_{\text{goal}} \cap E^c_{\text{sub}} \cap E^c_{\text{world}} \cap E^c_{\text{plan}}) \leq 1$ gives the claim. □

Theorem 1 gives a concrete way to talk about how improvements in individual modules translate into task-level performance.

3.2 Tree-structured subgoals

Many practical systems use hierarchical decompositions: instead of a flat list of subgoals, they build a tree where internal nodes correspond to intermediate objectives and leaves correspond to concrete skills or motion primitives [Dietterich, 2000, Sutton et al., 1999, Andreas et al., 2017]. In such settings it is natural to ask how decomposition errors scale with the structure of the tree.

Suppose that the ideal decomposition $M^\star_{\text{sub}}(\varphi^\star)$ can be represented as a rooted tree $\mathcal{T}^\star$. Each node v carries a subgoal ψ_v and is associated with a sub-trajectory τ_v. We say that node v is satisfied if $\tau_v \models \psi_v$, and the overall task is considered successful if all leaves are satisfied. Assume that the branching factor is at most B and the depth is at most D, so the number of leaves L satisfies $L \leq B^D$.

Let $\hat{\mathcal{T}}$ be the tree produced by M_{sub}. For simplicity, suppose that $\hat{\mathcal{T}}$ has the same structure as $\mathcal{T}^\star$ but possibly different node labels $\hat{\psi}_v$. Define node-wise decomposition error probabilities

$$\epsilon^{(v)}_{\text{sub}} \triangleq \mathbb{P}(\hat{\psi}_v \not\equiv \psi_v), \qquad \bar{\epsilon}_{\text{sub}} \triangleq \max_v \epsilon^{(v)}_{\text{sub}}. \tag{26}$$

Theorem 2 (Tree-structured decomposition)**.** *Under Assumption 1 and the tree structure described above,*

$$\epsilon_{task} \;\le\; \epsilon_{goal} + \epsilon_{world} + \epsilon_{plan} + L\bar{\epsilon}_{sub}, \qquad L \le B^D. \tag{27}$$

Proof sketch. If every node label matches its ideal counterpart ($\hat{\psi}_v \equiv \psi_v$ for all v), then we can reuse the argument from Theorem 1. If there are mismatches, then task failure can be caused by one or more incorrect node labels. A union bound over leaves shows that the probability of failure attributable to decomposition errors is at most $L\bar{\epsilon}_{\text{sub}}$. The other modules contribute as before. The key point is that the number of places where decomposition can fail grows with the number of leaves. □

This matches a basic intuition: in a hierarchical setting, the cost of decomposition errors grows with the size of the subgoal tree, not with the raw trajectory length.

3.3 CRUX selection as model choice

We now turn to the part of the pipeline where CRUX differs from the baseline: how it chooses among multiple candidate interpretations.

Fix an environment $\mathcal{M}$ and an instruction ℓ. Instead of producing a single interpretation, CRUX constructs a finite set of candidate goals

$$\mathcal{G} = \{\varphi_1, \ldots, \varphi_K\} \subset \Phi, \tag{28}$$

for example, variants that differ in the ordering of sub-tasks or in how strictly they interpret ambiguous phrases. For each φ_k, the downstream modules (decomposition, planner, world model) yield a policy π_k. Under a specified execution noise model (e.g., action slip and start perturbations), each policy has an associated robust success probability

$$p_k \triangleq \mathbb{P}\big(R(\tau, \varphi^\star) = 1 \,\big|\, \text{we run the pipeline with } \varphi_k\big). \tag{29}$$

CRUX does not observe the p_k directly. Instead, for each candidate k it runs N counterfactual rollouts in the world model. Each rollout consists of sampling a perturbed initial state and noise process, executing π_k, and recording whether the true task $\varphi^\star$ would have been satisfied. Let $Y_{k,1}, \ldots, Y_{k,N}$ be the resulting binary indicators. We define the empirical robust success

$$\hat{p}_k \triangleq \frac{1}{N} \sum_{j=1}^{N} Y_{k,j}, \tag{30}$$

and then select

$$\hat{k} \in \arg\max_{k \in \{1,\ldots,K\}} \hat{p}_k. \tag{31}$$

The agent executes $\pi_{\hat{k}}$ in the real environment.

This is a model-selection problem with bounded, Bernoulli-type noise. A standard concentration argument gives a finite-sample guarantee for how close CRUX's choice is to the best candidate.

Theorem 3 (Finite-sample guarantee for CRUX)**.** *Fix a candidate set $\mathcal{G}$ of size K and a rollout budget $N \ge 1$ per candidate. Assume that, for each k, the rollouts $Y_{k,1}, \ldots, Y_{k,N}$ are i.i.d. Bernoulli with mean p_k. Let $k^\star \in \arg\max_k p_k$. Then for any $\delta \in (0,1)$, with probability at least $1-\delta$,*

$$p_{k^\star} - p_{\hat{k}} \le 2\sqrt{\frac{1}{2N} \log \frac{2K}{\delta}}. \tag{32}$$

Proof. Fix k. By Hoeffding's inequality,

$$\mathbb{P}\big(|\hat{p}_k - p_k| > \varepsilon\big) \le 2\exp(-2N\varepsilon^2). \tag{33}$$

Applying a union bound over $k = 1, \ldots, K$ yields

$$\mathbb{P}\Big(\exists k : |\hat{p}_k - p_k| > \varepsilon\Big) \le 2K\exp(-2N\varepsilon^2). \tag{34}$$

Setting the right-hand side equal to δ and solving for ε gives

$$\varepsilon = \sqrt{\frac{1}{2N} \log \frac{2K}{\delta}}. \tag{35}$$

Thus with probability at least $1 - \delta$, we have $|\hat{p}_k - p_k| \leq \varepsilon$ for all k.

On this event,

$$\hat{p}_{k^\star} \leq p_{k^\star} + \varepsilon, \qquad \hat{p}_{\hat{k}} \geq p_{\hat{k}} - \varepsilon, \tag{36}$$

and by definition of $\hat{k}$,

$$\hat{p}_{\hat{k}} \geq \hat{p}_{k^\star}. \tag{37}$$

Putting these together,

$$p_{\hat{k}} - \varepsilon \;\geq\; \hat{p}_{\hat{k}} \;\geq\; \hat{p}_{k^\star} \;\geq\; p_{k^\star} - \varepsilon, \tag{38}$$

so

$$p_{k^\star} - p_{\hat{k}} \;\leq\; 2\varepsilon = 2\sqrt{\frac{1}{2N} \log \frac{2K}{\delta}}. \tag{39}$$

This holds with probability at least $1 - \delta$. □

Theorem 3 says that if we spend $N = O(\varepsilon^{-2} \log K)$ rollouts per candidate, then CRUX chooses an interpretation whose robust success is within $O(\varepsilon)$ of the best one in $\mathcal{G}$. The result does not depend on the internal structure of the candidates or the planner; it only assumes that the counterfactual world model reflects the noise model we care about.

4 CRUX in a stochastic gridworld

We now instantiate CRUX in a small gridworld and compare it to a single-interpretation baseline. The setting is intentionally simple, but it lets us tie the theory to concrete numbers.

4.1 Environment

The environment is a 6×6 grid. A small set of labeled objects, such as `A`, `B`, and `C`, is placed at random at distinct grid cells. The agent starts at a random empty cell. The state records the positions of the agent and the objects. The action set is $\{\text{up}, \text{down}, \text{left}, \text{right}\}$ with clamping at the borders, and episodes end after a fixed horizon.

Tasks are sequences of objects that must be visited in a specified order. For each episode we draw a random permutation of the object labels and take that as the true goal. We then generate a natural-language instruction with a simple template such as

> "Please visit A, then B, then C."

We run experiments in two regimes. In the *clean* regime, the environment is deterministic and the agent's commanded actions are executed as given from the original start state. In the *noisy* regime, we introduce:

- an action slip probability p_{slip}: with probability p_{slip} the commanded action is replaced with a randomly chosen different cardinal move; and
- a start perturbation radius r: before execution, the agent's initial position is shifted by a random offset of at most r cells in ℓ_1 distance, clamped to the grid.

The world model used for counterfactual rollouts uses the same stochastic rules.

A run is counted as successful if the agent visits the correct sequence of object locations in order before the horizon.

4.2 Agents

Both agents use the same BFS-based planner and the same stochastic world model. They differ only in how they perform goal interpretation.

Table 1: Clean and robust success in two noisy regimes. Means are estimated over multiple tasks; confidence intervals (CI) are shown where available.

	Clean success		Robust success			
Regime	Baseline	CRUX	Baseline		CRUX	
Moderate noise	1.000	1.000	0.032	(CI: $[0.024, 0.039]$)	0.054	(CI: $[0.042, 0.065]$)
More noise	1.000	1.000	0.011		0.019	

Baseline: single interpretation. The baseline agent parses the instruction once and commits to that interpretation. In our implementation this parser is a lightweight heuristic: it scans the string, extracts object labels (e.g. `A`, `B`, `C`), and orders them by first occurrence. The resulting sequence is treated as the target visit order. The agent then uses BFS to compute a shortest path that visits each object in the specified order. No further robustness reasoning is applied.

CRUX: multi-hypothesis interpretation with counterfactual selection. CRUX uses the same BFS planner but does not trust a single interpretation. Given the instruction, it generates several candidate sequences (base order, reverse order, truncated variants, and shuffled versions of the mentioned objects). For each candidate, CRUX plans a path with BFS and then uses the world model to run N counterfactual rollouts with action slip and start perturbations. It estimates the robust success probability of each candidate as the fraction of rollouts that satisfy the true goal and then selects the candidate with the highest estimated robust success. The policy for that candidate is then executed in the real environment.

5 Experiments

We now summarize the behavior of CRUX and the baseline in the gridworld described above. All reported results are averages over independently sampled tasks; robust success is estimated with multiple perturbed rollouts per task.

5.1 Metrics and protocol

We report two main metrics:

- **Clean success:** fraction of tasks solved in the deterministic environment with no action slip and no start perturbation.
- **Robust success:** for each agent and each task, we fix the policy and estimate its success probability under the noisy environment by running multiple perturbed rollouts, then average these estimates across tasks.

We consider two aggregate noise regimes (moderate and more severe) and, for the moderate regime, compute 95% confidence intervals for robust success using a normal approximation over tasks. We also sweep the action slip probability p_{slip} over a small set of values and track how robust success decays.

5.2 Aggregate results

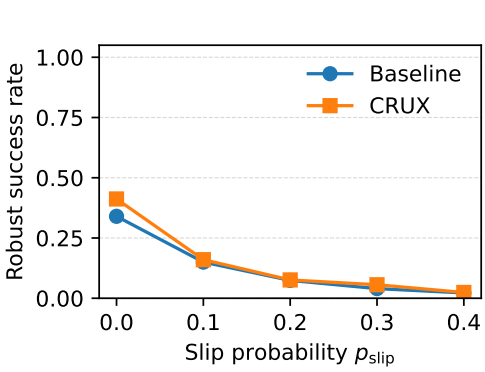
Table 1 shows the clean and robust success rates for the two noise regimes. In both regimes, the clean success of both agents is 1.000: the planners can solve all tasks when the environment is deterministic.

In the moderate noise regime, the baseline succeeds on roughly 3.2% of noisy rollouts, whereas CRUX achieves about 5.4%. The estimated 95% confidence interval for the baseline is $[0.024, 0.039]$, while for CRUX it is $[0.042, 0.065]$. These intervals do not overlap, which suggests that the improvement in robust success is not just sampling noise. Both agents have the same clean performance, so the gain comes purely from better behavior under perturbations.

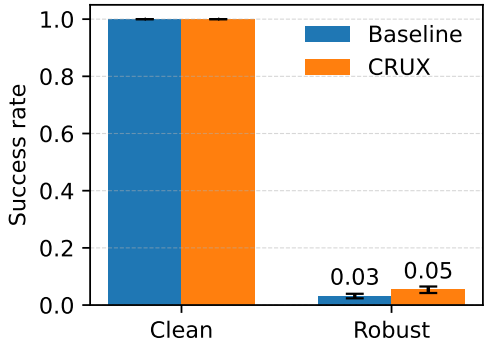
In the more severe noise regime, both agents degrade, as expected. However, CRUX still maintains a margin: robust success increases from 0.011 for the baseline to 0.019 for CRUX.

Table 2: Robust success as a function of action slip probability p_{slip}. Values are means over tasks and noisy rollouts.

p_{slip}	Baseline robust	CRUX robust
0.0	0.340	0.412
0.1	0.150	0.160
0.2	0.074	0.076
0.3	0.040	0.056
0.4	0.022	0.024



(a) Robust success vs. slip probability p_{slip}.



(b) Clean and robust success in the moderate noise regime.

Figure 1: Comparison of baseline and CRUX under different noise conditions.

5.3 Robust success as a function of noise

To see how robustness degrades with noise, we sweep the action slip probability p_{slip} from 0.0 to 0.4 in steps of 0.1, keeping the start perturbation radius fixed. For each value of p_{slip}, we run a fresh batch of tasks and estimate robust success.

Table 2 summarizes the results, and Figure 1 plots them.

6 Discussion and limitations

Our setting is deliberately small: a 6×6 gridworld, a simple interpreter, and a world model that mirrors the true dynamics plus noise. Even so, the modular error view is useful in practice. Clean success near 1.0 suggests the planner and world model are not the main bottlenecks, while gaps in robust success point to sensitivity in interpretation under noise. The bounds in Theorems 1 and 2 turn this diagnosis into explicit inequalities linking ϵ_{goal}, ϵ_{sub}, ϵ_{world}, and ϵ_{plan}.

The CRUX selection rule also behaves as the theory suggests. Treating interpretation as a multi-hypothesis choice and using counterfactual rollouts improves robustness without changing clean performance. The same pattern could carry over to richer settings such as VirtualHome programs [Puig et al., 2018] or neurosymbolic LTL planners [Sun and Shoukry, 2022, Wang et al., 2024].

There are clear limitations. We have not yet tested CRUX on real robots or with full LLM-based front-ends, which raises questions about sample efficiency and world-model accuracy. The analysis also assumes a rollout model that captures the execution noise of interest; when the world model is learned and imperfect [Hafner et al., 2019, Zhao et al., 2024], the bounds would need to be tightened or adjusted.

If embodied benchmarks move towards more modular evaluation—with explicit module-level metrics and perturbation models—ideas in the spirit of CRUX could help design agents whose robustness is not only higher, but also quantified.

Reproducibility statement

All definitions, assumptions, and proofs appear in the main text. The gridworld, CRUX agent, baseline agent, and plotting code (for Figure 1) are implemented in a small, self-contained Python codebase using only standard libraries. We will release anonymized code, configuration files, and figure-generation scripts upon publication.

References

Jacob Andreas, Dan Klein, and Sergey Levine. Modular multitask reinforcement learning with policy sketches. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 166–175. PMLR, 2017.

Thomas G. Dietterich. Hierarchical reinforcement learning with the MAXQ value function decomposition. *Journal of Artificial Intelligence Research*, 13:227–303, 2000.

Georgios E. Fainekos, Hadas Kress-Gazit, and George J. Pappas. Temporal logic motion planning for mobile robots. *IEEE Transactions on Robotics*, 25(6):1370–1381, 2009.

Danijar Hafner, Timothy P. Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. Learning latent dynamics for planning from pixels. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 2555–2565. PMLR, 2019.

Xavier Puig, Kevin Ra, Marko Boben, Jiaman Li, Tingwu Wang, Sanja Fidler, and Antonio Torralba. Virtualhome: Simulating household activities via programs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8494–8502, 2018.

Xiaowu Sun and Yasser Shoukry. Neurosymbolic motion and task planning for linear temporal logic tasks. *IEEE Robotics and Automation Letters*, 7(4):9383–9390, 2022.

Richard S. Sutton, Doina Precup, and Satinder P. Singh. Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. *Artificial Intelligence*, 112(1–2): 181–211, 1999.

Jun Wang, Jiaming Tong, Kaiyuan Tan, Yevgeniy Vorobeychik, and Yiannis Kantaros. Conformal temporal logic planning using large language models: Knowing when to do what and when to ask for help. *ACM Transactions on Cyber-Physical Systems*, 2024. To appear.

Zhigen Zhao, Shuo Cheng, Yan Ding, Ziyi Zhou, Shiqi Zhang, Danfei Xu, and Ye Zhao. A survey of optimization-based task and motion planning: From classical to learning approaches. *IEEE Transactions on Robotics*, 2024. Early access.