Robust Task-Aware State Estimation from Pretrained Perception

Guy Azran
CLAIR Lab
Taub Faculty of Computer Science
Technion – Israel Institute of Technology
Haifa 3200003, Israel
guy.azran@campus.technion.ac.il

Sarah Keren
CLAIR Lab
Taub Faculty of Computer Science
Technion – Israel Institute of Technology
Haifa 3200003, Israel
sarahk@cs.technion.ac.il

Abstract

Agents that perform state estimation using pretrained perception models, e.g., Vision-Language Models (VLMs), face the fundamental challenge that predictions are uncertain. Still, most approaches determinize these predictions into a single best-guess state and make decisions as if it were ground truth. This is brittle because when the guess is wrong, the agent's decisions may be unsafe or infeasible. At the other extreme, maintaining a full probability distribution over symbolic states is often intractable. We propose a middle path, drawing on the underlying probabilities the perception model already produces (e.g., a VLM's output-layer token distributions) to construct a *θ-confidence Minimal Belief Set (MBS_θ)*, a minimal subset of states whose cumulative probability meets a user-specified belief-mass threshold. The minimality of an MBS_θ ensures that the agent considers only the most likely states, enhancing robustness in downstream decision-making while maintaining computational tractability. Using a novel three-layered belief system, we efficiently approximate individual state probabilities from per-feature predictions and task-level constraints. On this belief construction, we base our MBS_θ construction, while maintaining only a lean factored representation of the belief. Our algorithm, *Probability-Ranked Valid State Search (PRaVaSS)*, takes advantage of the unique structure of this belief system to construct an MBS_θ with correctness guarantees and polynomial complexity in the unconstrained case. Using a VLM as the backbone of our state estimator, we validate our method's theoretical properties, scalability with problem size, and improvement of downstream decision-making when used in a task-planning pipeline.

1 Introduction

Autonomous agents increasingly rely on pretrained perception for state estimation from sensory input. For example, an agent may query a Vision-Language Model (VLM) to determine the truth value of semantic facts from images [1, 2, 3]. The resulting symbolic facts are passed to a task planner that selects high-level actions. This pipeline assumes, almost universally, that the perception output can be collapsed to a single most likely interpretation of the scene. However, this is a brittle assumption.

Pretrained perception models' predictions are uncertain. With VLMs, visual ambiguity, occlusion, lighting, noise, model error, and other factors all contribute to noisy predictions, where confidence can vary widely across state features. When the decision maker commits to a single guessed state and that guess is wrong, this may result in attempting unsafe actions, violating user-defined guardrails, or failing outright.

Example 1 (Kitchen Robot)**.** *A mobile manipulator is told to "put away the groceries", as depicted in Figure 1. It uses a task planner that operates over state features, e.g.,* `on-counter(eggs)`,

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

`is-open(fridge)`, `clear(shelf)`. *Using the onboard camera, the agent attempts to ground high-level state features by querying a VLM with semantic queries, e.g., "Is the top shelf clear?", "Are the eggs on the counter?", "Is the fridge door open?". The common robotics pipeline collapses the answers into a single most likely state by accepting the VLM's textual output "Yes" or "No", and plans as if that were the ground truth. But the scene is perceptually ambiguous: a glare on the fridge handle, a partially occluded carton, a bag blocking the shelf, etc. Suppose the VLM's underlying probabilistic outputs indicate that* `clear(top-shelf)` *is true with only 60% confidence, while* `clear(bottom-shelf)` *is true with 95% confidence. If the state estimation determinizes* `clear(top-shelf)=true`*, the agent may execute a* `place(milk,top-shelf)` *action and collide with a hidden object, knocking a glass jar off the shelf.*



Figure 1: Grocery task from Example 1.

In Example 1, the planner did not fail because it planned poorly, but because it planned from the wrong initial state. The root cause is brittle state estimation. At the other extreme, maintaining a full distribution over symbolic states and performing belief-space planning would be safer, but is generally intractable in combinatorial task-planning spaces.

We propose a practical middle ground and construct a θ-*confidence Minimal Belief Set (MBS$_\theta$)*: a minimal subset of states whose cumulative probability satisfies a user-specified belief-mass threshold. MBS$_\theta$ is a compact, task-consistent abstraction of the underlying uncertainty that captures the most likely states while excluding low-probability ones, enabling robust downstream decision-making without incurring the full complexity of belief-space planning. We present a novel three-layered belief system over which the MBS$_\theta$ is constructed, where the distribution over all states is represented by a lean factored representation and task-level constraints. We further propose an algorithm, *Probability-Ranked Valid State Search (PRaVaSS)*, which constructs an MBS$_\theta$ with correctness guarantees, and polynomial complexity in the unconstrained case, by taking advantage of the unique structure of this belief system.

Returning to Example 1, rather than committing to a single state where the top shelf is clear, PRaVaSS selects plausible states that include both possibilities. This uncertainty-aware state representation supports the safer decision to place the milk on the less accessible bottom shelf.

This problem sits at the intersection of neurosymbolic abstraction, VLM-based planning, and planning under uncertainty. In neurosymbolic abstraction, early approaches learned state abstractions by interacting with the environment [4, 5, 6, 7]. In VLM-based planning, recent work uses pretrained VLMs rather than relying on dangerous environment interactions, either directly as planners that generate action sequences from visual observations [1, 8, 9], or as grounders that convert visual observations into symbolic states for downstream planning [2, 3, 10, 11, 12]. Across these lines of work, perception output is typically a single deterministic symbolic state. In contrast, planning under uncertainty takes into consideration the uncertainty in the system's knowledge, e.g., by modeling the state space probabilistically [13, 14] or considering multiple states at a time [15, 16], but typically assumes a small state space due to computational intractability, inspiring recent work to consider less restrictive belief representations [17, 18]. Our work combines these ideas by using the probabilistic outputs of pretrained perception models to construct a compact, task-consistent belief set over symbolic states that can be used for robust downstream decision-making.

Methodologically, the MBS$_\theta$ objective is related to highest-density regions [19] and to set-valued prediction, which seeks the smallest prediction set containing the true label with guaranteed coverage [20]. Here, however, the "labels" are structured symbolic states from a combinatorial space, subject to task-specific validity constraints. Highest-density and top-k enumeration do not handle such constraints without enumerating the space, and set-valued classifiers do not exploit task structure, whereas PRaVaSS extracts the most likely task-valid states lazily, without materializing the exponentially large state space (Theorem 1).

The contributions of this work are as follows:

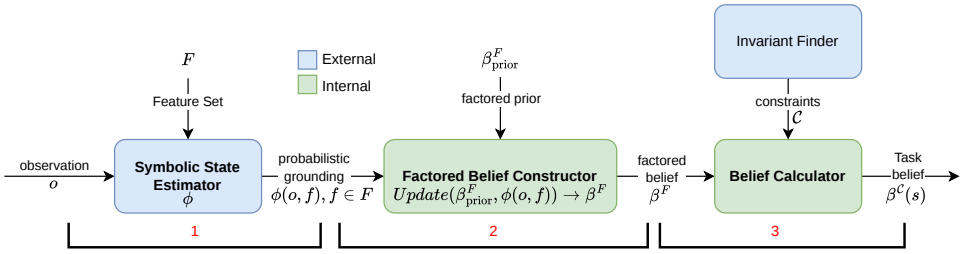


Figure 2: The three-layered belief pipeline. (1) An observation is processed by the SSE to obtain per-feature probabilities. (2) These are updated with an optional prior to produce the factored belief. (3) The factored belief and a set of task-specific constraints are then used to compute the belief.

- **Constrained belief representation**: We propose a three-layered belief system based on task-level information and per-feature predictions from a single observation, which conditions a factored product distribution on task-derived validity constraints to efficiently extract individual state probabilities from a compact belief representation.
- **MBS$_\theta$**: We define MBS$_\theta$ as the minimum-cardinality set of valid states whose cumulative probability meets a user-specified threshold and demonstrate its utility as a compact, task-consistent abstraction for downstream decision-making.
- **PRaVaSS algorithm**: We present PRaVaSS, an algorithm for computing an MBS$_\theta$ with correctness and complexity guarantees, and release an open-source implementation.[1]

The MBS$_\theta$ can serve as input to a range of downstream reasoners. Our empirical evaluation demonstrates the effectiveness of MBS$_\theta$ as the initial belief set in a task-planning pipeline on a variety of domains, showing that the uncertainty abstraction materially improves plan safety and success. Using a VLM as the backbone of our state estimator, we evaluate PRaVaSS on three domains spanning noisy visible state features and genuinely hidden initial-state features, demonstrating faithful coverage, compactness, scalability, and downstream utility.

2 From perception to task-consistent belief

To provide effective state estimation, the agent must assign probabilities to individual states. Because this is intractable for large state spaces, we propose a novel factored belief which is represented by a distribution over a set of state features. Using a pretrained perception model, the agent converts a raw sensory observation into per-feature probability distributions, which can be updated with an optional prior. We combine this factored belief with autonomously extracted task-specific constraints to calculate a belief for individual states without explicitly enumerating all possible states. The full pipeline for this three-layered belief system is illustrated in Figure 2.

Robotic manipulation is a natural motivating domain, since a Vision-Language Model (VLM) can provide useful information about the environment, as in Example 1. However, the formulation applies to any system that produces symbolic facts from pretrained perception. The key requirement is a function that maps observations to per-feature confidence scores. We show how this probability mapping is used to compute the probability for a given state using our three-layered belief system.

2.1 Factored belief

Probabilistic grounding. In the first stage of the pipeline, we convert a raw observation into per-feature probability distributions. For simplicity, we treat features as Boolean-valued. As such, for a feature space F, the state space is simply 2^F, where a state $s \in 2^F$ is a subset of features that are true in that state. Alternatively, we write $s(f) = \mathbb{1}[f \in s]$. This formulation naturally extends to other feature types, e.g., categorical features, by representing their distributions appropriately.

Let O be an observation space and let F be a feature space. We have access to a *Symbolic State Estimator (SSE)* $\phi : O \times F \to [0, 1] \cup \{\text{NULL}\}$. For a given observation $o \in O$ (e.g., an image)

[1] https://github.com/CLAIR-LAB-TECHNION/pravass_neurips2016

and for each feature $f \in F$ (e.g., "the top shelf is clear"), the SSE (e.g., a VLM) is queried to produce $\phi(o, f) \in [0, 1]$, representing its confidence that f holds in the observed scene. We also allow $\phi(o, f) = \text{NULL}$ when the observation provides insufficient evidence to assess f, e.g., when querying whether a shelf is clear when the fridge door is closed.

In practice, the SSE is commonly realized by a VLM [2, 21, 12]. Given an image observation o and a feature-specific query (e.g., "Is the top shelf clear?"), a forward pass yields log-probabilities over the vocabulary, from which we extract:

$$\Pr_{\text{VLM}}(\text{"Yes"}|o, \text{"Is the top shelf clear?"}) \quad \Pr_{\text{VLM}}(\text{"No"}|o, \text{"Is the top shelf clear?"})$$

Normalizing these two values gives the estimated probability $\phi(o, f)$ that f is true.

Incorporating prior information. The probabilistic grounding step produces per-feature estimates. However, this does not take into account any prior information already available, e.g., from previous observations or user-provided hints. The second stage of the pipeline, therefore, combines the grounding output with an optional prior distribution using a principled update rule, such as Bayesian update. We call the combined result *factored belief* and formally define it as follows:

Definition 1 (Factored belief)**.** *Let F be a set of state features. The* factored belief $\beta^F : F \to [0, 1]$ *defines a multivariate Bernoulli distribution over individual features.*

When no prior information is available, the factored belief equals the probabilistic grounding output directly: $\beta^F(f) = \phi(o, f)$ for each observable feature. When a prior distribution β^F_{prior} is available, the factored belief can be obtained by combining it with the grounding output, for example via Bayesian update. For features where $\phi(o, f) = \text{NULL}$, the factored belief retains the prior value if one exists, receives a default uninformative value (e.g., 0.5), or is handled downstream.

2.2 Extracting task constraints

In stage 3 of the pipeline in Figure 2, we convert the unconstrained belief into a more plausible task belief. We do so by incorporating task-specific constraints that rule out impossible states, thus concentrating probability mass on valid states. We define constraints on the state space as a function $\mathcal{C} : 2^F \to \{0, 1\}$, indicating whether a given state satisfies the constraints or not. Equivalently, we write $s \models \mathcal{C}$ to indicate $\mathcal{C}(s) = 1$, and $s \not\models \mathcal{C}$ to indicate $\mathcal{C}(s) = 0$.

To obtain task-level constraints, we need to leverage the structure of the task. Many symbolic domains impose structural constraints on which states are valid. Crucially, this validity notion is task-relative rather than perceptual. The task description determines which symbolic states are admissible and should therefore receive nonzero probability under the downstream uncertainty representation. In many cases, there are obvious constraints that can be identified by inspection and supplied manually or via a simple heuristic. For example, in Example 1, we can easily identify that an object cannot be in two places at once, and thus cannot occupy both the top and bottom shelves simultaneously. However, given a formal symbolic task description, we can often extract these constraints automatically.

A principled source of task-level constraints is the Fast Downward (FD) planning system [22]. FD accepts as input Planning Domain Definition Language (PDDL) [23], a common formalism for describing planning problems that defines the state features, actions and their effects, the initial state, and goal conditions. FD analyzes the PDDL description and identifies invariants in the domain. Importantly, FD induces these invariants by determining which states are reachable from the initial state. However, to obtain useful constraints, this initial state does not have to be the true initial state of the problem, only an ancestor state from which the current state is reachable.

One kind of constraint provided by FD is called mutex groups [24], defined as a set of feature sets $\{c_1, \ldots, c_m\} \subseteq 2^F$. For each i, at most one feature in c_i can be true at a time, i.e., $\mathcal{C}(s) = \mathbb{1}[\forall i, |c_i \cap s| \leq 1]$. An interesting property of disjoint mutex groups, i.e., $c_i \cap c_j = \emptyset$ for all $i \neq j$, is that the cumulative probability of constraint-satisfying states can be computed in linear time relative to the number of features $|F|$, in the worst case. This property will help us correctly calculate state probabilities after applying constraints, as explained in the next section.

2.3 Belief computation

The factored belief provides probabilities for individual features, but downstream reasoning often requires uncertainty over states rather than state features in isolation [16, 14]. The belief computation

stage therefore lifts these marginals to a distribution over symbolic states in 2^F and then refines that distribution using task structure, without enumerating the full state space. The belief computation is broken down in Figure 3, and described in detail below.

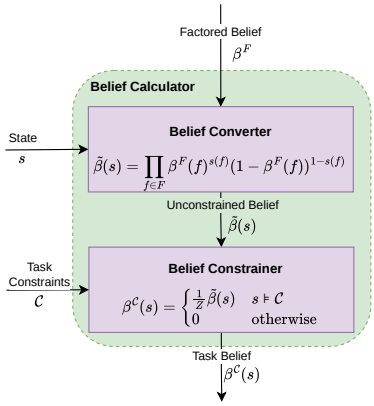


Figure 3: Belief calculator (from Figure 2).

Belief converter. The true joint distribution over all state features is unknown. That would require modeling the complex dependencies between features, which is often intractable. Instead, we make a simplifying assumption of conditional independence between features. This assumption is likely violated in practice, but it allows us to approximate the probability of a single state in linear time, which we can later refine using task-level constraints. We quantify the sensitivity of the resulting θ-confidence Minimal Belief Set (MBS$_\theta$) to correlated perception errors in Appendix J. In this setting, we define the *unconstrained belief* of a state $s \in 2^F$ as:

$$\tilde{\beta}(s) = \prod_{f \in F} \beta^F(f)^{s(f)}(1-\beta^F(f))^{1-s(f)} \quad (1)$$

Belief constrainer. The unconstrained belief captures the perceptual evidence but ignores task-specific structure. It assigns a nonzero probability to every state in 2^F, including states that violate the constraints of the domain. For example, in Example 1, an item cannot be both on the counter and in the fridge simultaneously. To address this, we incorporate task constraints to obtain a belief that concentrates probability mass on valid states.

To incorporate given task constraints $\mathcal{C} : 2^F \to \{0, 1\}$, we define the *task belief* $\beta^{\mathcal{C}}$ by conditioning on $\mathcal{C}$. States that violate the constraints are assigned zero probability, and the remaining states retain their unconstrained belief, normalized accordingly:

$$\beta^{\mathcal{C}}(s) = \begin{cases} \frac{1}{Z}\tilde{\beta}(s) & \text{if } s \models \mathcal{C} \\ 0 & \text{otherwise} \end{cases} \quad (2)$$

where $Z = \sum_{s \models \mathcal{C}} \tilde{\beta}(s)$ is a normalizing constant enforcing that $\beta^{\mathcal{C}}$ is a valid probability distribution. For this to be well-defined, at least one state satisfying $\mathcal{C}$ must have nonzero unconstrained belief.

Note that in the general case, computing Z exactly requires summing over all valid states, which may be expensive or intractable. However, using disjoint mutex groups, Z can be computed in linear time, as shown in Appendix G.4. Furthermore, we can use a relaxed normalization approach where $Z_{\text{relaxed}} \geq Z$ to compute a task belief that is not a true probability distribution, but rather a lower bound on the true constrained probability of each state. We can always choose $Z_{\text{relaxed}} = 1$, which corresponds to no normalization. Regardless of the choice of normalizer, the resulting task belief preserves the relative ordering of states by probability. The tradeoff, however, is that the task belief underestimates the true probabilities, leading to more conservative downstream reasoning that may consider more candidate states than strictly necessary. Consequently, whenever the reported cumulative mass of a set reaches a threshold θ, so does its true constrained mass, so the relaxation costs minimality rather than validity (Corollary 2).

3 Selecting a compact uncertainty set

The task belief $\beta^{\mathcal{C}}$ assigns coherent probabilities to valid symbolic states, jointly capturing perceptual uncertainty and task structure. However, $\beta^{\mathcal{C}}$ is still a distribution over a combinatorial state space. Taking into account the entire state space, whose size is exponential in $|F|$, will render downstream reasoning computationally intractable. On the other hand, collapsing the belief to a single most likely state discards perceptual uncertainty and yields fragile decisions under noisy perception.

We seek a principled middle ground that is the smallest set of valid states whose cumulative probability meets a user-specified belief-mass threshold θ, yielding a robust handoff that is compact enough to remain tractable yet faithful to the underlying uncertainty. We introduce the *MBS$_\theta$*, which formalizes this middle ground.

Algorithm 1 Probability-Ranked Valid State Search (PRaVaSS)

Require: $\beta^F : F \to [0,1], \theta \in [0,1], \mathcal{C} : 2^F \to \{0,1\}, Z \in (0,1]$
Ensure: $b \in \arg\min_{b' \subseteq 2^F} \left\{ |b'| \middle| \sum_{s \in b'} \beta^{\mathcal{C}}(s) \geq \theta \right\}$ s.t. $\forall s \in b, \forall s' \notin b \quad \beta^{\mathcal{C}}(s) \geq \beta^{\mathcal{C}}(s')$.

1: $s_0(f) \leftarrow \mathbb{1}[\beta^F(f) > 0.5] \quad \forall f \in F$ ▷ Start with a most likely state
2: $\tilde{\beta}(s_0) \leftarrow$ CalculateUnconstrainedBelief(β^F, s_0) ▷ Calculated via Equation (1)
3: $Q \leftarrow$ Empty max-heap; Visited $\leftarrow \{s_0\}$
4: Q.Insert(s_0, $\tilde{\beta}(s_0)$)
5: $b \leftarrow \emptyset$; $p \leftarrow 0$ ▷ Selected states and their cumulative probability
6: **while** $p < \theta \wedge Q \neq \emptyset$ **do**
7: $s, \tilde{\beta}(s) \leftarrow Q$.ExtractMax()
8: **if** $s \models \mathcal{C}$ **then** ▷ Only include constraint-satisfying states
9: $\beta^{\mathcal{C}}(s) \leftarrow \frac{1}{Z}\tilde{\beta}(s)$ ▷ Calculated task belief via Equation (2)
10: $b \leftarrow b \cup \{s\}$; $p \leftarrow p + \beta^{\mathcal{C}}(s)$
11: **end if**
12: **for** $f \in F$ **do**
13: $s' \leftarrow$ Flip(s, f) ▷ Flip f's value: $s'(f) = 1 - s(f)$
14: **if** $s' \notin$ Visited **then**
15: Visited $\leftarrow$ Visited $\cup \{s'\}$
16: $\tilde{\beta}(s') \leftarrow$ CalculateUnconstrainedBelief(β^F, s') ▷ Calculated via Equation (1)
17: Q.Insert(s', $\tilde{\beta}(s')$)
18: **end if**
19: **end for**
20: **end while**
21: **return** b

Definition 2 (MBS$_\theta$)**.** *Let $\beta : 2^F \to [0,1]$ be a belief, $\theta \in [0,1]$ be a probability threshold, and $\mathcal{C}$ be a set of constraints on the symbolic state space 2^F. A* θ-confidence Minimal Belief Set (MBS$_\theta$)*, denoted* MBS$_\theta(\beta, \mathcal{C})$*, is a subset of states $b \subseteq 2^F$ of minimal cardinality such that $\sum_{s \in b} \beta(s) \geq \theta$ and $\forall s \in b, \forall s' \notin b$, it holds that $\beta(s) \geq \beta(s')$.*

Attributed mass vs. coverage. In Definition 2, θ bounds the probability mass that the belief β attributes to the returned set. This mass equals the probability that the true state is contained in the set, which we call *true-state coverage*, only insofar as β is well calibrated. We therefore measure coverage directly against ground truth in Section 4, and report the empirical relation between attributed mass and coverage in Appendix H.

Naively searching for an MBS$_\theta$ by enumerating all valid states and sorting them according to their probabilities would be highly inefficient. We propose Probability-Ranked Valid State Search (PRaVaSS), a novel algorithm described in Algorithm 1. PRaVaSS conducts a best-first search over symbolic states, where two states are considered neighbors when they differ in the value of exactly one feature. PRaVaSS disassembles the belief calculator depicted in Figure 3, sorting states by their unconstrained belief and selecting them according to their task belief. The search begins from the most likely state, obtained by setting each feature to its most probable value, and progressively expands neighbors in non-increasing order of probability by flipping the state bits (feature values). At each step, the highest-priority state is extracted from a max-heap. If it satisfies the task constraints, it is added to the growing MBS$_\theta$, and otherwise it is skipped but still expanded so that valid neighbors remain reachable. The search halts as soon as the cumulative probability of the collected states crosses the threshold θ. See Appendix A for a step-by-step numerical example describing the search process on a small instance with a guiding illustration (Figure 8).

In more detail, in Algorithm 1, lines 1–4 initialize the search at s_0 with a max-heap containing only this state. Line 7 extracts the highest-priority state from the heap, based on its unconstrained belief (computed via Equation (1)). The extracted state is added to the output belief set if it satisfies the constraints (line 10). The loop at line 12 expands the extracted state by flipping each feature in turn, enqueuing any unvisited neighbor (line 17). The termination condition (line 6) halts the search once the cumulative probability of selected states meets or exceeds θ.

The key insight behind PRaVaSS is that the heap prioritizes by the unconstrained belief $\tilde{\beta}$ (computed via Equation (1)) rather than the task belief $\beta^{\mathcal{C}}$, yet constraint-satisfying states are extracted in non-increasing order of $\beta^{\mathcal{C}}$ as long as the heap is not empty. This lets us prioritize exploration while guaranteeing that the output is an MBS_θ without enumerating the full state space.

Theorem 1. *In Algorithm 1, when a constraint-satisfying state s is extracted from the heap Q, then for all $s' \notin b$ it holds that $\beta^{\mathcal{C}}(s) \geq \beta^{\mathcal{C}}(s')$, where $\beta^{\mathcal{C}}$ is the task belief computed by Equation* (2)*.*

Sketch. Algorithm 1 explores states via a max-heap ordered by the unconstrained belief $\tilde{\beta}$. The *canonical path* from s_0 to any state s, formed by flipping differing features, is monotonically non-increasing in $\tilde{\beta}$ since each flip moves a feature from its most likely value. Consequently, when s is extracted, any unextracted s' is either in the heap with $\tilde{\beta}(s') \leq \tilde{\beta}(s)$, or has an unextracted ancestor s'' in the heap on its canonical path with $\tilde{\beta}(s) \geq \tilde{\beta}(s'') \geq \tilde{\beta}(s')$. Thus, states are extracted in non-increasing order of $\tilde{\beta}$, meaning constraint-satisfying states are extracted in non-increasing order of their proportional task belief $\beta^{\mathcal{C}}$. □

From Theorem 1, it follows that PRaVaSS correctly returns an MBS_θ as defined in Definition 2.

Corollary 1. *The PRaVaSS algorithm described in Algorithm 1 returns an MBS_θ as defined in Definition 2 relative to the task belief $\beta^{\mathcal{C}}$ as calculated in Equation* (2)*.*

Theorem 1 and Corollary 1 are exactness guarantees relative to the stated task belief: PRaVaSS returns exactly the MBS_θ of $\beta^{\mathcal{C}}$. They do not guarantee robustness to misspecification of $\beta^{\mathcal{C}}$, e.g., through violations of the conditional independence assumption of Section 2.3, whose effect we measure empirically in Section 4 and Appendix J.

In the worst case, the number of explored states is exponential in $|F|$. Even if we bound the MBS_θ size to k, there might still be many constraint-violating states that need to be explored before finding k valid states. In the absence of constraints, we can provide a tighter bound on the runtime. With constraints, we hope to encounter a manageable number of constraint-violating states in practice.

Proposition 1. *If $\mathcal{C} = \emptyset$, Algorithm 1 runs in $O(|F|k\log(|F|k))$ time and $O(|F|^2 k)$ bits, where k is the number of states in the MBS_θ.*

In summary, the exact-minimality guarantee of Corollary 1 holds for any constraint set, the polynomial bound of Proposition 1 is proven for the unconstrained case, and the efficiency of PRaVaSS in constrained domains is demonstrated empirically in Section 4. Full proofs are in Appendices B to D. With this, the MBS_θ can be computed efficiently and used as input to downstream reasoners, e.g., as the initial belief set for planning under initial-state uncertainty.

4 Experiments

The goals of our empirical evaluation are threefold: to show that (1) MBS_θ is a good belief representation of a pretrained perception model; (2) PRaVaSS is an efficient method for computing an MBS_θ; and (3) PRaVaSS can be integrated into downstream decision-making to achieve improved robustness. All experiment code and data are available at `https://github.com/CLAIR-LAB-TECHNION/pravass_neurips2016`.

Domains. We evaluate three domains that reflect different aspects of uncertainty. **Photorealistic Blocksworld (PRB)** [25] is a blocksworld environment with photorealistic rendering, providing clean symbolic constraints and enabling controlled scaling. It is also visually challenging due to realistic lighting, textures, and occlusions, leading to uncertain state estimation. **Groceries World (GW)** [2] is a tabletop manipulation domain with realistic visual ambiguity. A robot must rearrange grocery items on different colored tables. The symbolic description does not fully capture the uncertain environmental dynamics. **ALFWorld (ALF)** [26], adapted from the ALFRED benchmark [27], is used as a bounded one-shot stress test for hidden initial-state features, including task-relevant objects that may be hidden in a receptacle, producing genuinely unresolvable uncertainty. Example observations from each domain are shown in Figure 4, while full details, including state features, actions, and objects, are provided in Appendix E. Full details on data construction are in Appendix F.

State estimator and calibration. The SSE was implemented using the S3E [2] Python library. Raw per-feature probability estimates are calibrated via Platt scaling [28], which fits a logistic recalibration



(a) PRB (b) GW (c) ALF

Figure 4: Example observations from each evaluation domain.

of the per-feature logit on a held-out validation split. These calibrated marginals are shared across all baselines, ensuring that differences in output quality reflect only the state-set construction method, not the perception model. All experiments presented in this section use an SSE based on a single fixed VLM backbone and prompting scheme, with GPT-4.1 [29] serving as the underlying model. Appendix I repeats Experiments 1 and 3 with five open-weight VLMs under the same pipeline and calibration protocol.

Constraint extraction. For PRB and GW, task constraints are extracted from the PDDL domain and problem files using the FD translator [22, 24], which discovers mutex groups and structural invariants. For ALF, FD was not applicable due to the problem structure, so we used a domain-specific heuristic to extract key structural invariants, e.g., the agent can only hold one object at a time. Appendix G details the process of extracting constraints and computing the normalizer for each domain.

Experiment 1: "How steep is the tradeoff between true-state coverage and belief set compactness for MBS$_\theta$ as controlled by θ, and how does it compare to other belief set constructions?"

We compare MBS$_\theta$ to three baselines. **Maximum A Posteriori (MAP)**: the single most likely state, i.e., the state with each feature set to its most likely value, yielding a singleton belief set. **Feature Abstention**: the set of states obtained by abstaining on all features whose marginal probability is below a threshold τ [30]. Features with a marginal above τ are determinized to their most likely value, and the rest are treated as ambiguous, i.e., both assignments are included in the returned set. For each θ, we find the smallest τ such that the returned set exceeds the θ mass on average. **Random θ-Mass**: valid states are randomly selected until their cumulative task belief exceeds θ. All methods accept as input the same calibrated marginals from the SSE and the resulting probabilistic grounding (the output from stage 1 in Figure 2). Features assigned NULL by the SSE are treated as ambiguous, i.e., both assignments are included in the belief set. MAP is a point-estimate method, and thus ignores NULL assignments and forces a truth assignment for each feature.

Results. As the belief set grows, the true-state coverage generally increases, but at the cost of degraded robustness and heavier computation. Therefore, we want coverage to increase more for little growth in belief set size. Figure 5 illustrates this coverage-compactness tradeoff. Each curve corresponds to a method. Each point on the curve corresponds to a different θ (or τ calibrated for θ), which affects the size of the belief set. MAP is a singleton, and thus does not grow in size, but its coverage is a lower bound on that of any belief set that includes the MAP, including MBS$_\theta$. Feature abstention erratically jumps in set size for small changes in coverage, since slightly changing the abstention threshold τ can vastly increase the number of abstained features. In contrast, the MBS$_\theta$ curve is smooth and generally dominates feature abstention, indicating that MBS$_\theta$ achieves a better coverage-compactness tradeoff. Note that in ALF, there is a small window where feature abstention dominates MBS$_\theta$ for low θ values, but this is because the feature-abstention subset is not minimal (see Appendix H).

In the GW domain, Random θ-Mass is omitted because it fails to finish running on 90% of instances within its timeout window. In the PRB and ALF domains, it has the steepest coverage-compactness curve, but at the cost of extremely large set sizes. For example, at $\theta = 0.9$, the set size of Random θ-Mass is over 40 times that of MBS$_\theta$ in both PRB and ALF. This is even more pronounced at lower θ values. Moreover, unlike the other methods, Random θ-Mass does not necessarily contain the most likely state, and thus its coverage can be lower than that of MAP despite its large size. A complete table with coverage and set size values is provided in Appendix H. That appendix also plots realized

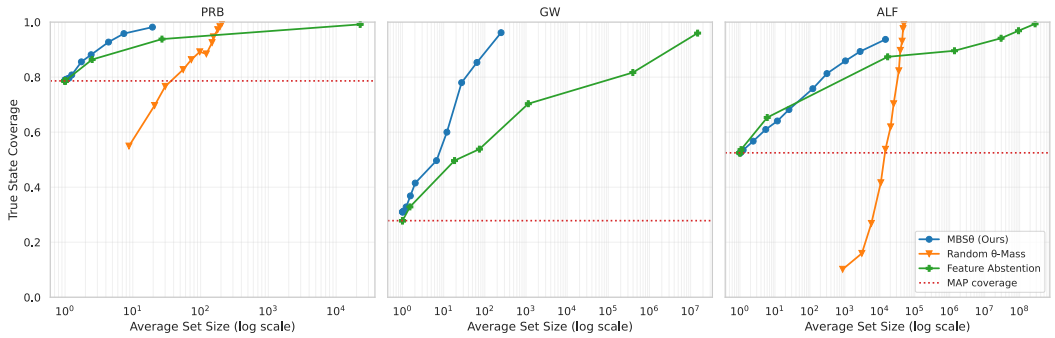


Figure 5: True-state coverage vs. average output set size. Top left is better.

set mass against empirical true-state coverage, which agree within 0.11 for GPT-4.1 at θ between 0.9 and 0.99, and Appendix I shows that MBS_θ retains its coverage-compactness advantage on all six VLM backbones.

Experiment 2: "Is PRaVaSS an efficient method for constructing an MBS_θ compared to naive approaches, and how does it scale with problem size?"

We compare PRaVaSS to an exhaustive approach that computes the MBS_θ by enumerating all $2^{|F|}$ states, computing their task belief $\beta^{\mathcal{C}}$, sorting them, and finding the minimal prefix that crosses the θ mass. Each method is given a 10 s window for each test instance to compute the MBS_θ. We measure how long each method takes to compute the MBS_θ, and how many states it maintains during the search (which corresponds to memory usage). Hardware configurations are detailed in Appendix K. Here, we want to compute an MBS_θ, and thus do not compare to the baselines from other experiments.

Results. Figure 6 shows a cactus plot of the runtime of PRaVaSS and exhaustive enumeration. A point (x, y) on the cactus plot indicates that x instances were solved by the method within y seconds. The plot shows the dominance of PRaVaSS even at the highest tested threshold $\theta = 0.99$, which forgoes only 1% of the probability mass. This shows that PRaVaSS efficiently drops a large mass of low-probability states that waste precious computation time and memory. See Appendix L for a scaling analysis based on problem size, as well as a complete table of runtime and memory values across domains and θ values. There, we show that PRaVaSS can scale to problems with 42% more features than exhaustive enumeration at $\theta = 0.99$, and much more at lower θ. Although all three domains carry constraints, the median number of states visited remains below 160 at every θ (Table 9), so the exponential worst case discussed in Section 3 is not met in practice.

Experiment 3: "Can PRaVaSS be integrated into downstream decision-making tasks and achieve improved robustness compared to point estimates?"

To demonstrate downstream effects, we use PRaVaSS to construct an MBS_θ, and use a robust planner [16, 31] that finds plans that achieve the goal from every state in a given belief set. The ALF domain is excluded from this experiment because its partially observable nature requires the agent to actively collect additional information to ensure robust decision-making. Building on its efficiency, we can

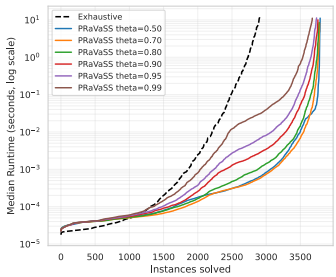


Figure 6: Runtime vs. instances solved.

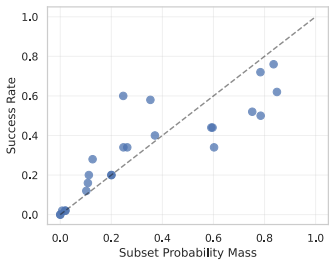


Figure 7: Average attributed mass vs. plan success rate for Adaptive MBS Search (AMS).

Table 1: Downstream planning success rate.

Method	PRB	GW
MAP	16.7% [14.4%, 18.8%]	70.7% [67.2%, 73.8%]
Adaptive Abst.	16.8% [14.7%, 19.1%]	73.3% [69.8%, 76.8%]
AMS (Ours)	**32.6%** [30.1%, 35.3%]	**80.0%** [76.7%, 83.0%]

use PRaVaSS iteratively to find the largest θ for which a robust plan exists, an approach we call Adaptive MBS Search (AMS). AMS provides the attributed mass of the selected belief set as a confidence score for plan success instead of relying on a user-specified threshold. We compare this approach to MAP, where we plan from the single most likely state, and an adaptive version of feature abstention that finds the largest τ for which a plan is found. We use the CPOR planner [32] with an underlying FD planner [22]. AMS uses the full set of constraints extracted by FD, whose mutex groups overlap, so the exact normalizer requires weighted model counting (Appendix G). We therefore use the relaxed normalizer of Section 2.3, which preserves the ordering of states, the only quantity the search consumes, while reported masses remain lower bounds. A closed-loop observe-update-replan study on ALF, in which AMS stays above MAP by 20.9 points before and 16.5 points after a single observation, is reported in Appendix N. For full details, see Appendix M.

Results. Table 1 summarizes the downstream planning results. Bracketed rows are 95% bootstrap confidence intervals (CI) using 10,000 paired resamples over test instances, where success is measured by whether the found plan reaches the goal. Bold values indicate the best-performing method in each domain, and underlined values the second-best. AMS achieves a 13% improvement over MAP in GW, and a 95% improvement in PRB. This shows the power of constraint-based belief set construction and the versatility of PRaVaSS, even in the presence of approximated probability masses obtained with a relaxed normalizer. Two ablations in Appendix M separate the belief-set construction from the constraint information. With identical calibrated marginals and identical full constraints, AMS exceeds adaptive abstention by 15.7 points on PRB (paired 95% CI $[13.6, 17.8]$) and by 5.2 points on GW ($[3.2, 7.3]$), and exceeds its own loose-constraint variant by 16.2 and 4.5 points, respectively, whereas replacing the relaxed normalizer with the exact one changes success by at most 0.7 points. The smaller GW gain is largely a headroom effect, since MAP already succeeds on over 70% of GW instances against a 95.0% oracle ceiling, and the gain grows on backbones with degraded yet informative perception (Appendix I).

Running PRaVaSS iteratively enables AMS to communicate its confidence in a plan through the probability mass of the belief set. Figure 7 scatters the average success rate as a function of AMS's attributed mass across test instances in the PRB domain. Each point on the plot represents the success rate of AMS for a given θ on all PRB instances of a certain horizon and object count configuration, corresponding to different levels of problem difficulty. The results show a strong positive correlation between attributed mass and success rate. The Pearson correlation is $r = 0.866$, with $p \approx 4 \times 10^{-8}$, indicating that the attributed mass is a strong predictor of plan success. Appendix M contains additional results for this experiment, including results showing that planning over MBS_θ can retain its performance when SSE classification is wrong, highlighting how MBS_θ enhances robustness.

5 Conclusion

By translating the probabilistic outputs of foundation models into rigorous symbolic representations and providing a mathematically grounded handoff from uncertain perception to symbolic planning, our framework offers a practical stepping stone toward more reliable neurosymbolic AI systems. Rather than forcing a choice between brittle single-state determinization and intractable full belief-space planning, our θ-confidence Minimal Belief Set (MBS_θ), extracted via the Probability-Ranked Valid State Search (PRaVaSS) algorithm, yields an uncertainty abstraction permissive enough to respect perceptual ambiguity, structured enough to obey task logic, and compact enough for tractable downstream decision-making. Crucially, it achieves this with formal correctness guarantees relative to the stated belief and polynomial complexity in the unconstrained case, significantly outperforming baselines in coverage and compactness, while scaling to problems with up to 154% more state features than exhaustive approaches when capturing the majority of the state space probability mass.

Limitations and future work. We note that the quality of MBS_θ is dependent on the accuracy of the Symbolic State Estimator (SSE), and that θ bounds the mass attributed by a belief that assumes conditional independence between features, rather than certified true-state coverage. Furthermore, our belief system is static and does not take exploration into account. Future extensions will support distribution-free calibration of θ against true-state coverage, temporal belief refinement, learned inter-feature dependencies, and active information-gathering actions, ultimately enabling closed-loop robotic agents that reason safely through an ambiguous world.

Broader impact. For a detailed analysis of societal impacts, see Appendix O.

Acknowledgments and Disclosure of Funding

This work received no third-party funding or support. The authors declare no competing interests. We thank the anonymous reviewers and area chair, whose comments led directly to the ablation, multi-backbone, sensitivity, and closed-loop studies added in this version.

References

[1] Zhutian Yang, Caelan Garrett, Dieter Fox, Tomás Lozano-Pérez, and Leslie Pack Kaelbling. Guiding Long-Horizon Task and Motion Planning with Vision Language Models. In *2025 IEEE International Conference on Robotics and Automation (ICRA)*, pages 16847–16853, May 2025.

[2] Guy Azran, Yuval Goshen, Kai Yuan, and Sarah Keren. S3E: Semantic Symbolic State Estimation With Vision-Language Foundation Models. In *AAAI 2025 Workshop LM4Plan*, February 2025.

[3] Yichao Liang, Nishanth Kumar, Hao Tang, Adrian Weller, Joshua B. Tenenbaum, Tom Silver, João F. Henriques, and Kevin Ellis. VisualPredicator: Learning Abstract World Models with Neuro-Symbolic Predicates for Robot Planning. In *The Thirteenth International Conference on Learning Representations*, October 2024.

[4] Alper Ahmetoglu, M. Yunus Seker, Justus Piater, Erhan Oztop, and Emre Ugur. DeepSym: Deep Symbol Generation and Rule Learning for Planning from Unsupervised Robot Interaction. *Journal of Artificial Intelligence Research*, 75:709–745, November 2022.

[5] George Konidaris, Leslie Pack Kaelbling, and Tomas Lozano-Pérez. From Skills to Symbols: Learning Symbolic Representations for Abstract High-Level Planning. *Journal of Artificial Intelligence Research*, 61:215–289, January 2018.

[6] Masataro Asai, Hiroshi Kajino, Alex Fukunaga, and Christian Muise. Classical Planning in Deep Latent Space. *Journal of Artificial Intelligence Research*, 74:1599–1686, August 2022.

[7] Masataro Asai. Unsupervised Grounding of Plannable First-Order Logic Representation from Images. arXiv:1902.08093, March 2019.

[8] Jiafei Duan, Wentao Yuan, Wilbert Pumacay, Yi Ru Wang, Kiana Ehsani, Dieter Fox, and Ranjay Krishna. Manipulate-Anything: Automating Real-World Robots using Vision-Language Models. In *8th Annual Conference on Robot Learning*, September 2024.

[9] Yingdong Hu, Fanqi Lin, Tong Zhang, Li Yi, and Yang Gao. Look Before You Leap: Unveiling the Power of GPT-4V in Robotic Vision-Language Planning. arXiv:2311.17842, December 2023.

[10] Siwei Chen, Anxing Xiao, and David Hsu. LLM-State: Open World State Representation for Long-horizon Task Planning with Large Language Model. arXiv:2311.17406, April 2024.

[11] Yan Ding, Xiaohan Zhang, Saeid Amiri, Nieqing Cao, Hao Yang, Chad Esselink, and Shiqi Zhang. Robot Task Planning and Situation Handling in Open Worlds. arXiv:2210.01287, September 2024.

[12] Xiaohan Zhang, Zainab Altaweel, Yohei Hayamizu, Yan Ding, Saeid Amiri, Hao Yang, Andy Kaminski, Chad Esselink, and Shiqi Zhang. DKPROMPT: Domain Knowledge Prompting Vision-Language Models for Open-World Planning. arXiv:2406.17659, June 2024.

[13] Hanna Kurniawati, David Hsu, and Wee Sun Lee. SARSOP: Efficient Point-Based POMDP Planning by Approximating Optimally Reachable Belief Spaces. In *Robotics: Science and Systems IV*. Robotics: Science and Systems Foundation, June 2008.

[14] Leslie Pack Kaelbling, Michael L. Littman, and Anthony R. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101(1):99–134, May 1998.

[15] Blai Bonet and Hector Geffner. Planning under partial observability by classical replanning: Theory and experiments. In *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence*, IJCAI'11, pages 1936–1941, Barcelona, Catalonia, Spain, July 2011. AAAI Press.

[16] H. Palacios and H. Geffner. Compiling Uncertainty Away in Conformant Planning Problems with Bounded Width. *Journal of Artificial Intelligence Research*, 35:623–675, August 2009.

[17] Sarah Keren, Sara Bernardini, Kofi Kwapong, and David C. Parkes. Reasoning About Plan Robustness Versus Plan Cost for Partially Informed Agents. *Proceedings of the International Conference on Principles of Knowledge Representation and Reasoning*, 17(1):550–559, July 2020.

[18] Ran Taig and Ronen I. Brafman. Compiling Conformant Probabilistic Planning Problems into Classical Planning. *Proceedings of the International Conference on Automated Planning and Scheduling*, 23:197–205, June 2013.

[19] Rob J. Hyndman. Computing and Graphing Highest Density Regions. *The American Statistician*, 50(2):120–126, May 1996.

[20] Mauricio Sadinle, Jing Lei, and Larry Wasserman. Least Ambiguous Set-Valued Classifiers With Bounded Error Levels. *Journal of the American Statistical Association*, 114(525):223–234, January 2019.

[21] Matteo Merler, Nicola Dainese, Minttu Alakuijala, Giovanni Bonetta, Pietro Ferrazzi, Yu Tian, Bernardo Magnini, and Pekka Marttinen. ViPlan: A Benchmark for Visual Planning with Symbolic Predicates and Vision-Language Models. arXiv:2505.13180, May 2025.

[22] M. Helmert. The Fast Downward Planning System. *Journal of Artificial Intelligence Research*, 26:191–246, July 2006.

[23] D. McDermott, M. Ghallab, A. Howe, Craig A. Knoblock, A. Ram, M. Veloso, Daniel S. Weld, and D. Wilkins. PDDL—The planning domain definition language. Technical report, Yale Center for Computational Vision and Control, Yale University, The AIPS-98 Planning Competition Committee, 1998.

[24] Malte Helmert. Concise finite-domain representations for PDDL planning tasks. *Artificial Intelligence*, 173(5):503–535, April 2009.

[25] Masataro Asai. Photo-Realistic Blocksworld Dataset. arXiv:1812.01818, December 2018.

[26] Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Côté, Yonatan Bisk, Adam Trischler, and Matthew Hausknecht. ALFWorld: Aligning Text and Embodied Environments for Interactive Learning. In *International Conference on Learning Representations*, October 2020.

[27] Mohit Shridhar, Jesse Thomason, Daniel Gordon, Yonatan Bisk, Winson Han, Roozbeh Mottaghi, Luke Zettlemoyer, and Dieter Fox. ALFRED: A benchmark for interpreting grounded instructions for everyday tasks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[28] J. Platt. Probabilistic Outputs for Support Vector Machines and Comparisons to Regularized Likelihood Methods. In *Advances in Large Margin Classifiers*, pages 61–74. MIT Press, 1999.

[29] OpenAI. Introducing GPT-4.1 in the API. https://openai.com/index/gpt-4-1/, April 2025.

[30] Ran El-Yaniv and Yair Wiener. On the Foundations of Noise-free Selective Classification. *Journal of Machine Learning Research*, 11(53):1605–1641, 2010.

[31] David E. Smith and Daniel S. Weld. Conformant Graphplan. In *AAAI/IAAI*, July 1998.

[32] Shlomi Maliah, Radimir Komarnitski, and Guy Shani. Computing Contingent Plan Graphs using Online Planning. *ACM Trans. Auton. Adapt. Syst.*, 16(1):1:1–1:30, January 2022.

[33] Shuai Bai, Yuxuan Cai, Ruizhe Chen, et al. Qwen3-VL Technical Report. arXiv:2511.21631, November 2025.

[34] Gemma Team. Gemma 4 Technical Report. arXiv:2607.02770, July 2026.

A PRaVaSS numerical example

Figure 8 shows a numerical example of how the state space is explored in Algorithm 1 using bit-flip operations. The example consists of a three-feature state space. The factored belief is given as $\beta^F = [0.4, 0.7, 0.9]$ for the first, second, and third features, respectively. The desired probability threshold is set to $\theta = 0.8$. The constraints are that the first and second features are mutually exclusive and exhaustive, i.e., exactly one of them is true in any valid state. The belief normalizer for Equation (2) is

$$Z = 1 - (\tilde{\beta}(000) + \tilde{\beta}(001) + \tilde{\beta}(110) + \tilde{\beta}(111)) = 1 - (0.018 + 0.162 + 0.028 + 0.252) = 0.54$$

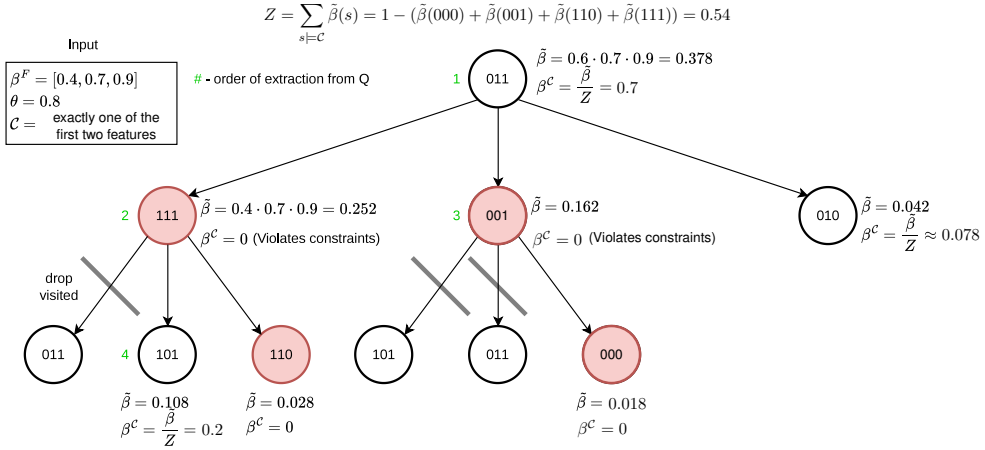


Figure 8: Search for an MBS_θ on three features using the PRaVaSS algorithm, expanding from state 011 via bit flips. Indices appearing to the left of certain nodes indicate the order in which they are extracted from the max-heap, based on the unconstrained belief $\tilde{\beta}$. Red-shaded nodes violate constraints and are excluded from the MBS_θ.

The max-heap is initialized with the state 011, which is the most probable state. This is the first state extracted from the max-heap and added to the MBS_θ. It is expanded via bit flips to states 111, 001, and 010, which are added to the heap. States 111 and 001 violate the constraints, and are assigned zero probability in the task belief. The other task beliefs are computed by normalizing their unconstrained beliefs. After this iteration, $b = \{011\}$, and the cumulative probability of states in this set is $p = \beta^{\mathcal{C}}(011) = 0.7$.

State 111 has the highest unconstrained belief among the three nodes currently in the heap, and is thus extracted next. However, since it violates the constraints, it is not added to the MBS_θ. Expanding 111 reveals states 101, 110, and the already visited 011, so states 101 and 110 are added to the heap. After this iteration, we still have $b = \{011\}$.

Next, state 001 is extracted from the heap, but it also violates the constraints and is not added to the MBS_θ. Expanding 001 reveals the new state 000. After this iteration, we still have $b = \{011\}$.

Finally, state 101 is extracted from the heap and added to the MBS_θ, increasing the cumulative probability to $p = \beta^{\mathcal{C}}(011) + \beta^{\mathcal{C}}(101) = 0.7 + 0.2 = 0.9$. This crosses the threshold $\theta = 0.8$, and the algorithm terminates with $b = \{011, 101\}$ as the resulting MBS_θ.

Had we used the task belief as the heap key instead of the unconstrained belief, state 010 would have been extracted before 111 and 001, so it would have been added to the MBS_θ before 101 was discovered. This would have resulted in a cumulative probability of approximately 0.778, just below the threshold, so the search would have continued and added a third state. However, $\{011, 101\}$ reaches the threshold with only two states, so the three-state output would not have been an MBS_θ by definition.

B Proof of Theorem 1

Definition 3 (Bit diffset)**.** *Let $s, s' \in 2^F$. The* bit diffset $D(s, s')$ *is defined as the set of features that differ between s and s', i.e.,*

$$D(s, s') = \{f \in F | s(f) \neq s'(f)\}$$

Definition 4 (Bit-flip operator)**.** *Let $s \in 2^F$ and $f \in F$. The* bit-flip operator *Flip(s, f) returns a new state $s' \in 2^F$ such that:*

$$\forall f' \in F \quad s'(f') = \begin{cases} 1 - s(f) & \text{if } f' = f \\ s(f') & \text{otherwise} \end{cases}$$

Definition 5 (Canonical path)**.** *Let $s, s' \in 2^F$. Let $D(s, s') = \{f_1, \ldots, f_n\}$ where the features are numbered according to a fixed total ordering over F. The* canonical path *from s to s' is the path of states $s = s_0 \to s_1 \to \cdots \to s_n = s'$ achieved by flipping the bits of state s_0 one by one in the order of the features. That is, for $i \in \{1, \ldots, n\}$, define $s_i =$ Flip(s_{i-1}, f_i), from $s_0 = s$ leading to $s_n = s'$.*

Lemma 1. *In Algorithm 1, let S^{ext} denote the set of states that have been extracted from the heap Q thus far. Then at the moment of max-extraction from Q, it holds that $Q \cup S^{ext} =$ Visited.*

Proof. Note that every insertion to the heap is accompanied by an insertion of the same state to the visited set. All states that have ever been added to the heap are either still in the heap or have been extracted from it. Since no state is ever removed from the visited set, the lemma holds. □

Lemma 2. *In Algorithm 1, the unconstrained belief values of the states on the canonical path from s_0 to any state s are monotonically non-increasing, i.e.,*

$$\tilde{\beta}(s_0) \geq \cdots \geq \tilde{\beta}(s_n) = \tilde{\beta}(s)$$

Proof. Assume by contradiction that there exists $0 < j \leq n$ such that $0 \leq \tilde{\beta}(s_{j-1}) < \tilde{\beta}(s_j)$. Then all of the components that comprise the product in $\tilde{\beta}(s_j)$ are nonzero. Thus, we can safely divide by

$$\prod_{f \in F \setminus \{f_j\}} \beta^F(f)^{s_{j-1}(f)}(1 - \beta^F(f))^{1 - s_{j-1}(f)}$$

From here, we have:

$$\begin{aligned}
&\tilde{\beta}(s_{j-1}) < \tilde{\beta}(s_j) \\
\overset{1}{\iff} &\prod_{f \in F} \beta^F(f)^{s_{j-1}(f)}(1 - \beta^F(f))^{1 - s_{j-1}(f)} < \prod_{f \in F} \beta^F(f)^{s_j(f)}(1 - \beta^F(f))^{1 - s_j(f)} \\
\overset{2}{\iff} &\beta^F(f_j)^{s_{j-1}(f_j)}(1 - \beta^F(f_j))^{1 - s_{j-1}(f_j)} < \beta^F(f_j)^{s_j(f_j)}(1 - \beta^F(f_j))^{1 - s_j(f_j)} \\
\overset{3}{\iff} &\beta^F(f_j)^{s_{j-1}(f_j)}(1 - \beta^F(f_j))^{1 - s_{j-1}(f_j)} < \beta^F(f_j)^{1 - s_{j-1}(f_j)}(1 - \beta^F(f_j))^{s_{j-1}(f_j)}
\end{aligned}$$

Transition 1 follows from the definition of $\tilde{\beta}$ in Equation (1). Transition 2 is a division by all components in the product except for the one corresponding to f_j, which are nonzero, and are equal since $s_{j-1}(f) = s_j(f)$ for all $f \neq f_j$ by the definition of the canonical path in Definition 5. Transition 3 follows from the definition of the bit-flip operation in Definition 4, which states that $s_j(f_j) = 1 - s_{j-1}(f_j)$.

If $s_{j-1}(f_j) = 1$, then the above simplifies to

$$\begin{aligned}
&\beta^F(f_j)^1(1 - \beta^F(f_j))^0 < \beta^F(f_j)^0(1 - \beta^F(f_j))^1 \\
&\Rightarrow \beta^F(f_j) < 0.5
\end{aligned}$$

If $s_{j-1}(f_j) = 0$, then the above simplifies to

$$\begin{aligned}
&\beta^F(f_j)^0(1 - \beta^F(f_j))^1 < \beta^F(f_j)^1(1 - \beta^F(f_j))^0 \\
&\Rightarrow \beta^F(f_j) > 0.5
\end{aligned}$$

Since $s_{j-1}(f_j) = s_0(f_j)$, then both cases contradict the definition of s_0, which for all features is defined as $s_0(f) = \mathbb{1}[\beta^F(f) > 0.5]$. □

Lemma 3. *In Algorithm 1, when a state $s \in 2^F$ is extracted from the max-heap Q, then for all $s' \in 2^F$ not yet extracted from Q, it holds that $\tilde{\beta}(s) \geq \tilde{\beta}(s')$.*

Proof. Let $S_k^{\text{ext}} = \{s_0, \ldots, s_{k-1}\}$ be the set of the first k states extracted from Q, numbered by the order in which they were extracted. We must show that for all k, for all states $s \in S_k^{\text{ext}}$, and for all $s' \notin S_k^{\text{ext}}$, it holds that $\tilde{\beta}(s) \geq \tilde{\beta}(s')$. We do this by induction on k.

Basis: When $k = 1$, this is the first iteration, where the only value in the heap is s_0. By definition, for any $s \neq s_0$, it holds that $\tilde{\beta}(s_0) \geq \tilde{\beta}(s)$.

Assumption: Assume that for k it holds that for all $s \in S_k^{\text{ext}}$ and $s' \notin S_k^{\text{ext}}$ we have $\tilde{\beta}(s) \geq \tilde{\beta}(s')$.

Induction step: Let $S_{k+1}^{\text{ext}} = \{s_0, \ldots, s_k\}$. By the induction assumption, $S_k^{\text{ext}} = S_{k+1}^{\text{ext}} \setminus \{s_k\}$ contains the top k most probable states, i.e., for all $s \in S_k^{\text{ext}}$ and for all $s' \notin S_k^{\text{ext}}$, it holds that $\tilde{\beta}(s) \geq \tilde{\beta}(s')$. It is left to show that for all $s \notin S_{k+1}^{\text{ext}}$, it holds that $\tilde{\beta}(s_k) \geq \tilde{\beta}(s)$.

Assume by contradiction that there exists a state $s \notin S_{k+1}^{\text{ext}}$ such that $\tilde{\beta}(s) > \tilde{\beta}(s_k)$. Then $s \notin Q$ because otherwise it would have been extracted before s_k. By Lemma 1, s has not yet been visited.

On the canonical path $s_0 = u_0 \to \cdots \to u_n = s$, since u_0 has been visited and $u_n = s$ has not yet been visited, there exists $0 < j \leq n$ such that u_j has not yet been visited and u_{j-1} has.

- Since u_{j-1} is in the visited set, then by Lemma 1, it is either in the heap or has been extracted from the heap. It cannot have been extracted from the heap because then it would have been expanded, and u_j would have been visited. Therefore, u_{j-1} is in the heap.
- Since s_k was extracted but u_{j-1} was not, it follows that $\tilde{\beta}(s_k) \geq \tilde{\beta}(u_{j-1})$.
- By Lemma 2, $\tilde{\beta}(u_{j-1}) \geq \tilde{\beta}(u_n) = \tilde{\beta}(s)$.

Putting it all together, we get:

$$\tilde{\beta}(s_k) \geq \tilde{\beta}(u_{j-1}) \geq \tilde{\beta}(s) > \tilde{\beta}(s_k)$$

which is a contradiction. Thus, for all $s \notin S_{k+1}^{\text{ext}}$, it holds that $\tilde{\beta}(s_k) \geq \tilde{\beta}(s)$. □

Theorem 1. *In Algorithm 1, when a constraint-satisfying state s is extracted from the heap Q, then for all $s' \notin b$ it holds that $\beta^{\mathcal{C}}(s) \geq \beta^{\mathcal{C}}(s')$, where $\beta^{\mathcal{C}}$ is the task belief computed by Equation* (2)*.*

Proof. Let s be a constraint-satisfying state extracted from the heap at some iteration. We must show that for all $s' \notin b$, it holds that $\beta^{\mathcal{C}}(s) \geq \beta^{\mathcal{C}}(s')$.

If s' violates the constraints, then trivially $\beta^{\mathcal{C}}(s') = 0 \leq \beta^{\mathcal{C}}(s)$. Otherwise, note that b is comprised of all constraint-satisfying states that were extracted from the heap before s. Therefore, s' has not yet been extracted from the heap. By Lemma 3, we have:

$$\tilde{\beta}(s) \geq \tilde{\beta}(s') \iff \frac{\tilde{\beta}(s)}{Z} \geq \frac{\tilde{\beta}(s')}{Z} \iff \beta^{\mathcal{C}}(s) \geq \beta^{\mathcal{C}}(s')$$

Thus, the theorem holds for all $s' \notin b$. □

C Proof of Corollary 1

Corollary 1. *The PRaVaSS algorithm described in Algorithm 1 returns an MBS$_\theta$ as defined in Definition 2 relative to the task belief $\beta^{\mathcal{C}}$ as calculated in Equation* (2)*.*

Proof. By Theorem 1, each state added to b has $\beta^{\mathcal{C}}$ at least as large as that of any state not yet in b, so b contains the $|b|$ states with the largest $\beta^{\mathcal{C}}$. The algorithm halts the moment the cumulative probability first reaches θ, which means that before the last addition, the cumulative probability was strictly below θ. Since no other set of $|b| - 1$ states can have larger cumulative probability than the top $|b| - 1$ already in b, no smaller set crosses the threshold, so b has minimal cardinality and satisfies the ordering condition of Definition 2. □

D Proof of Proposition 1

Proposition 1. *If $\mathcal{C} = \emptyset$, Algorithm 1 runs in $O(|F|k\log(|F|k))$ time and $O(|F|^2k)$ bits, where k is the number of states in the MBS$_\theta$.*

D.1 Time complexity

Claim 1. *The time complexity of Algorithm 1 is $O(|F|k\log(|F|k))$.*

Proof. **Pre-iteration cost.** In lines 1–5 of the algorithm, we linearly compute s_0 and its probability, and initialize collections with at most one item. This requires $O(|F|)$ operations.

Inner-iteration cost. The internal *for* loop in line 12 consists of a bit flip, a "visited" check, an insertion to the visited set, and an insertion to the heap. Using random access, the bit flip is $O(1)$. Checking membership in the visited set is $O(1)$. Inserting into the visited set is $O(\log(M))$ where M is the size of the set. The heap insertion is $O(\log(N))$ where N is the size of the heap. Assuming none of the states are visited (worst case), then we require $O(|F|(\log(N)+\log(M)))$ for all iterations. We must also acknowledge the time to compute the probability at each iteration. If naively implemented, this would take $O(|F|)$, meaning throughout the iteration, this would take $O(|F|^2)$. However, by updating the probability incrementally according to only the changed bit (divide by unflipped value and multiply by flipped value), this operation takes $O(1)$.

Outer-iteration cost. The *while* loop in line 6 consists of a max extraction from the heap, an insertion of this value to b, and the internal *for* loop. Since b has size at most k, the insertion can be implemented in $O(\log(k))$. The max extraction from the heap requires $O(\log(N))$ where N is the size of the heap. The *while* conditions are checkable in $O(1)$. The inner *for* loop is $O(|F|(\log(N)+\log(M)))$. Considering that at most $|F|$ values are inserted into the visited set and heap at every iteration, then $M = N = |F|k$ at most. Overall, each iteration in the *while* loop requires:

$$\begin{aligned}&O(\log(k)) + O(\log(N)) + O(|F|(\log(N)+\log(M)))\\ =&O(\log(k)) + O(|F|(\log(|F|k)+\log(|F|k)))\\ =&O(|F|\log(|F|k)).\end{aligned}$$

There are at most k iterations, so the overall time complexity of the *while* loop is $O(|F|k\log(|F|k))$.

Since the pre-iteration costs are negligible compared to the iteration cost, the final upper bound on the time complexity stands at $T(F,k) = O(|F|k\log(|F|k))$. □

D.2 Space complexity

Claim 2. *The space complexity of Algorithm 1 is $O(|F|^2k)$ bits.*

Proof. As seen in the proof of time complexity, we store the following values at any one time:

- The visited set, containing $O(|F|k)$ states.
- The heap, containing $O(|F|k)$ states.

Each state can be represented using a bitmask of $|F|$ bits. Therefore $S(F,k) = O(|F|^2k)$ bits. □

E Evaluation domains

This appendix gives a high-level overview of the three evaluation domains used in our experiments: Photorealistic Blocksworld (PRB), Groceries World (GW), and ALFWorld (ALF). All of these domains are available under permissive open-source licenses (Apache 2.0 and MIT). The complete Planning Domain Definition Language (PDDL) domain files for all three are included in our code repository (`https://github.com/CLAIR-LAB-TECHNION/pravass_neurips2016`), which we recommend consulting for the precise predicate signatures, action preconditions, and effects. Data collection from these domains is described in Appendix F, and constraint extraction and normalizer computation are described in Appendix G.

E.1 Photorealistic Blocksworld (PRB)

PRB [25] is a shape-stacking domain in which the agent rearranges blocks and spheres on a table. The challenge is primarily *perceptual*: the photorealistic rendering induces strong observation-level ambiguity (lighting, textures, occlusions) over what are otherwise clean and well-understood symbolic dynamics. The state vocabulary captures the support relation between shapes, distinguishing whether a shape rests directly on the table or on top of a specific block, and which shapes are clear on top. Blocks and spheres are typed separately, since spheres cannot be stacked upon. The action set consists of moves that pick up a clear shape and place it onto the table or onto a clear block; preconditions only check the destination, while universally quantified conditional effects update the previous support without requiring the planner to commit to where the shape started. Because of this design, the size of the state space grows quickly with the number of shapes (each pair of distinct shapes contributes a candidate stacking literal), making PRB our main vehicle for stress-testing scalability under controlled conditions.

E.2 Groceries World (GW)

GW [2] is a tabletop manipulation domain in which a robot sorts grocery items between a fixed set of tables using a single gripper. In contrast to PRB, the visual scene is clearer: the difficulty comes from *abstraction uncertainty*, where the symbolic vocabulary does not fully capture the underlying physical dynamics, e.g., an item may fall on the floor, but the task description enforces that it must be on a table or in the robot's gripper. The vocabulary is intentionally minimal: item-on-table relations together with gripper-state predicates indicating whether the gripper is empty and, if not, which specific item is held. There is no internal structure on items or tables (no containers, no item attributes). The action set covers grasping and releasing, with conditional effects that gracefully handle uncertainty over which table an item currently occupies or which item the gripper holds. The number of features per instance is fixed, which makes GW useful as a complementary stress test to PRB: the symbolic structure is much smaller, but each individual feature carries more dynamical weight.

E.3 ALFWorld (ALF)

ALF [26], adapted from the ALFRED benchmark [27], is a household task domain in which an agent navigates a kitchen-style environment, manipulates items, and performs object transformations such as cleaning, heating, cooling, slicing, and toggling. We use ALF as a bounded one-shot stress test for *hidden* initial-state features: task-relevant objects can be placed inside closed receptacles, producing genuinely unresolvable uncertainty from a single observation. ALF has a substantially richer symbolic structure than the other two domains. Objects are organized under a typed hierarchy that distinguishes manipulable items from receptacles, and both are further parameterized by type-codes (e.g., fridge, sink, microwave for receptacles; knife, butter knife for items) governed by static type-compatibility relations. The dynamic state vocabulary covers visibility and proximity to the agent, containment and open/closed status of receptacles, the agent's grasping state, and per-object physical-state attributes (clean, hot, cool, toggled, sliced) alongside the affordance predicates that gate them. Static type and affordance predicates are fixed within an instance and act as priors for the SSE; the dynamic predicates are the ones over which probabilistic estimates are produced and over which the MBS_θ is computed. The action set spans navigation to a target item or receptacle, opening and closing receptacles, picking up and putting down items, and the transformation actions, all using visibility-based gating and conditional effects to remain safe under partial observability.

F Data construction

For each domain, we generate single-observation instances by sampling a hidden ground-truth state and rendering one observation. In PRB, instances are parameterized by the number of objects $N \in \{3, 4, 5, 6, 7, 8\}$ and the goal horizon $H \in \{1, 2, 4, 6\}$. We generate 50 test and 10 validation instances per (N, H) cell, yielding 1,200 test and 240 validation images. In GW, instances are parameterized by $H \in \{1, 2, 4, 6\}$; we generate 150 test and 50 validation instances per H, yielding 600 test and 200 validation images. In ALF, we randomly sample 2,000 test and 1,000 validation instances from the 7,080 available tasks. All data was processed by the GPT-4.1 Vision-Language Model (VLM) [29] to extract symbolic state feature probabilities. The model was accessed through the OpenAI API and is subject to OpenAI's Terms of Use, Usage Policies, and Sharing & Publication

Policy, which permit API use and welcome research publications related to the OpenAI API when compliant with those policies. The flow of the prompts and extraction of the probabilities were handled by S3E [2], which is available under the MIT license. For the multi-backbone study in Appendix I, the same images were additionally processed through the same S3E pipeline by five open-weight VLMs run locally: Qwen3-VL-8B-Instruct and Qwen3-VL-32B-Instruct [33], and Gemma 4 E2B, E4B, and 31B [34], all distributed under the Apache 2.0 license. The final dataset is linked from our code repository (`https://github.com/CLAIR-LAB-TECHNION/pravass_neurips2016`).

G Domain constraint extraction

This appendix describes how we extract symbolic state-validity constraints from each planning task and how we compute the normalizing constant Z used when conditioning the unconstrained belief on those constraints (Section 2).

G.1 Notation and two constraint sets

Recall from Section 2 that for a feature set F, an at-most-one mutex group $c \subseteq F$ requires that at most one of its atoms is true in any valid state. We maintain two related constraint sets per task instance:

- $\mathcal{C}_{\text{all}}$: *all* extracted at-most-one groups, possibly overlapping. These are the strongest available validity constraints and are stored in the JSON field `constraints`.
- $\mathcal{C}_Z$: a *pairwise-disjoint* subfamily, $c_i \cap c_j = \emptyset$ for all $i \neq j$, stored as `selected_groups`. Disjointness is what makes the normalizer of Appendix G.4 computable in closed form.

G.2 Fast Downward extraction (PRB and GW)

For PRB and GW, $\mathcal{C}_{\text{all}}$ and $\mathcal{C}_Z$ are extracted using the Fast Downward (FD) translator's invariant machinery [24]. We use FD version 22.06, which is publicly available under the GPLv3 license. We parse the PDDL domain and problem, run FD's grounding/reachability procedure, and then run its invariant finder on the reachable grounded task. The translator is invoked with `--keep-unimportant-variables` and `--skip-variable-reordering` so that no variables are dropped or relabeled before fact-group extraction, and invariant generation is supplied with the reachable action-parameter information returned by grounding (omitting it would yield fewer invariants). FD produces (i) a possibly overlapping set of grounded mutex groups, used as $\mathcal{C}_{\text{all}}$, and (ii) a disjoint cover of reachable facts selected for the SAS$^+$ encoding, used as $\mathcal{C}_Z$.

Synthetic seed initial state. FD invariant generation is tied to facts and actions reachable from the supplied initial state. Some benchmark problems encode only a partial relevant initial state, which can cause FD to prune objects that have no asserted truth value, in turn dropping the mutex groups that involve them. To avoid this, we synthesize, before invoking FD, a deterministic seed initial state from the domain schema and the object universe of the original problem, and feed that synthetic problem to FD instead of the original. The synthetic seed is constructed only from publicly available structural information (object names and predicate signatures) and does not read the hidden ground truth, so the extracted constraints remain task-structural rather than label-derived. For GW, every item is placed on a single canonical table, and the gripper is empty; for PRB, every shape is directly on the table and clear on top.

What FD recovers. In GW, FD recovers per-object placement mutexes (an object is on at most one table or in the gripper) and a gripper-capacity mutex (the gripper is empty or holds at most one object). The selected disjoint representation splits these into per-object table-location groups plus one gripper-state group, since the full groups overlap on `robot-gripping`(o) atoms. In PRB, FD recovers per-shape support mutexes (a shape is directly on the table or on at most one block) and per-block occupancy/clearance mutexes (a block is clear or supports at most one shape), with overlap on `shape-on-top-of-block`(s, b) atoms; the selected groups choose a disjoint cover from the same fact-group computation.

G.3 Custom ALF constraints

For ALF, FD's invariant finder does not recover useful nontrivial mutex groups, even when run over all reachable grounded atoms. Two of the most natural-looking candidate constraints are also unsound on this dataset: ALF's $\texttt{inreceptacle}(o, r)$ predicate is not single-valued (e.g., a pan can be associated with multiple stove burners simultaneously), and visibility predicates likewise admit multiple simultaneously true atoms. We therefore add a small set of lifted constraints that is supported by both the ALF action model and the observed ground-truth states.

Let $H = \{\texttt{holds}(o) : o \in O\}$ be the set of grounded holding atoms.

- *Global gripper-capacity:* $\mathrm{AtMostOne}(H)$. Added to both $\mathcal{C}_{\mathrm{all}}$ and $\mathcal{C}_Z$ when $|H| \geq 2$; otherwise omitted as vacuous.
- *Held-vs-contained exclusions:* for each reachable $\texttt{holds}(o)$ and each reachable $\texttt{inreceptacle}(o, r)$, the pairwise mutex $\mathrm{AtMostOne}(\{\texttt{holds}(o), \texttt{inreceptacle}(o, r)\})$. These are added only to $\mathcal{C}_{\mathrm{all}}$, since they overlap on $\texttt{holds}(o)$ and would destroy the disjoint factorization of $\mathcal{C}_Z$.

G.4 Normalizer computation

The normalizer for a constraint set $\mathcal{C}$ is $Z_{\mathcal{C}} = \sum_{s \subseteq F} \mathbb{1}[s \models \mathcal{C}]\, \tilde{\beta}(s)$. Two computations are run.

Closed-form on disjoint selected groups. Because the groups in $\mathcal{C}_Z = \{c_1, \ldots, c_m\}$ are pairwise disjoint and atoms outside any selected group integrate to one under the independent Bernoulli product distribution $\tilde{\beta}$, the normalizer factorizes as $Z_{\mathcal{C}_Z} = \prod_{i=1}^{m} Z_{c_i}$. For a single at-most-one group, only the all-false assignment and the $|c|$ single-true assignments contribute nonzero mass:

$$Z_c = \prod_{f \in c} (1 - \beta^F(f)) + \sum_{f \in c} \beta^F(f) \prod_{f' \in c,\ f' \neq f} (1 - \beta^F(f')).$$

A two-state dynamic program tracking "zero processed atoms true" and "exactly one processed atom true" computes Z_c in $O(|c|)$ time, so $Z_{\mathcal{C}_Z}$ is computed in $O(\sum_i |c_i|) \leq O(|F|)$.

Exact variable elimination on the full constraint set. $\mathcal{C}_{\mathrm{all}}$ generally contains overlapping groups, so the closed-form product above does not apply. We additionally attempt an exact weighted model count of $Z_{\mathcal{C}_{\mathrm{all}}}$ by variable elimination over the factor graph whose unary factors are the per-atom Bernoulli probabilities $\beta^F(f)$ and whose hard factors are the at-most-one indicators $\mathbb{1}[\sum_{f \in c_i} s(f) \leq 1]$. The graph is decomposed into connected components (two constraints share a component iff they share an atom), each component is solved independently using a min-scope variable ordering, and the per-component results are multiplied. Variable elimination is exact but exponential in the induced width; we enforce a configurable bound on the largest intermediate factor scope (default 24). When this bound is exceeded, the variable-elimination attempt is abandoned and the closed-form $Z_{\mathcal{C}_Z}$ is reported as the primary normalizer instead.

G.5 Selected groups as a relaxed normalizer

When variable elimination terminates within the scope bound, the primary normalizer is the exact $Z_{\mathcal{C}_{\mathrm{all}}}$ and no relaxation is involved. When it does not, the fallback $Z_{\mathcal{C}_Z}$ is the relaxed normalizer Z_{relaxed} of Section 2.3. The fallback is sound because every selected at-most-one group is either one of the full groups or a subset of one, so $\mathrm{AtMostOne}(c) \implies \mathrm{AtMostOne}(c')$ for any $c' \subseteq c$. This gives $\Omega(\mathcal{C}_{\mathrm{all}}) \subseteq \Omega(\mathcal{C}_Z)$ where $\Omega(\mathcal{C})$ denotes the set of states satisfying $\mathcal{C}$, and hence by nonnegativity of $\tilde{\beta}$,

$$Z_{\mathcal{C}_{\mathrm{all}}} = \sum_{s \models \mathcal{C}_{\mathrm{all}}} \tilde{\beta}(s) \leq \sum_{s \models \mathcal{C}_Z} \tilde{\beta}(s) = Z_{\mathcal{C}_Z} = Z_{\text{relaxed}}.$$

Disjointness and the relaxation play different roles: disjointness makes $Z_{\mathcal{C}_Z}$ cheap to compute, while the logical implication $\mathcal{C}_{\mathrm{all}} \models \mathcal{C}_Z$ makes it a sound upper bound on $Z_{\mathcal{C}_{\mathrm{all}}}$.

A naive alternative would multiply the per-group normalizers of overlapping full constraints. This is unsound: under the factored Bernoulli belief, the validity events of overlapping groups are not

Table 2: Constraint sources by domain. The primary normalizer is the variable-elimination exact $Z_{\mathcal{C}_{\text{all}}}$ when it stays within the intermediate-scope bound, with the closed-form $Z_{\mathcal{C}_Z}$ as a sound relaxed fallback.

Domain	$\mathcal{C}_{\text{all}}$ (`constraints`)	$\mathcal{C}_Z$ (`selected_groups`)	**Source**
GW	FD all mutex groups: object-placement and gripper-capacity	FD selected SAS$^+$ groups	FD on synthetic seed
PRB	FD all mutex groups: per-shape support and per-block occupancy/clearance	FD selected SAS$^+$ groups	FD on synthetic seed
ALF	Global holding mutex plus pairwise held-vs-contained exclusions	Global holding mutex only (when $\lvert H\rvert \geq 2$)	Custom lifted constraints

independent, and the resulting product can be *smaller* than $Z_{\mathcal{C}_{\text{all}}}$. For example, with atoms x, y, z, all marginals equal to 0.5, and overlapping groups $c_1 = \{x, y\}$, $c_2 = \{y, z\}$, the exact probability of satisfying both is $5/8$, while $\Pr(c_1 \text{ valid}) \Pr(c_2 \text{ valid}) = (3/4)^2 = 9/16 < 5/8$. Overlapping constraints must therefore either be normalized by a method that accounts for the overlap, such as variable elimination, or replaced by a disjoint family that is logically implied by the full constraints.

The relaxation preserves the θ guarantee of Definition 2 in a one-sided form, which we state formally.

Corollary 2 (One-sided guarantee under relaxed normalization)**.** *Let $Z_{\text{relaxed}} \geq Z$, let $\beta^{\mathcal{C}}$ be the exactly normalized task belief of Equation* (2)*, and let b be the set returned by Algorithm 1 run with Z_{relaxed} in place of Z. If the algorithm terminates with reported cumulative mass $p \geq \theta$, then $\sum_{s \in b} \beta^{\mathcal{C}}(s) \geq \theta$, and b is a prefix of the same probability ordering that contains the exact-normalizer $\text{MBS}_\theta(\beta^{\mathcal{C}}, \mathcal{C})$. If instead the total reported mass Z/Z_{relaxed} is below θ, the algorithm terminates with an empty heap and $p < \theta$, a condition that is detectable and must be reported rather than silently approximated.*

Proof. For every valid state, $\tilde{\beta}(s)/Z_{\text{relaxed}} \leq \tilde{\beta}(s)/Z = \beta^{\mathcal{C}}(s)$, so the reported mass of any set is a lower bound on its exact mass, which gives the first claim. Both normalizers are constants, so they do not change the ordering of states by belief, and Algorithm 1 extracts the same sequence of valid states under either one (Theorem 1). The algorithm halts at the first prefix of this sequence whose reported mass reaches θ. Since the reported mass of every prefix is at most its exact mass, this prefix is at least as long as the first prefix whose exact mass reaches θ, which is the exact-normalizer MBS_θ by Corollary 1. Finally, the reported mass summed over all valid states equals Z/Z_{relaxed}. If this is below θ, no prefix reaches θ, so the loop of Algorithm 1 runs until the heap is empty and exits with $p < \theta$. □

The cost of the relaxation is therefore minimality, never validity: the returned set can be a superset of the exact-normalizer MBS_θ, but never smaller.

G.6 Summary by domain

Table 2 summarizes the constraint sources by domain. This design separates validity checking from tractable normalization: $\mathcal{C}_{\text{all}}$ provides the strongest available symbolic consistency tests and feeds the variable-elimination exact normalizer, while the disjoint $\mathcal{C}_Z$ provides an always-available closed-form normalizer for a sound relaxation.

H Full size vs. coverage results

Table 3 summarizes the static state-set quality on a wide range of θ values from 0.01 to 0.99. This is the data used to plot Figure 5, providing precise numbers for accurate comparison between methods. We extend this in Figure 11, where we also show the tradeoff between set size and the probability mass of the belief set returned by each method. Here, we see a strong positive correlation between coverage and attributed mass, indicating that our three-layered belief system has merit in real-world tasks, provided the SSE is well calibrated.

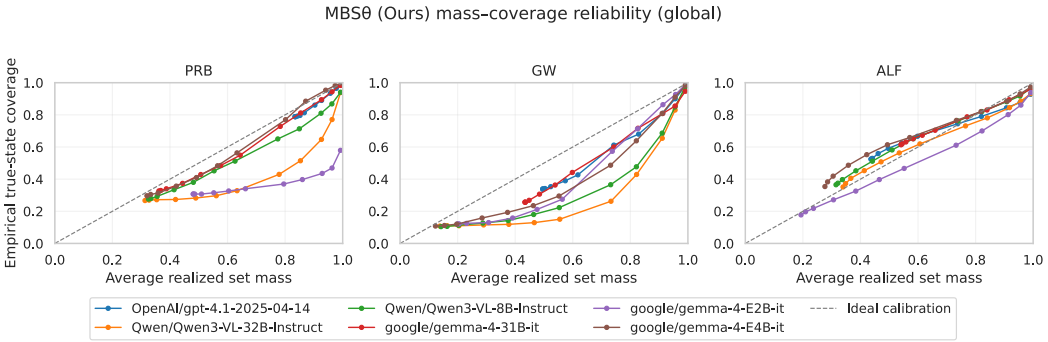


Figure 9: Average realized set mass vs. empirical true-state coverage of MBS_θ for each VLM backbone under global Platt calibration (one logistic recalibration per domain, shared by all features). Each point is one θ value from the grid of Table 3, and the dashed diagonal is ideal calibration.

Attributed mass vs. true-state coverage. θ bounds the mass that the task belief attributes to the returned set (Section 3), whereas coverage is measured against ground truth. Figure 9 plots, for MBS_θ, the average realized set mass against the empirical true-state coverage, where each point is one θ value and the diagonal corresponds to a perfectly calibrated state-level belief. For GPT-4.1, the absolute gap between the two is at most 0.11 across all three domains for θ between 0.9 and 0.99, the operating points used in Experiment 3, and at most 0.20 anywhere on the θ grid. Deviations from conditional independence and residual feature-level miscalibration surface as this gap, which grows for the weaker open-weight backbones of Appendix I. Figure 10 shows the same view for each backbone with all baselines overlaid. The curves in both figures are computed on the common support of instances completed at every θ using the later pipeline run described in Appendix I, and the GPT-4.1 values differ from those of Table 3.

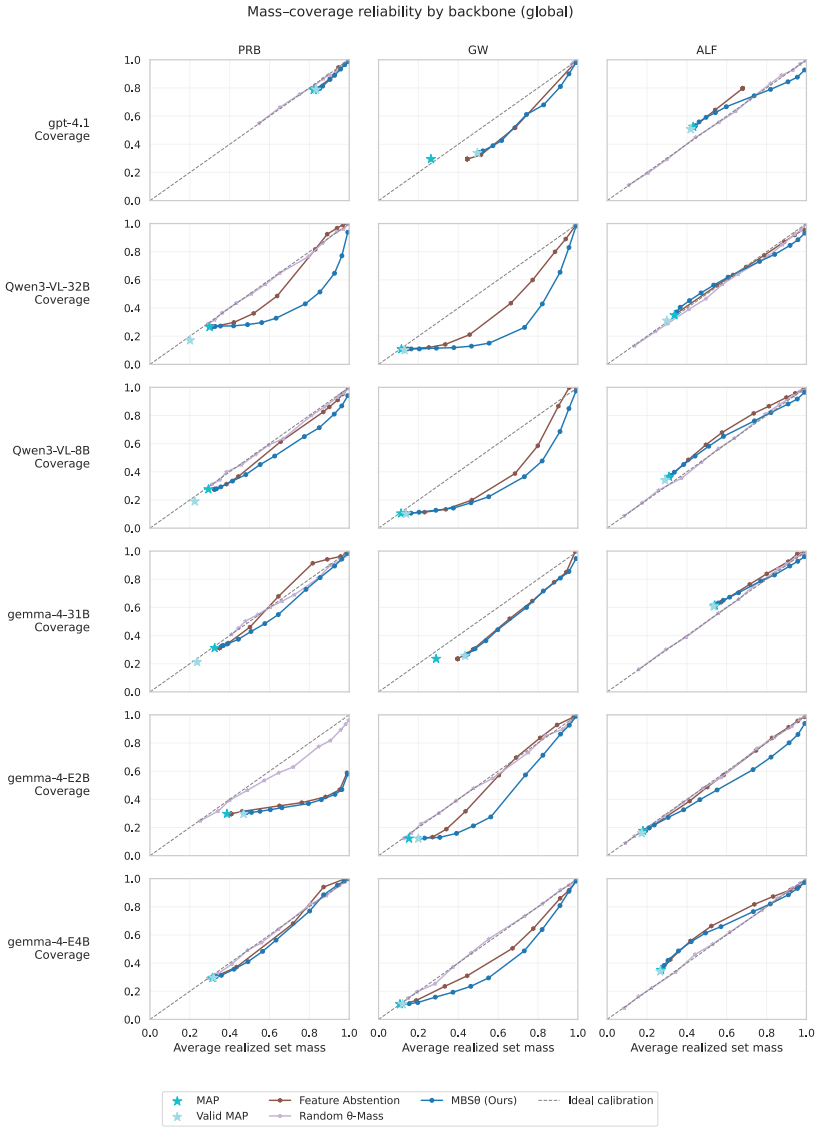


Figure 10: Average realized set mass vs. empirical true-state coverage per VLM backbone (rows) and domain (columns), with all methods overlaid. Curves sweep θ and stars mark the θ-independent Maximum A Posteriori (MAP) and valid MAP points.

Table 3: Belief set quality across all belief-mass thresholds. Coverage is the probability of the true state being included in the belief set, and set size refers to the number of states in that belief set. Values in brackets are 95% bootstrap confidence intervals using 10,000 paired resamples over test instances. Higher coverage and lower set size are better.

	Domain	ALF			GW			PRB		
θ	Method	Coverage	Average Set Size	Average Mass	Coverage	Average Set Size	Average Mass	Coverage	Average Set Size	Average Mass
0.01	MBS_θ (Ours)	0.52 [0.50, 0.55]	1.00 [1.00, 1.00]	0.41 [0.40, 0.42]	0.31 [0.28, 0.35]	1.00 [1.00, 1.00]	0.47 [0.46, 0.49]	0.79 [0.77, 0.81]	1.00 [1.00, 1.00]	0.84 [0.83, 0.86]
	Random θ-Mass	0.10 [0.09, 0.12]	866.04 [696.02, 1069.11]	0.10 [0.09, 0.11]	0.95 [0.89, 1.00]	23.27 [1.00, 59.12]	0.96 [0.92, 1.00]	0.55 [0.52, 0.58]	8.77 [6.31, 11.52]	0.55 [0.52, 0.58]
	Feature Abstention	0.52 [0.50, 0.55]	1.00 [1.00, 1.00]	0.41 [0.40, 0.42]	0.28 [0.24, 0.31]	1.00 [1.00, 1.00]	0.44 [0.42, 0.46]	0.79 [0.77, 0.81]	1.00 [1.00, 1.00]	0.84 [0.82, 0.85]
0.05	MBS_θ (Ours)	0.53 [0.50, 0.55]	1.04 [1.03, 1.06]	0.41 [0.40, 0.42]	0.31 [0.28, 0.34]	1.00 [1.00, 1.01]	0.47 [0.46, 0.49]	0.79 [0.77, 0.81]	1.00 [1.00, 1.00]	0.84 [0.83, 0.86]
	Random θ-Mass	0.16 [0.14, 0.18]	3069.37 [2534.38, 3665.43]	0.17 [0.15, 0.18]	0.95 [0.89, 1.00]	61.69 [1.00, 144.58]	0.96 [0.90, 1.00]	0.70 [0.67, 0.72]	20.95 [15.27, 27.00]	0.69 [0.67, 0.71]
	Feature Abstention	0.52 [0.50, 0.55]	1.00 [1.00, 1.00]	0.41 [0.40, 0.42]	0.28 [0.24, 0.31]	1.00 [1.00, 1.00]	0.44 [0.42, 0.46]	0.79 [0.76, 0.81]	1.00 [1.00, 1.00]	0.84 [0.82, 0.85]
0.10	MBS_θ (Ours)	0.53 [0.51, 0.56]	1.24 [1.19, 1.28]	0.42 [0.40, 0.43]	0.31 [0.28, 0.34]	1.01 [1.00, 1.02]	0.47 [0.46, 0.49]	0.79 [0.77, 0.81]	1.01 [1.00, 1.02]	0.84 [0.83, 0.86]
	Random θ-Mass	0.27 [0.25, 0.29]	5739.67 [4676.61, 6803.03]	0.27 [0.26, 0.28]	0.95 [0.89, 1.00]	170.23 [39.53, 383.89]	0.97 [0.92, 1.00]	0.77 [0.74, 0.79]	30.08 [21.57, 39.39]	0.77 [0.75, 0.79]
	Feature Abstention	0.52 [0.50, 0.55]	1.00 [1.00, 1.00]	0.41 [0.40, 0.42]	0.28 [0.25, 0.31]	1.00 [1.00, 1.00]	0.44 [0.42, 0.46]	0.79 [0.76, 0.81]	1.00 [1.00, 1.00]	0.84 [0.82, 0.85]
0.20	MBS_θ (Ours)	0.57 [0.55, 0.59]	2.39 [2.18, 2.58]	0.44 [0.43, 0.45]	0.31 [0.27, 0.35]	1.07 [1.04, 1.09]	0.48 [0.46, 0.50]	0.79 [0.77, 0.81]	1.03 [1.02, 1.06]	0.85 [0.84, 0.86]
	Random θ-Mass	0.42 [0.39, 0.44]	10692.09 [8707.97, 12765.55]	0.41 [0.40, 0.43]	0.97 [0.92, 1.00]	206.36 [1.00, 463.72]	0.98 [0.95, 1.00]	0.83 [0.81, 0.85]	55.33 [38.66, 75.37]	0.84 [0.82, 0.85]
	Feature Abstention	0.52 [0.50, 0.55]	1.00 [1.00, 1.00]	0.41 [0.40, 0.42]	0.28 [0.24, 0.31]	1.00 [1.00, 1.00]	0.44 [0.42, 0.46]	0.79 [0.76, 0.81]	1.00 [1.00, 1.00]	0.84 [0.82, 0.85]
0.30	MBS_θ (Ours)	0.61 [0.59, 0.63]	5.46 [4.92, 6.10]	0.48 [0.47, 0.49]	0.33 [0.29, 0.37]	1.23 [1.19, 1.28]	0.51 [0.49, 0.52]	0.79 [0.77, 0.82]	1.08 [1.05, 1.12]	0.85 [0.84, 0.86]
	Random θ-Mass	0.54 [0.51, 0.56]	14531.53 [12105.55, 17425.03]	0.53 [0.52, 0.54]	0.97 [0.92, 1.00]	313.11 [1.00, 773.28]	0.99 [0.97, 1.00]	0.86 [0.84, 0.88]	73.04 [49.59, 101.09]	0.87 [0.86, 0.89]
	Feature Abstention	0.52 [0.50, 0.55]	1.00 [1.00, 1.00]	0.41 [0.40, 0.42]	0.28 [0.24, 0.31]	1.00 [1.00, 1.00]	0.44 [0.42, 0.45]	0.79 [0.76, 0.81]	1.00 [1.00, 1.00]	0.84 [0.82, 0.85]
0.40	MBS_θ (Ours)	0.64 [0.62, 0.66]	11.76 [10.35, 13.24]	0.53 [0.52, 0.53]	0.37 [0.33, 0.40]	1.55 [1.49, 1.63]	0.56 [0.55, 0.57]	0.80 [0.77, 0.82]	1.15 [1.10, 1.20]	0.86 [0.85, 0.87]
	Random θ-Mass	0.62 [0.59, 0.64]	20427.42 [16700.43, 24694.47]	0.62 [0.61, 0.63]	0.98 [0.95, 1.00]	360.30 [1.00, 810.62]	0.99 [0.98, 1.00]	0.89 [0.87, 0.91]	97.19 [60.52, 138.75]	0.90 [0.89, 0.91]
	Feature Abstention	0.54 [0.51, 0.56]	1.07 [1.06, 1.08]	0.42 [0.40, 0.43]	0.28 [0.24, 0.32]	1.00 [1.00, 1.00]	0.44 [0.42, 0.46]	0.79 [0.77, 0.81]	1.00 [1.00, 1.00]	0.84 [0.82, 0.85]
0.50	MBS_θ (Ours)	0.68 [0.66, 0.70]	25.24 [22.30, 28.44]	0.59 [0.58, 0.59]	0.41 [0.38, 0.45]	2.04 [1.92, 2.15]	0.60 [0.59, 0.62]	0.81 [0.78, 0.83]	1.26 [1.18, 1.35]	0.87 [0.86, 0.88]
	Random θ-Mass	0.70 [0.68, 0.73]	24937.30 [20295.77, 30291.91]	0.69 [0.68, 0.70]	0.98 [0.94, 1.00]	407.08 [1.00, 932.66]	0.99 [0.99, 1.00]	0.88 [0.87, 0.90]	121.63 [74.96, 175.06]	0.92 [0.91, 0.92]
	Feature Abstention	0.65 [0.63, 0.67]	6.02 [5.13, 7.07]	0.51 [0.50, 0.52]	0.33 [0.29, 0.36]	1.51 [1.43, 1.62]	0.51 [0.49, 0.52]	0.79 [0.76, 0.81]	1.00 [1.00, 1.00]	0.84 [0.82, 0.85]
0.70	MBS_θ (Ours)	0.76 [0.74, 0.78]	121.36 [105.81, 138.25]	0.73 [0.73, 0.74]	0.50 [0.46, 0.54]	6.69 [6.39, 7.04]	0.74 [0.74, 0.75]	0.86 [0.84, 0.87]	1.74 [1.55, 1.97]	0.91 [0.91, 0.92]
	Random θ-Mass	0.82 [0.80, 0.84]	34983.43 [28719.78, 42244.02]	0.81 [0.80, 0.81]	1.00 [1.00, 1.00]	246.00 [1.00, 549.30]	0.99 [0.98, 1.00]	0.93 [0.91, 0.94]	147.71 [89.10, 217.77]	0.95 [0.94, 0.95]
	Feature Abstention	0.87 [0.86, 0.89]	16799.79 [8711.69, 27580.03]	0.71 [0.70, 0.72]	0.50 [0.46, 0.54]	18.38 [15.17, 22.08]	0.70 [0.69, 0.71]	0.79 [0.76, 0.81]	1.00 [1.00, 1.00]	0.84 [0.82, 0.85]
0.80	MBS_θ (Ours)	0.81 [0.80, 0.83]	307.66 [264.46, 348.33]	0.82 [0.82, 0.82]	0.60 [0.56, 0.64]	11.99 [11.29, 12.71]	0.83 [0.82, 0.83]	0.88 [0.86, 0.90]	2.43 [2.08, 2.86]	0.93 [0.93, 0.94]
	Random θ-Mass	0.90 [0.88, 0.91]	38716.36 [31639.23, 46603.39]	0.87 [0.87, 0.87]	0.97 [0.92, 1.00]	315.78 [1.00, 696.75]	0.99 [0.98, 1.00]	0.95 [0.93, 0.96]	155.09 [99.63, 225.51]	0.96 [0.96, 0.97]
	Feature Abstention	0.90 [0.88, 0.91]	1.37×10^6 [8.35×10^5, 2.02×10^6]	0.80 [0.79, 0.80]	0.54 [0.49, 0.58]	73.83 [62.88, 84.68]	0.79 [0.79, 0.80]	0.79 [0.76, 0.81]	1.00 [1.00, 1.00]	0.84 [0.82, 0.85]
0.90	MBS_θ (Ours)	0.86 [0.84, 0.87]	1039.21 [889.86, 1185.09]	0.91 [0.91, 0.91]	0.78 [0.75, 0.81]	27.31 [25.74, 28.95]	0.91 [0.91, 0.91]	0.93 [0.91, 0.94]	4.38 [3.48, 5.45]	0.96 [0.96, 0.96]
	Random θ-Mass	0.93 [0.92, 0.94]	43608.47 [35055.79, 52859.34]	0.93 [0.93, 0.93]	1.00 [1.00, 1.00]	423.30 [1.00, 994.85]	1.00 [1.00, 1.00]	0.97 [0.96, 0.98]	178.98 [116.77, 262.84]	0.98 [0.98, 0.98]
	Feature Abstention	0.94 [0.93, 0.95]	2.96×10^7 [1.94×10^7, 4.33×10^7]	0.90 [0.90, 0.91]	0.70 [0.67, 0.74]	1138.63 [871.31, 1438.83]	0.90 [0.89, 0.90]	0.86 [0.84, 0.88]	2.51 [1.77, 3.45]	0.92 [0.91, 0.92]
0.95	MBS_θ (Ours)	0.89 [0.88, 0.91]	2737.81 [2297.09, 3152.69]	0.96 [0.96, 0.96]	0.85 [0.82, 0.88]	64.33 [60.79, 68.02]	0.96 [0.95, 0.96]	0.96 [0.95, 0.97]	7.38 [5.77, 9.28]	0.98 [0.98, 0.98]
	Random θ-Mass	0.98 [0.97, 0.98]	46391.44 [37589.30, 54982.88]	0.97 [0.97, 0.97]	1.00 [1.00, 1.00]	506.17 [1.00, 1115.55]	1.00 [1.00, 1.00]	0.98 [0.98, 0.99]	192.88 [122.52, 279.53]	0.99 [0.99, 0.99]
	Feature Abstention	0.97 [0.96, 0.98]	9.33×10^7 [5.99×10^7, 1.30×10^8]	0.95 [0.95, 0.96]	0.82 [0.79, 0.85]	3.95×10^5 [3.28×10^5, 4.66×10^5]	0.95 [0.95, 0.95]	0.94 [0.92, 0.95]	26.92 [7.41, 58.78]	0.96 [0.95, 0.96]
0.99	MBS_θ (Ours)	0.94 [0.92, 0.95]	14548.28 [12145.79, 16947.27]	0.99 [0.99, 0.99]	0.96 [0.94, 0.98]	248.48 [235.85, 261.54]	0.99 [0.99, 0.99]	0.98 [0.97, 0.99]	19.47 [14.42, 25.76]	1.00 [1.00, 1.00]
	Random θ-Mass	0.99 [0.99, 1.00]	48959.12 [40236.30, 59356.38]	1.00 [1.00, 1.00]	1.00 [1.00, 1.00]	555.42 [1.00, 1290.43]	1.00 [1.00, 1.00]	1.00 [0.99, 1.00]	206.76 [134.24, 303.06]	1.00 [1.00, 1.00]
	Feature Abstention	0.99 [0.99, 1.00]	2.77×10^8 [1.79×10^8, 3.99×10^8]	0.99 [0.99, 0.99]	0.96 [0.94, 0.97]	1.48×10^7 [1.39×10^7, 1.57×10^7]	0.99 [0.99, 0.99]	0.99 [0.99, 1.00]	22645.68 [2408.07, 50652.07]	0.99 [0.99, 0.99]
point	MAP	0.52 [0.50, 0.55]	1.00 [1.00, 1.00]	0.40 [0.39, 0.41]	0.28 [0.24, 0.32]	1.00 [1.00, 1.00]	0.31 [0.29, 0.33]	0.79 [0.76, 0.81]	1.00 [1.00, 1.00]	0.83 [0.82, 0.85]

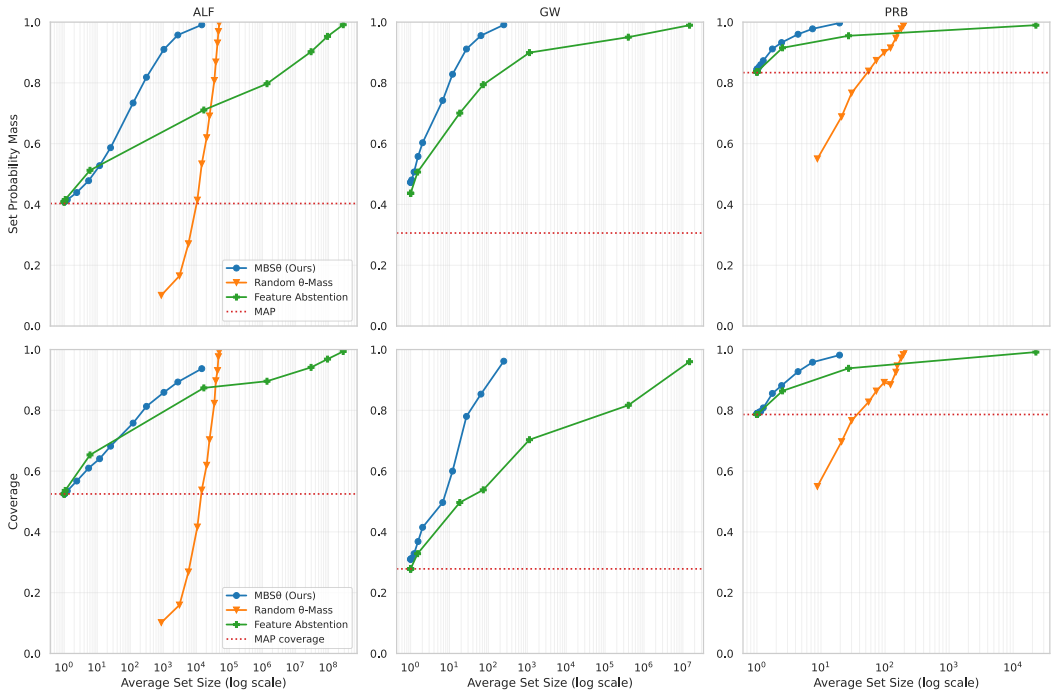


Figure 11: Set size tradeoffs for a wide range of θ values. (Bottom) Set size vs. coverage (probability of the true state being included in the belief set). (Top) Set size vs. attributed mass (the probability mass of the states in the belief set according to the task belief).

I Results across VLM backbones

The main text uses GPT-4.1 as the SSE backbone. To test whether the results depend on this choice, we re-ran Experiments 1 and 3 with five open-weight VLMs: Qwen3-VL-8B and Qwen3-VL-32B [33], and Gemma 4 E2B, E4B, and 31B [34]. All backbones share the prompting scheme, Platt calibration protocol, constraints, and planning budget used for GPT-4.1 (Section 4). The results in this appendix come from a single later run of the pipeline covering all six backbones, with Platt scaling re-fit on a held-out calibration split, and are computed on the common support of instances completed at every θ. The GPT-4.1 values differ from those of Tables 1 and 3.

Coverage and compactness. Table 4 reports true-state coverage and belief set size at $\theta = 0.9$ for every backbone, and Figure 12 shows the full coverage-compactness frontiers. In addition to the baselines of Experiment 1, we report *valid MAP*, the MAP state projected onto the constraint-satisfying states, i.e., MAP with the same strict constraints as MBS_θ. It does not close the coverage gap to MBS_θ on any backbone, so the belief set contributes beyond constraint filtering. Set sizes are interquartile means over instances, which are robust to the heavy tail of the feature-abstention sets. For GPT-4.1 only, the Feature Abstention thresholds τ are those calibrated for the earlier run of Table 3 and were not re-calibrated after the Platt re-fit, so the achieved mass does not track θ on GW and ALF. On GW at $\theta = 0.9$, τ exceeds every calibrated confidence, so all features are abstained. This affects that row of Table 4 and its curves in Figures 10 and 12, and Table 3 gives the comparison with matched thresholds. MBS_θ preserves its coverage-per-state advantage on all six backbones, and the advantage is often largest for the open-weight models, whose calibration is weaker and whose MAP estimates degrade most. For example, on PRB with Qwen3-VL-8B, MBS_θ reaches 0.81 coverage where MAP reaches 0.28. Large sets on poorly calibrated backbones are the method behaving as specified: when the marginals are diffuse, reaching the requested mass requires more states, and the set size is a legible signal of perception quality.

Downstream planning. Table 5 repeats Experiment 3 with Adaptive MBS Search (AMS) for every backbone. It adds an oracle that plans from the true state as a ceiling, and the AMS-Loose variant,

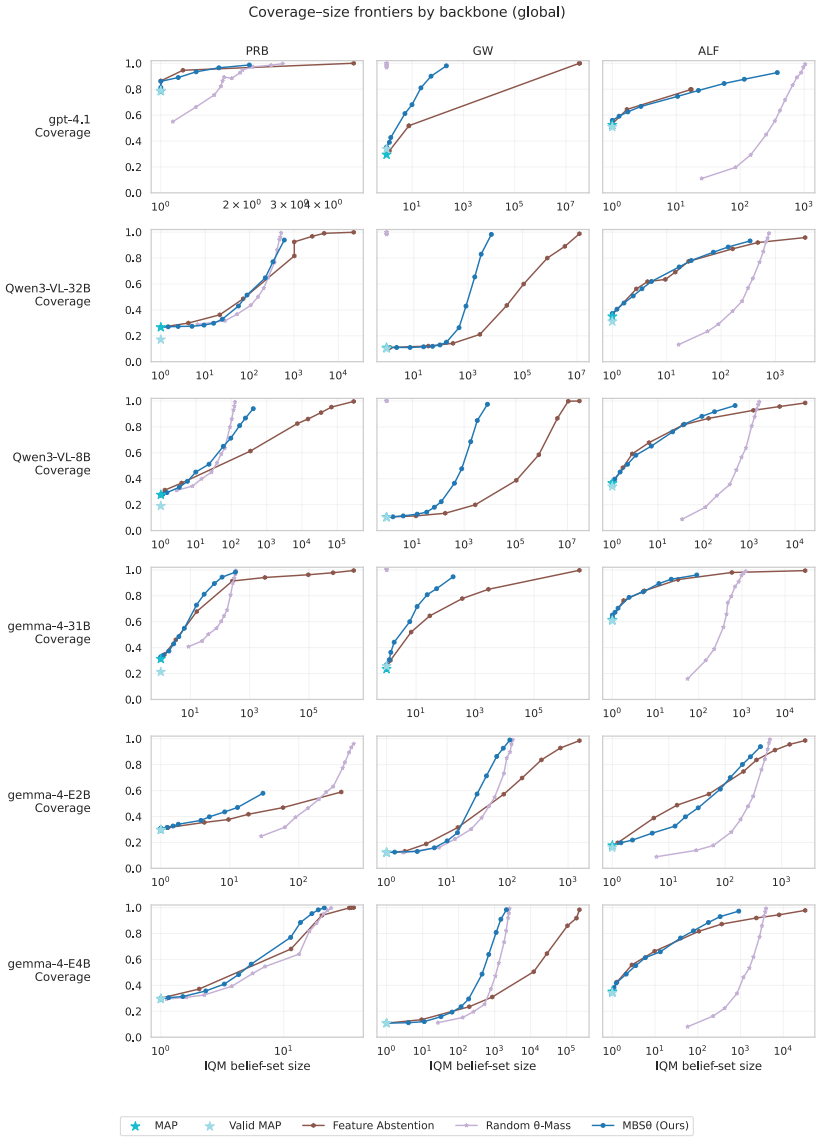


Figure 12: Coverage-compactness frontiers per VLM backbone (rows) and domain (columns) under global Platt calibration. Curves sweep θ, the horizontal axis is the interquartile mean belief set size on a log scale, and stars mark the θ-independent MAP and valid MAP points. Top left is better.

Table 4: Experiment 1 across VLM backbones at $\theta = 0.9$ under global Platt calibration. Cov. is the true-state coverage with its 95% bootstrap confidence interval over test instances and Size is the interquartile mean belief set size. Valid MAP is the MAP state projected onto the constraint-satisfying states. Higher coverage and lower size are better. The GPT-4.1 Feature Abstention row uses thresholds from the earlier run (see text).

		PRB		GW		ALF	
Backbone	Method	Cov. [95% CI]	Size	Cov. [95% CI]	Size	Cov. [95% CI]	Size
GPT-4.1	MBS_θ (Ours)	0.93 [0.92, 0.95]	1.35	0.81 [0.78, 0.84]	21.9	0.84 [0.83, 0.86]	56.1
	Valid MAP	0.79 [0.77, 0.81]	1	0.34 [0.30, 0.38]	1	0.51 [0.48, 0.53]	1
	MAP	0.79 [0.76, 0.81]	1	0.29 [0.26, 0.33]	1	0.52 [0.50, 0.55]	1
	Feature Abstention	0.86 [0.84, 0.88]	1	1.00 [1.00, 1.00]	3.36×10^7	0.80 [0.78, 0.82]	17
Qwen3-VL-32B	MBS_θ (Ours)	0.65 [0.61, 0.68]	226	0.65 [0.61, 0.69]	1.75×10^3	0.84 [0.83, 0.86]	72.1
	Valid MAP	0.17 [0.15, 0.19]	1	0.10 [0.08, 0.13]	1	0.31 [0.29, 0.33]	1
	MAP	0.27 [0.23, 0.30]	1	0.11 [0.08, 0.13]	1	0.35 [0.33, 0.37]	1
	Feature Abstention	0.97 [0.95, 0.98]	2.59×10^3	0.80 [0.77, 0.83]	8.12×10^5	0.87 [0.85, 0.89]	163
Qwen3-VL-8B	MBS_θ (Ours)	0.81 [0.78, 0.84]	171	0.69 [0.65, 0.73]	1.8×10^3	0.88 [0.87, 0.90]	91.4
	Valid MAP	0.19 [0.17, 0.21]	1	0.10 [0.08, 0.13]	1	0.34 [0.32, 0.36]	1
	MAP	0.28 [0.25, 0.31]	1	0.10 [0.08, 0.13]	1	0.37 [0.34, 0.39]	1
	Feature Abstention	0.91 [0.89, 0.93]	3.48×10^4	0.87 [0.84, 0.89]	4.07×10^6	0.93 [0.92, 0.94]	1.23×10^3
Gemma 4 31B	MBS_θ (Ours)	0.89 [0.87, 0.92]	64.1	0.81 [0.78, 0.84]	23.8	0.89 [0.88, 0.91]	12
	Valid MAP	0.21 [0.19, 0.23]	1	0.26 [0.22, 0.29]	1	0.61 [0.59, 0.63]	1
	MAP	0.31 [0.28, 0.35]	1	0.23 [0.20, 0.27]	1	0.61 [0.59, 0.64]	1
	Feature Abstention	0.96 [0.95, 0.98]	9.5×10^4	0.78 [0.74, 0.81]	366	0.93 [0.91, 0.94]	33.1
Gemma 4 E2B	MBS_θ (Ours)	0.44 [0.40, 0.46]	8.44	0.86 [0.83, 0.89]	66.2	0.80 [0.78, 0.82]	202
	Valid MAP	0.30 [0.27, 0.32]	1	0.12 [0.10, 0.15]	1	0.16 [0.14, 0.17]	1
	MAP	0.30 [0.27, 0.32]	1	0.12 [0.10, 0.15]	1	0.18 [0.16, 0.19]	1
	Feature Abstention	0.42 [0.39, 0.45]	18.6	0.84 [0.81, 0.86]	363	0.91 [0.90, 0.93]	760
Gemma 4 E4B	MBS_θ (Ours)	0.95 [0.94, 0.96]	17	0.81 [0.78, 0.84]	1.11×10^3	0.89 [0.87, 0.90]	177
	Valid MAP	0.30 [0.27, 0.32]	1	0.11 [0.09, 0.14]	1	0.34 [0.32, 0.36]	1
	MAP	0.30 [0.27, 0.32]	1	0.11 [0.08, 0.13]	1	0.35 [0.33, 0.37]	1
	Feature Abstention	1.00 [0.99, 1.00]	34.1	0.86 [0.83, 0.89]	1.06×10^5	0.92 [0.91, 0.93]	2.34×10^3

which uses only the disjoint selected constraint groups of Appendix G. AMS improves over MAP on all six PRB backbones and on four of six GW backbones. On the two remaining GW backbones, Gemma 4 E2B and E4B, the marginals are too diffuse to be informative: every method including MAP stays near 10% under the planning budget while the oracle remains at 95.0%, and AMS falls back to MAP-level performance. The GW gain grows substantially when perception is degraded yet still informative, e.g., AMS reaches 68.7% against 31.0% for MAP with Qwen3-VL-32B, which supports the headroom explanation of the modest GPT-4.1 gain on GW given in Section 4. The high timeout rates of AMS-Loose on the open-weight backbones arise because the disjoint groups prune far fewer of the $2^{|F|}$ completions, so the prefix search spends its budget on large candidate sets, which is exactly the failure that the full constraint set prevents.

Table 5: Experiment 3 across VLM backbones under global Platt calibration: downstream planning success rate with 95% bootstrap confidence intervals over the 1,200 PRB and 600 GW test instances. TO is the fraction of instances on which the method exhausted the planning budget, which count as failures. The oracle plans from the true state. AMS-Loose uses only the disjoint selected constraint groups, and AMS the full constraint set with the relaxed normalizer, as in Table 1.

		PRB		GW	
Backbone	Method	Success [95% CI]	TO	Success [95% CI]	TO
GPT-4.1	Oracle	100.0% [99.7, 100.0]	0%	95.0% [93.0, 96.5]	0%
	MAP	16.7% [14.7, 18.9]	0%	71.7% [67.9, 75.1]	0%
	Adaptive Abst.	16.8% [14.7, 19.0]	0%	74.0% [70.3, 77.3]	0%
	AMS-Loose	16.2% [14.2, 18.4]	0%	74.7% [71.0, 78.0]	1%
	AMS	32.4% [29.8, 35.1]	0%	79.2% [75.7, 82.2]	2%
Qwen3-VL-32B	Oracle	100.0% [99.7, 100.0]	0%	95.0% [93.0, 96.5]	0%
	MAP	18.3% [16.2, 20.6]	0%	31.0% [27.4, 34.8]	0%
	Adaptive Abst.	18.5% [16.4, 20.8]	0%	31.8% [28.2, 35.7]	0%
	AMS-Loose	17.0% [15.0, 19.2]	57%	31.2% [27.6, 35.0]	68%
	AMS	37.8% [35.1, 40.5]	0%	68.7% [64.8, 72.2]	6%
Qwen3-VL-8B	Oracle	100.0% [99.7, 100.0]	0%	95.0% [93.0, 96.5]	0%
	MAP	22.3% [20.1, 24.8]	0%	33.7% [30.0, 37.5]	0%
	Adaptive Abst.	22.8% [20.5, 25.3]	0%	35.8% [32.1, 39.7]	0%
	AMS-Loose	20.2% [18.1, 22.6]	52%	34.8% [31.1, 38.7]	64%
	AMS	55.4% [52.6, 58.2]	0%	65.3% [61.4, 69.0]	0%
Gemma 4 31B	Oracle	100.0% [99.7, 100.0]	0%	95.0% [93.0, 96.5]	0%
	MAP	35.0% [32.4, 37.7]	0%	66.2% [62.3, 69.8]	0%
	Adaptive Abst.	35.1% [32.4, 37.8]	0%	69.0% [65.2, 72.6]	0%
	AMS-Loose	29.3% [26.8, 32.0]	46%	69.3% [65.5, 72.9]	2%
	AMS	84.1% [81.9, 86.0]	0%	80.5% [77.1, 83.5]	0%
Gemma 4 E2B	Oracle	100.0% [99.7, 100.0]	0%	95.0% [93.0, 96.5]	0%
	MAP	24.8% [22.5, 27.4]	0%	10.2% [8.0, 12.8]	0%
	Adaptive Abst.	28.1% [25.6, 30.7]	0%	10.2% [8.0, 12.8]	0%
	AMS-Loose	19.8% [17.7, 22.2]	2%	10.2% [8.0, 12.8]	8%
	AMS	42.8% [40.1, 45.7]	0%	10.2% [8.0, 12.8]	6%
Gemma 4 E4B	Oracle	100.0% [99.7, 100.0]	0%	95.0% [93.0, 96.5]	0%
	MAP	17.6% [15.5, 19.8]	0%	11.5% [9.2, 14.3]	0%
	Adaptive Abst.	17.7% [15.6, 19.9]	0%	11.5% [9.2, 14.3]	0%
	AMS-Loose	16.8% [14.8, 19.1]	4%	10.3% [8.1, 13.0]	62%
	AMS	32.2% [29.7, 34.9]	0%	10.3% [8.1, 13.0]	52%

J Sensitivity to correlated perception errors

The belief converter of Section 2.3 assumes conditional independence between features. Task constraints restore the dominant task-level dependencies through mutex structure, but perceptual dependencies, e.g., a single lighting condition that corrupts several features at once, remain unmodeled. The coverage curves of Experiment 1 already absorb the real error structure of a VLM, since coverage is measured against ground truth. Here we isolate what conditional independence alone costs in a controlled study.

Protocol. For each test instance we take the real calibrated marginals p^0 and select a feature block that a shared nuisance would corrupt together: either the features of the largest mutex-coupled groups, the adversarial case since the constraint layer already couples them, or the least-decided features, the "shared lighting" case. We then define a correlated joint as a mixture driven by a latent nuisance event $E \sim \text{Bernoulli}(\pi)$,

$$P_{\text{true}}(s) = (1-\pi)\prod_{f\in F} p_f^0(s(f)) + \pi \prod_{f\in F} p_f^1(s(f)),$$

where p^1 equals p^0 outside the block and inside it each marginal is pulled toward its opposite, $p_f^1 = p_f^0 + \rho\,(1 - 2p_f^0)$, so that $\rho = 1$ mirrors the marginal. Conditioned on E, the features

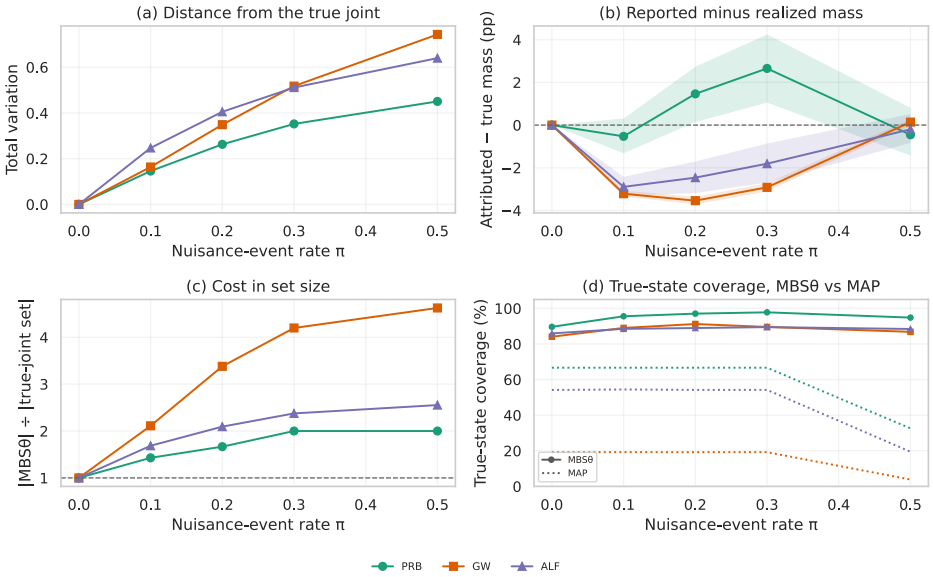


Figure 13: Correlated perception errors with the mutex-coupled block, $\rho = 1$, $\theta = 0.9$, and GPT-4.1 under global Platt calibration, as a function of the nuisance-event rate π. (a) Total variation between the product belief and the correlated joint. (b) Attributed minus true mass of the returned MBS_θ in percentage points, with 95% bootstrap confidence bands. (c) Size of the product-belief MBS_θ relative to the true joint's own minimal θ-mass set. (d) Ground-truth coverage of MBS_θ (solid) and MAP (dotted).

are independent, but marginally they are not. The estimator receives the mixture's exact marginals $(1-\pi)p^0 + \pi p^1$, so it is perfectly feature-calibrated by construction and the conditional-independence lift is the only remaining discrepancy. No ground truth enters the belief. We run PRaVaSS on the resulting product belief and score the returned set against the correlated joint and against a certified oracle, the true joint's own minimal θ-mass set over valid states, computed by a best-first search over both mixture components with a stopping certificate. We report five quantities: the total variation between the product belief and the joint, computed exactly on the affected block; the attributed minus true mass of the returned set, where the true mass is the exact expected coverage under the joint; the fraction of instances whose true mass still reaches θ; the set size relative to the oracle set; and the ground-truth coverage of each method. We sweep $\pi \in \{0, 0.1, 0.2, 0.3, 0.5\}$ with $\rho = 1$ at $\theta = 0.9$ for GPT-4.1 under global Platt calibration. The case $\pi = 0$ is the uncorrelated control, where the joint is the product belief and every distortion metric is exactly zero. Instances whose oracle set is not certified are dropped for every method, so all contrasts are paired.

Results. Figure 13 and Tables 6 to 8 summarize the sweep. Three findings follow. First, even when corruption drives the total variation between the product belief and the true joint up to 0.74, the gap between attributed and true set mass stays within 4.4 points across all domains, corruption rates, and block choices, and its sign varies across domains and block modes. The true mass can fall below the requested θ in the hardest cells, but the shortfall tracks this small gap rather than collapsing. Second, the cost of the independence assumption is set size rather than coverage: the product-belief set grows up to 4.6 times the certified true-joint set at the same mass target. Third, ground-truth coverage of the product-belief MBS_θ stays between 84% and 98% across the entire sweep, while MAP and valid MAP collapse to as low as 3.8% under heavy corruption. Under exactly this kind of correlated error, determinization collapses and the belief set absorbs the misspecification by growing, which is the design intent.

Scope. This is a synthetic sensitivity analysis, not an estimate of the natural error correlation of any VLM: π and ρ are swept rather than estimated from data, and the nuisance is a single shared binary event over one block. Learned inter-feature dependencies and distribution-free calibration of θ against coverage remain future work.

Table 6: Distortion of the product belief under correlated perception errors (mutex-coupled block, $\rho = 1$, $\theta = 0.9$, GPT-4.1). n is the number of paired instances, total variation is the distance between the product belief and the correlated joint, the mass gap is the attributed minus true mass of the returned MBS_θ in percentage points with 95% bootstrap confidence intervals, Reaches θ is the fraction of instances whose true mass still meets θ, and the last column is the size of the product-belief MBS_θ relative to the true joint's own minimal θ-mass set.

Domain	π	n	Total variation	Mass gap (pp)	Reaches θ	Size ratio
PRB	0.00	135	0.000	0.0 [0.0, 0.0]	100.0%	1.00×
	0.10	135	0.146	−0.5 [−1.3, +0.3]	87.4%	1.43×
	0.20	135	0.263	+1.5 [+0.1, +2.7]	67.4%	1.67×
	0.30	135	0.352	+2.7 [+1.1, +4.2]	75.6%	2.00×
	0.50	135	0.450	−0.5 [−1.4, +0.8]	85.2%	2.00×
GW	0.00	182	0.000	0.0 [0.0, 0.0]	100.0%	1.00×
	0.10	182	0.164	−3.2 [−3.3, −3.1]	100.0%	2.11×
	0.20	182	0.349	−3.5 [−3.7, −3.4]	100.0%	3.38×
	0.30	182	0.517	−2.9 [−3.1, −2.7]	100.0%	4.20×
	0.50	182	0.743	+0.1 [−0.0, +0.3]	50.0%	4.62×
ALF	0.00	192	0.000	0.0 [0.0, 0.0]	100.0%	1.00×
	0.10	191	0.247	−2.9 [−3.3, −2.4]	93.2%	1.69×
	0.20	190	0.405	−2.5 [−3.2, −1.7]	88.4%	2.09×
	0.30	190	0.511	−1.8 [−2.7, −0.9]	88.9%	2.38×
	0.50	190	0.640	−0.2 [−0.8, +0.5]	81.6%	2.55×

Table 7: Ground-truth coverage under correlated perception errors (mutex-coupled block, $\rho = 1$, $\theta = 0.9$, GPT-4.1), with the median returned set size in parentheses. The true-joint column is the oracle that knows the correlated joint.

Domain	π	n	MBS_θ (product belief)	MBS_θ (true joint)	Valid MAP	MAP
PRB	0.00	135	89.6% (2)	89.6% (2)	67.4% (1)	66.7% (1)
	0.10	135	95.6% (4)	91.1% (3)	67.4% (1)	66.7% (1)
	0.20	135	97.0% (5)	91.1% (3)	67.4% (1)	66.7% (1)
	0.30	135	97.8% (6)	91.1% (3)	67.4% (1)	66.7% (1)
	0.50	135	94.8% (8)	91.9% (3)	32.6% (1)	32.6% (1)
GW	0.00	182	84.1% (22)	84.1% (22)	25.8% (1)	19.2% (1)
	0.10	182	89.0% (43)	84.1% (22)	25.8% (1)	19.2% (1)
	0.20	182	91.2% (68.5)	84.1% (22)	25.8% (1)	19.2% (1)
	0.30	182	89.6% (76.5)	84.1% (22)	25.8% (1)	19.2% (1)
	0.50	182	86.8% (88)	84.1% (22)	3.8% (1)	3.8% (1)
ALF	0.00	192	85.9% (43)	85.9% (43)	54.2% (1)	54.2% (1)
	0.10	191	88.5% (76)	86.9% (44)	54.5% (1)	54.5% (1)
	0.20	190	88.9% (94)	86.8% (42.5)	54.2% (1)	54.2% (1)
	0.30	190	89.5% (118.5)	86.3% (43)	54.2% (1)	54.2% (1)
	0.50	190	88.4% (139)	86.3% (46)	19.5% (1)	19.5% (1)

Table 8: Mass gap (attributed minus true mass of the returned MBS_θ, percentage points, with 95% bootstrap confidence intervals) and fraction of instances whose true mass reaches θ, for the two corruption block modes ($\rho = 1$, $\theta = 0.9$, GPT-4.1).

		Mutex-coupled block		Least-decided block	
Domain	π	Mass gap (pp)	Reaches θ	Mass gap (pp)	Reaches θ
PRB	0.10	-0.5 [-1.3, $+0.3$]	87%	$+0.8$ [-0.1, $+1.8$]	74%
	0.20	$+1.5$ [$+0.1$, $+2.7$]	67%	$+3.8$ [$+2.4$, $+5.2$]	55%
	0.30	$+2.7$ [$+1.1$, $+4.2$]	76%	$+4.4$ [$+2.8$, $+6.4$]	65%
	0.50	-0.5 [-1.4, $+0.8$]	85%	-0.5 [-1.8, $+1.1$]	93%
GW	0.10	-3.2 [-3.3, -3.1]	100%	-1.5 [-1.9, -1.0]	90%
	0.20	-3.5 [-3.7, -3.4]	100%	-2.2 [-2.5, -1.7]	87%
	0.30	-2.9 [-3.1, -2.7]	100%	-2.3 [-2.7, -1.9]	88%
	0.50	$+0.1$ [-0.0, $+0.3$]	50%	-1.7 [-2.0, -1.3]	84%
ALF	0.10	-2.9 [-3.3, -2.4]	93%	$+1.2$ [$+0.7$, $+1.8$]	30%
	0.20	-2.5 [-3.2, -1.7]	88%	$+3.8$ [$+2.9$, $+4.6$]	27%
	0.30	-1.8 [-2.7, -0.9]	89%	$+2.3$ [$+1.2$, $+3.4$]	51%
	0.50	-0.2 [-0.8, $+0.5$]	82%	-0.1 [-0.9, $+0.7$]	65%

K Computational resources used

K.1 Worker type and hardware specifications

- Architecture & CPU: Internal HPC cluster using x86_64 architecture with dual-socket Intel(R) Xeon(R) 6710E CPUs (128 total cores per node, 64 cores per socket, 1 thread per core).
- Memory: 1 TB (1031821 MB) of total RAM available per compute node.
- Storage: 438 GB local NVMe storage (/dev/nvme0n1p2) per node, backed by network-attached storage (e.g., Vast Data NFS).

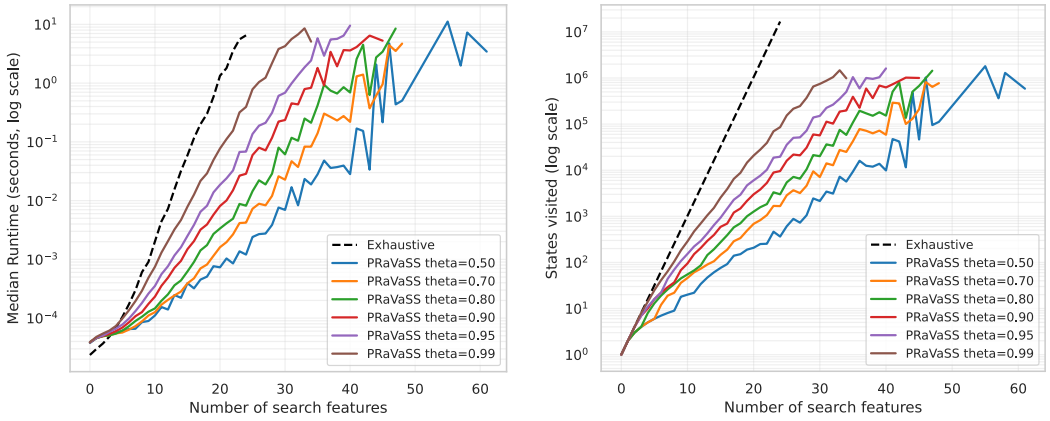
K.2 Compute and memory per experiment

- Execution Time: Individual CPU experiment runs ranged from less than a minute to a maximum wall-clock time of 10 hours.
- Memory Utilization: Peak memory usage (MaxRSS) varied significantly by job, ranging from 3.5 MB up to a maximum of 226.8 GB (221460920K).
- Open-weight VLM inference (Appendix I) is the only GPU workload. Each of the five open-weight backbones was run once over all evaluation splits as a single job allocated four NVIDIA L40 GPUs (48 GB each) on the same cluster, with the model loaded in bfloat16. Active inference wall-clock time per backbone ranged from roughly 30 hours (Qwen3-VL-8B) to 80 hours (Gemma 4 31B), about 270 hours in total, estimated from output timestamps. All subsequent calibration, MBS_θ construction, and planning experiments consume the cached predictions and run on CPU only, as for GPT-4.1.

L Full runtime results

Figure 14a and Figure 14b show the runtime and memory scaling of PRaVaSS as a function of the number of features considered in the search, for the PRB and ALF domains. The GW domain was omitted because it has a fixed number of features, and thus does not provide insight into feature-count scaling behavior. The runtime plot shows the growing gap between PRaVaSS and exhaustive enumeration as the number of search features increases, with PRaVaSS scaling much more gracefully. At $\theta = 0.99$, PRaVaSS can find the MBS_θ in problems with 42% more features than exhaustive enumeration, meaning that PRaVaSS can scale significantly just by forgoing 0.01 probability mass. This advantage grows up to 154% at $\theta = 0.5$, scaling significantly as we place more trust in the SSE's output. Similarly, the memory plot shows that PRaVaSS maintains significantly fewer states than exhaustive enumeration, and thus uses much less memory. The gap is significant even at the highest-coverage operating point of $\theta = 0.99$, showing that there is a big mass of low-probability states that PRaVaSS can safely ignore to achieve significant efficiency gains.

Table 9 reports runtime, memory, and scaling behavior of PRaVaSS over different θ values compared to exhaustive enumeration across all domains. Results show that exhaustive enumeration was unable to scale beyond 24 search features.



(a) Runtime vs. number of search features.

(b) States visited vs. number of search features.

Figure 14: Scaling behavior of PRaVaSS across test instances and θ values. (a) Runtime scaling as a function of the number of search features for PRB and ALF. (b) Memory scaling as a function of the number of search features for PRB and ALF. GW is omitted because it has a fixed number of features.

Table 9: Runtime and scaling behavior of PRaVaSS and exhaustive enumeration. Solved entries report counts with percentages in parentheses; bracketed values are 95% bootstrap confidence intervals using 10,000 paired resamples over test instances.

θ	Solved instances	Max search features solved	Median runtime (s)	Median states visited
0.50	3,789 (99.71%)	61	1.56×10^{-4} $[1.43\times10^{-4}, 1.67\times10^{-4}]$	16.00 $[12.48, 18.00]$
0.70	3,779 (99.45%)	48	1.14×10^{-4} $[1.02\times10^{-4}, 1.24\times10^{-4}]$	36.00 $[31.00, 43.52]$
0.80	3,774 (99.32%)	47	1.34×10^{-4} $[1.20\times10^{-4}, 1.47\times10^{-4}]$	45.00 $[43.00, 55.00]$
0.90	3,756 (98.84%)	45	1.80×10^{-4} $[1.49\times10^{-4}, 2.17\times10^{-4}]$	68.00 $[55.00, 86.26]$
0.95	3,737 (98.34%)	40	2.64×10^{-4} $[2.15\times10^{-4}, 3.09\times10^{-4}]$	106.00 $[82.00, 121.00]$
0.99	3,674 (96.68%)	34	4.32×10^{-4} $[3.38\times10^{-4}, 5.19\times10^{-4}]$	159.00 $[122.00, 186.00]$
Exhaustive	2,906 (76.47%)	24	1.66×10^{-4} $[1.31\times10^{-4}, 2.04\times10^{-4}]$	64.00 $[32.00, 64.00]$

M Extended results for Experiment 3

Experiment 3 used an iterative approach that we call AMS to find the largest θ for which a robust plan can be found from an MBS_θ produced by PRaVaSS. First, we run PRaVaSS with a large θ to find a large MBS_θ, where states are sorted by their probability mass in descending order. This is a side effect of how PRaVaSS expands different states throughout its search. With this large MBS_θ, we perform a binary search for the largest prefix of the sorted MBS_θ for which the CPOR planner finds some plan. The probability mass of the resulting MBS_θ is used as a confidence score for plan success. Similarly, adaptive abstention performs a binary search for an abstention threshold τ for which CPOR finds a plan from the feature-abstention belief set.

The AMS variant reported in Table 1 uses a more comprehensive set of constraints than in previous experiments. These constraints are not disjoint mutex groups, and thus an exact normalizer requires weighted model counting, which is exponential in the induced width in the worst case (Appendix G.4).

Table 10: Downstream planning success rate in the PRB and GW domains. Bracketed rows are 95% bootstrap confidence intervals using 10,000 paired resamples over test instances. Bold values indicate the best-performing method in each domain, underlined values indicate the second-best.

Method	PRB	GW
MAP	16.7%	70.7%
	[14.4%, 18.8%]	[67.2%, 73.8%]
Adaptive Abst.	<u>16.8%</u>	<u>73.3%</u>
	[14.7%, 19.1%]	[69.8%, 76.8%]
AMS-Loose	16.2%	73.2%
	[14.1%, 18.0%]	[69.5%, 76.3%]
AMS	**32.6%**	**80.0%**
	[30.1%, 35.3%]	[76.7%, 83.0%]

Table 11: Constraint-strength and normalizer ablation for GPT-4.1: downstream planning success rate with 95% bootstrap confidence intervals over the 1,200 PRB and 600 GW test instances. All methods receive the same calibrated marginals and, except AMS-Loose, the same strict constraint set. TO is the fraction of instances on which the method exhausted the planning budget, counted as failures.

	PRB		GW	
Method	Success [95% CI]	TO	Success [95% CI]	TO
MAP	16.7% [14.7, 18.9]	0%	71.7% [67.9, 75.1]	0%
Adaptive Abst.	16.8% [14.7, 19.0]	0%	74.0% [70.3, 77.3]	0%
AMS-Loose	16.2% [14.2, 18.4]	0%	74.7% [71.0, 78.0]	1%
AMS (strict, relaxed Z)	32.4% [29.8, 35.1]	0%	79.2% [75.7, 82.2]	2%
AMS (strict, exact Z)	32.3% [29.7, 35.0]	0%	79.8% [76.4, 82.8]	0%

Instead, we use a relaxed normalizer as suggested in Section 2.3. A comparison with AMS using looser restrictions is available in Table 10. In the GW domain, AMS-Loose achieves a 3.5% improvement over MAP, and matches adaptive abstention. In the PRB domain, AMS-Loose performs 3% worse than MAP and 3.6% worse than adaptive abstention. AMS, however, outperforms all other methods in both domains, achieving a 13% improvement over MAP in GW, and a 95% improvement in PRB. This shows the power of constraint-based belief set construction and the versatility of PRaVaSS, even in the presence of approximated probability masses obtained with a relaxed normalizer.

Constraint-strength and normalizer ablation. To separate the belief-set construction from the constraint information, Table 11 runs the identical AMS pipeline with the loose (disjoint selected groups) and strict (full) constraint sets of Appendix G, the latter under both the relaxed and the exact variable-elimination normalizer, alongside MAP and adaptive abstention fed the same calibrated marginals and the same strict constraints. The rows come from a single run of the Experiment 3 pipeline covering all backbones of Appendix I, so the GPT-4.1 values differ marginally from those of Table 1. Table 12 reports paired bootstrap contrasts over matched instances using 10,000 resamples. Every method except AMS-Loose receives the same constraint information, yet only the belief-set construction converts it into a large gain: with identical strict constraints, AMS exceeds adaptive abstention by 15.7 points on PRB and 5.2 points on GW, and exceeds its own loose-constraint variant by 16.2 and 4.5 points. The exact normalizer shifts success by at most 0.7 points, an order of magnitude below the constraint-layer effect, which confirms that the prefix search consumes only the ordering of states, which the relaxed normalizer preserves while its reported masses remain lower bounds (Corollary 2). Valid MAP in Appendix I, i.e., MAP with the same strict constraints, likewise does not close the coverage gap to MBS_θ. The constraints are extracted automatically by FD for PRB and GW, so the strict sets are not privileged hand-crafted information.

Figure 15 breaks down the results seen in Table 1 by horizon, showing the downstream planning success rate for each method as a function of the horizon of the test instance.

Table 13 shows how much performance is retained on solvable test instances where the most likely state is the true state (MAP-correct) vs. test instances where the most likely state is not the true state (MAP-wrong), for each method in the PRB and GW domains. We report the success rate of each

Table 12: Paired contrasts for the ablation of Table 11: difference in planning success (percentage points) on matched test instances, with 95% paired bootstrap confidence intervals using 10,000 resamples.

Domain	Contrast	Difference	95% CI
PRB	AMS (strict, relaxed Z) − MAP	+15.8	[+13.7, +17.9]
	AMS (strict, relaxed Z) − Adaptive Abst.	+15.7	[+13.6, +17.8]
	AMS (strict, relaxed Z) − AMS-Loose	+16.2	[+14.2, +18.4]
	AMS (strict, exact Z) − AMS (strict, relaxed Z)	−0.1	[−0.2, +0.0]
GW	AMS (strict, relaxed Z) − MAP	+7.5	[+5.3, +9.8]
	AMS (strict, relaxed Z) − Adaptive Abst.	+5.2	[+3.2, +7.3]
	AMS (strict, relaxed Z) − AMS-Loose	+4.5	[+2.7, +6.5]
	AMS (strict, exact Z) − AMS (strict, relaxed Z)	+0.7	[+0.2, +1.3]

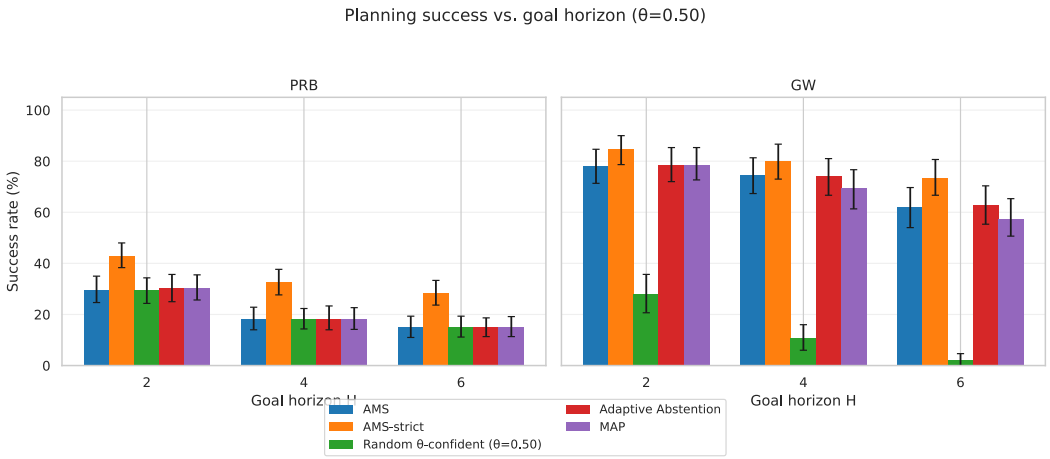


Figure 15: Downstream planning success rate breakdown by horizon in the PRB and GW domains. Error bars are 95% bootstrap confidence intervals using 10,000 paired resamples over test instances. In the legend, AMS-strict is AMS, AMS is AMS-Loose, and Random θ-confident is Random θ-Mass.

compared method on the MAP-correct and MAP-wrong strata, which we refer to as the *retention* and *recovery* performance of each method, respectively. In this table, we show performance for $\theta = 0.5$, which corresponds to the belief set containing the majority of the probability mass. Figure 16 shows the recovery performance of each method as a function of θ.

Table 13: Retention and recovery performance in the PRB and GW domains for $\theta = 0.5$. Bracketed values are 95% bootstrap confidence intervals using 10,000 paired resamples over test instances.

Domain	PRB		GW	
Stratum	MAP-correct (Retention)	MAP-wrong (Recovery)	MAP-correct (Retention)	MAP-wrong (Recovery)
(stratum size)	n=200	n=1000	n=424	n=146
MBS_θ-loose (Ours)	97.0% [94.7%, 99.0%]	0.0% [0.0%, 0.0%]	90.8% [87.4%, 93.4%]	4.8% [1.4%, 8.2%]
MBS_θ (Ours)	97.5% [95.5%, 99.3%]	5.7% [4.4%, 7.4%]	98.6% [97.4%, 99.5%]	21.2% [15.1%, 28.8%]
Rnd. θ-Mass	95.5% [92.5%, 98.0%]	0.0% [0.0%, 0.0%]	14.4% [11.2%, 17.7%]	0.0% [0.0%, 0.0%]
Feature Abstention	100.0% [100.0%, 100.0%]	0.0% [0.0%, 0.0%]	98.8% [97.6%, 99.8%]	1.4% [0.0%, 3.4%]
MAP	100.0% [100.0%, 100.0%]	0.0% [0.0%, 0.0%]	100.0% [100.0%, 100.0%]	0.0% [0.0%, 0.0%]

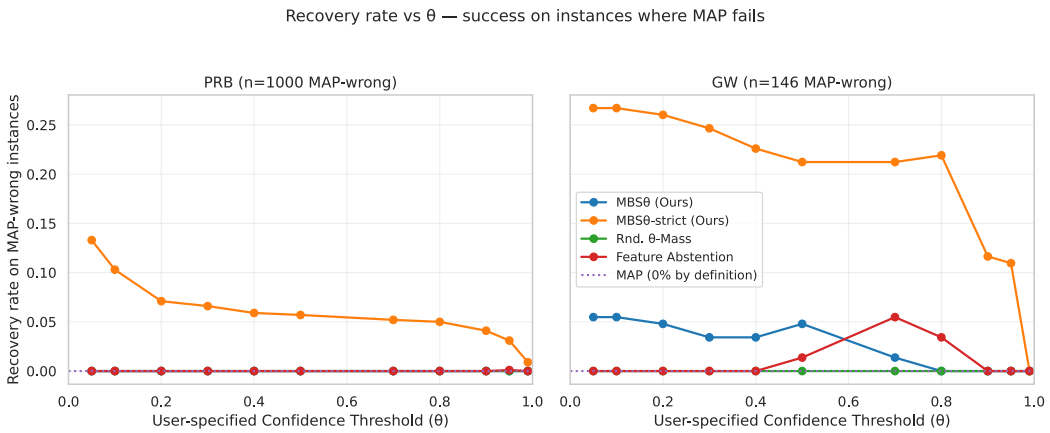


Figure 16: Recovery performance in the PRB and GW domains as a function of θ. In the legend, MBS$_\theta$-strict is MBS$_\theta$ and MBS$_\theta$ is MBS$_\theta$-loose of Table 13.

N Closed-loop planning with hidden features

The claims of this paper concern single-shot belief set construction, and ALF is excluded from Experiment 3 because robust decision-making there requires the agent to gather information. To test whether the advantage of AMS carries over to a closed loop, we ran an observe-update-replan study on ALF with GPT-4.1 as the backbone.

Protocol. Of the 2,000 ALF test tasks, 417 have between two and six declared non-visible, goal- or action-relevant uncertainty units with a constraint-valid alternative. A unit is either a Boolean feature or a categorical group that keeps its exactly-one constraint, such as the location of an object. We froze a seeded sample of 120 of them, of which 5 were excluded for every method because the closed-world initial state contained no true member of a declared location group, leaving 115 paired tasks. Every atom in the declared units is set to probability 0.5 and all other features are clamped to the same constraint-valid MAP assignment for every method, so the belief is a uniform prior over the legal completions of the hidden units. Each method plans (round $k = 0$), one shared relevant unit selected by a seeded uniform policy is then revealed noiselessly, the belief is updated for that unit only, the common constrained state ordering is recomputed, and each method replans ($k = 1$). Constraint-valid MAP (hereafter MAP), AMS, and adaptive abstention receive the identical uncertainty vocabulary, constraints, observation, exact normalizer, and 30 s per-planner-call budget, and an oracle that plans from the true state supplies a ceiling. Success means that the returned action sequence reaches the goal when executed from the true initial state, and planner failures remain in the denominator. Integrity checks found no method errors, timeouts, or denominator mismatches, and whenever MAP finds a plan, AMS and adaptive abstention retain at least that plan.

Results. Tables 14 and 15 and Figure 17 report the outcome. Before the observation, MAP succeeds on 72.2% of the tasks, adaptive abstention on 90.4%, AMS on 93.0%, and the oracle on 100%. After it, these rise to 80.0%, 93.0%, 96.5%, and 100%. AMS exceeds MAP by 20.9 percentage points before the observation (paired 95% CI $[13.9, 28.7]$) and by 16.5 points after it ($[9.6, 24.3]$). Its advantage over adaptive abstention is not significant before the observation ($+2.6$ points, $[-0.9, 7.0]$) and becomes significant after it ($+3.5$ points, $[0.9, 7.8]$). The observation recovers 9 MAP, 4 adaptive-abstention, and 4 AMS failures, with 0, 1, and 0 regressions, respectively, and it concentrates the belief: the median mass retained by the selected sets rises to approximately 100% after the observation at essentially unchanged set sizes. Adaptive abstention offers the better efficiency tradeoff, reaching near-AMS success with smaller sets and under half the search time, whereas AMS attains the highest success.

Scope. This is a controlled stress test under synthetic maximal uncertainty on the declared hidden units. It demonstrates a complete observe-update-replan loop with discriminative planning gains, and

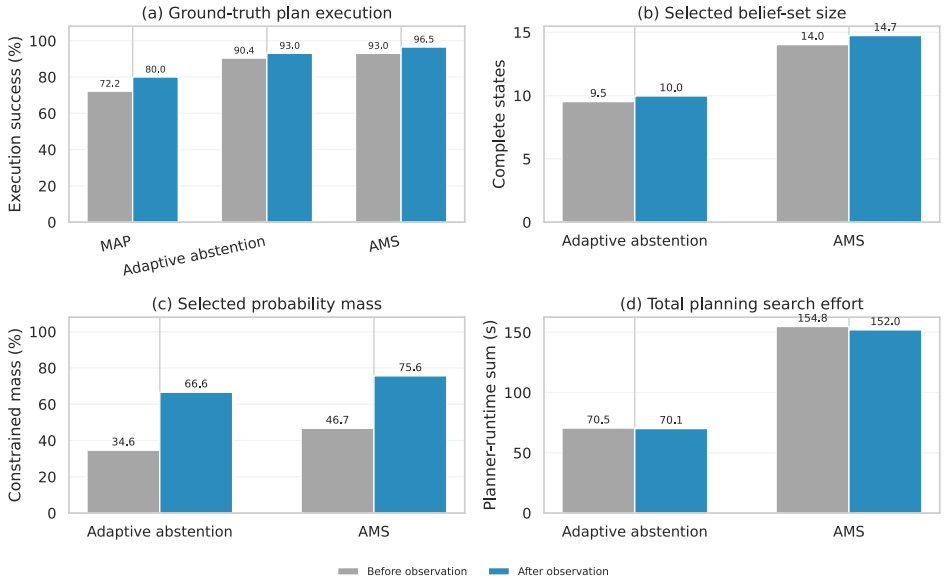


Figure 17: Closed-loop planning on ALF with GPT-4.1 under global Platt calibration (115 paired tasks), before and after one shared observation. (a) Ground-truth plan-execution success. (b) Mean number of complete states in the selected planning set. (c) Mean exact constrained mass of that set. (d) Mean sum of planner runtimes over all candidate attempts.

Table 14: Closed-loop planning on ALF with GPT-4.1 (115 paired tasks). Top: ground-truth plan-execution success before ($k = 0$) and after ($k = 1$) one shared observation; the oracle, which plans from the true state, succeeds on all 115 tasks in both rounds. Bottom: paired differences in percentage points with 95% paired bootstrap confidence intervals.

Method	Before observation	After observation	Change
MAP	83/115 (72.2%)	92/115 (80.0%)	+7.8
Adaptive Abst.	104/115 (90.4%)	107/115 (93.0%)	+2.6
AMS	107/115 (93.0%)	111/115 (96.5%)	+3.5
Paired contrast	**Before observation**	**After observation**	
AMS − MAP	+20.9 [+13.9, +28.7]	+16.5 [+9.6, +24.3]	
Adaptive Abst. − MAP	+18.3 [+12.2, +25.2]	+13.0 [+7.8, +19.1]	
AMS − Adaptive Abst.	+2.6 [−0.9, +7.0]	+3.5 [+0.9, +7.8]	

it does not estimate natural VLM uncertainty rates, simulate a physical information-gathering action, or solve many-step active sensing, which we leave to future work.

Table 15: Efficiency of the set-valued methods in the closed-loop study, mean (median) over the 115 paired tasks. Belief-set size is the number of complete states in the selected planning set, probability mass is the exact constrained mass of that set, search time is the sum of planner runtimes over all candidate attempts including the MAP fallback, and attempts is the mean number of planner calls.

Round	Method	Belief-set size	Probability mass	Search time	Attempts
Before observation	AMS	14.0 (20)	46.7% (45.5%)	154.8 s (186.4 s)	6.4
	Adaptive Abst.	9.5 (12)	34.6% (25.0%)	70.5 s (73.2 s)	3.6
After observation	AMS	14.7 (20)	75.6% (100.0%)	152.0 s (174.6 s)	6.5
	Adaptive Abst.	10.0 (12)	66.6% (100.0%)	70.1 s (72.9 s)	3.7

O Broader impact

This paper proposes a methodological tool for translating uncertain perception outputs into compact symbolic belief sets that can be consumed by classical reasoners. Its primary intended societal upside is improved robustness and safety of decision-making systems built on top of pretrained perception, particularly in robotic and embodied settings. By exposing perceptual ambiguity, the method lets downstream planners avoid actions that are unsafe or infeasible under genuinely ambiguous conditions, supporting the development of more trustworthy neurosymbolic systems.

We see two main avenues for potential negative impact. First, the method relies on assumptions that may be violated in deployment, including calibrated per-feature probability estimates, conditional independence in the belief converter, and faithful task constraint extraction. If these assumptions are violated, the resulting belief set may misrepresent the true uncertainty, which could be more harmful than acknowledged uncertainty in safety-critical applications. Second, formal correctness guarantees on the belief representation may encourage over-trust in downstream pipelines that still depend on the perception model, the constraint specification, and the planner. Practitioners deploying such systems should validate the calibration of the underlying perception model on the target domain, audit the extracted task constraints, and treat the probability mass parameter θ as a tunable risk budget rather than a guarantee of plan success.

Because the work is evaluated on benchmark and simulated task-planning domains rather than deployed robotic systems, we do not foresee a direct path to immediate harm, but we encourage future deployments to pair this representation with monitoring of perception calibration and validation of task constraints.

P Terms

belief A probability distribution over possible initial symbolic states representing uncertainty about the true initial state in conformant probabilistic planning. 1–7, 13, 39

belief set A set of possible initial symbolic states representing uncertainty about the true initial state in conformant planning. 2, 3, 6, 8–10, 20, 23–26, 32, 34, 36

factored belief A function that assigns independent probabilities to individual features, representing the marginal probability that each feature is true. 3, 4, 13, 39

symbolic state A symbolic representation of the world state in terms of features relevant to the planning task, typically expressed in a formal language such as STRIPS. 6, 39

task belief A belief computed by conditioning the unconstrained belief on task-specific constraints (see Equation (2)). 4–9, 13, 15, 20, 21, 24

unconstrained belief A belief computed via a factored belief by assuming conditional independence between features, before applying task-specific constraints (see Equation (1)). 4–7, 13, 14, 18, 39

Q Acronyms

AMS *Adaptive MBS Search*. An adaptive choosing of the MBS_θ via binary search for the largest subset for which there exists a plan. 9, 10, 24, 26, 27, 32–34, 36–38

FD *Fast Downward*. A planning system for classical planning problems. 4, 8, 10, 18–20, 33

MAP *Maximum A Posteriori*. The probability-maximizing state feature assignment. 8, 10, 22, 24–29, 33, 34, 36–38

MBS_θ *θ-confidence Minimal Belief Set*. The smallest subset of states whose cumulative probability meets or exceeds a given threshold (see Definition 2). 1–3, 5–10, 13, 15–17, 20, 21, 23, 24, 26, 28–33, 35, 36, 39, 40

PDDL *Planning Domain Definition Language*. A language for defining task planning problems. 4, 8, 16, 18

PRaVaSS *Probability-Ranked Valid State Search*. The algorithm that produces the MBS_θ. 1–3, 6, 7, 9, 10, 13, 15, 28, 31–33, 40

SSE *Symbolic State Estimator*. A function that provides probabilistic estimates of symbolic features given sensory observations. 3, 4, 7, 8, 10, 17, 20, 24, 31

VLM *Vision-Language Model*. A multimodal foundation model that combines text and image input. 1–4, 8, 9, 17, 18, 21, 22, 24–28, 31, 37

NeurIPS Paper Checklist

1. **Claims**

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

   Answer: [Yes]

   Justification: The abstract and introduction state the paper's main contributions: the MBS_θ representation, the PRaVaSS algorithm, correctness and complexity guarantees, and empirical evaluation in task-planning domains. These claims are supported by the formal development in Sections 2–3 and the experiments in Section 4.

   Guidelines:

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

2. **Limitations**

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

   Answer: [Yes]

   Justification: The paper discusses important limitations and assumptions, including the conditional independence assumption in the belief converter (with a controlled sensitivity study in Appendix J), the distinction between the attributed mass bounded by θ and true-state coverage (Section 3 and Appendix H), the exponential worst-case search behavior under constraints with the polynomial bound restricted to the unconstrained case, the single-shot scope of the claims (Appendix N), and future extensions to temporal refinement, learned dependencies, and closed-loop robotic execution.

   Guidelines:

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

judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

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

   Answer: [Yes]

   Justification: The paper states its theoretical results in Theorem 1, Corollary 1, and Proposition 1, and provides a proof sketch of Theorem 1 in the main text together with full proofs in Appendices B to D.

   Guidelines:

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

4. **Experimental result reproducibility**

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

   Answer: [Yes]

   Justification: Section 4 describes the domains, baselines, metrics, state estimator, calibration procedure, and constraint extraction, while Appendices E to H, K and L provide additional details on domains, data construction, constraints, compute, and full results. The paper also links the public repository containing all experiment code (`https://github.com/CLAIR-LAB-TECHNION/pravass_neurips2016`).

   Guidelines:

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

(c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).

(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

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

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

   Answer: [Yes]

   Justification: All experiment code is publicly available at `https://github.com/CLAIR-LAB-TECHNION/pravass_neurips2016`, with step-by-step instructions for reproducing every result, and the final dataset is linked from that repository. The appendices describe the domains, data construction, constraint extraction, and compute resources needed to reproduce the experiments.

   Guidelines:

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

6. **Experimental setting/details**

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

   Answer: [Yes]

   Justification: Section 4 specifies the evaluation domains, calibration setup, VLM backbone, constraint extraction procedure, and baseline inputs. Appendix F provides validation and test split sizes for PRB, GW, and ALF and lists the additional VLM backbones of Appendix I; Appendices J and N specify the protocols of the sensitivity and closed-loop studies.

   Guidelines:

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

7. **Experiment statistical significance**

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

   Answer: [Yes]

   Justification: Appendices H and L report 95% bootstrap confidence intervals for coverage, set size, mass, runtime, and states visited, and Appendices I, J, M and N report bootstrap confidence intervals, including paired contrasts, for the multi-backbone, sensitivity, ablation, and closed-loop results. These intervals accompany the main empirical comparisons and scaling results.

   Guidelines:

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

8. **Experiments compute resources**

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

   Answer: [Yes]

   Justification: Appendix K reports the compute environment, including the internal HPC cluster, CPU architecture, number of cores, RAM, local storage, execution-time range, memory-utilization range, and the hardware used for open-weight VLM inference. Experiment 2 in Section 4 and Appendix L further report runtime and memory-related scaling behavior.

   Guidelines:

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

9. **Code of ethics**

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

Answer: [Yes]

Justification: The work evaluates algorithms on benchmark or generated task-planning domains and does not report human-subject experiments, crowdsourcing, or collection of sensitive personal data. We are not aware of any aspect of the research that conflicts with the NeurIPS Code of Ethics.

Guidelines:

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

10. **Broader impacts**

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

Answer: [Yes]

Justification: Appendix O provides a broader impact statement covering positive impacts (safer task planning under uncertain perception) and negative impacts (risk of over-trust if the assumptions of calibrated perception, conditional independence, and faithful constraint extraction are violated in deployment).

Guidelines:

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

11. **Safeguards**

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

Answer: [N/A]

Justification: The paper does not release a high-risk pretrained model, image generator, language model, or scraped dataset. The released assets are experiment code and generated or benchmark-based task-planning data, so special misuse safeguards are not applicable.

Guidelines:

- The answer [N/A] means that the paper poses no such risks.

- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

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

    Answer: [Yes]

    Justification: The paper cites the existing datasets, tools, models, and libraries it uses, including PRB, GW/S3E, ALFWorld/ALFRED, Fast Downward, GPT-4.1, Qwen3-VL, and Gemma 4. The appendix explicitly states the licenses or terms of use for the evaluation domains, S3E, Fast Downward, GPT-4.1 API usage, and the open-weight VLMs.

    Guidelines:

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

13. **New assets**

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

    Answer: [Yes]

    Justification: The paper introduces experiment code and generated evaluation data, provides them with documentation in a public repository (`https://github.com/CLAIR-LAB-TECHNION/pravass_neurips2016`), and documents the data construction process in Appendix F. Since the data are generated from benchmark or simulated task domains, consent from human participants is not applicable.

    Guidelines:

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

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

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

Answer: [N/A]

Justification: The paper does not involve crowdsourcing or research with human subjects. The experiments use generated or benchmark-based task-planning instances and VLM-based state estimation.

Guidelines:

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

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

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

Answer: [N/A]

Justification: The paper does not involve crowdsourcing or research with human subjects. Therefore, IRB approval or equivalent review is not applicable.

Guidelines:

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

16. **Declaration of LLM usage**

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

Answer: [Yes]

Justification: The paper explicitly describes the use of a VLM as the symbolic state estimator and explains how token probabilities for answers such as "Yes" and "No" are normalized into per-feature probabilities. Section 4 further states that GPT-4.1 is used as the fixed VLM backbone in the main text, and Appendix I lists the five open-weight VLMs used in the multi-backbone study.

Guidelines:

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